Your title here

Generated by Doxygen 1.8.3.1

Thu May 8 2014 18:53:47

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 meow Namespace Reference

Classes

- class Color3_Space

  channel
- class HSL
- class HSLf
- class HSLf_Space

  Y(), ***U***(), ***V***()
- class HSV
- class HSVf
- class HSVf_Space

  Y(), ***U***(), ***V***()
- class RGB
- class RGBf
- class RGBi
- class RGBi_Space

  Red, ***Green***, ***Blue***
- class RGBf_Space

  Red, ***Green***, ***Blue***
- class YUV
- class YUVf
- class YUVf_Space

  Y(), ***U***(), ***V***()
- class BinaryIndexTree

  `SegmentTree`
- class DisjointSet
- class HashTableList

  keylisthash_table
- class KD_Tree

  `k-dimension` ***tree***
- class MergeableHeap

  `Maximum-Heap` , ***heap***, *`merge`*
- class SegmentTree
- class SplayTree

  , *Key->Value* . *`std::map`* , *`split`* , *`merge`* , *`keyOffset`*

- class SplayTree_Range

  SplayTree, , (operator `SegmentTree`)
- class VP_Tree

  KD_Tree
- class Vector2D

  2D's vector
- class Vector3D

  3D's vector
- class Bitmap

- struct SceneInfo
- class BundleAdjustment
- class BundleAdjustment_LM
- class Camera

  Camera.
- class Eye

  `Camera` offset transformation
- class FeaturePoint

- class FeaturePointsDetector
- class FeaturePointsDetector_Harris

  Harris corner detect.
- class FeaturePointsMatch
- class FeaturePointsMatch_K_Match
- class IdentityPoints

  `std::map<ID,`Vector<Scalar> >
- class Photo

- class ViewPort

- class WatchBall

  ***camera**, offset, rotation*
- class LinearTransformation

  A base class for implementing kinds of linear transformations.
- class Rotation3D

  Rotation a point/vector alone an axis with given angle in 3D world.
- class Matrix

  matrix
- class Transformation

  A base class for implementing kinds of transformations.
- class BallProjection

  A ball projection is to project the given vector to a hyper-sphere.
- class PhotoProjection

  *A **photo projection** is a kind of transformation that project point/vector to a flat **photo**.*
- class Vector

  vector
- class ObjArray

  `std::vector`, ObjBase
- class ObjBase

  Base, read, write, create, ...
- class ObjDictionary

std::map , ObjBase

- class ObjProperties

- class ObjSelector

  register, runtimestringnewclass
- class ObjType

  Type , ObjBase
- class ReaderWriter_int
- class ReaderWriter_size_t
- class ReaderWriter_double
- class ReaderWriter_string
- class ImplementInterface
- class RegisterInterface
- class Self

  A little class use for packing the data part of another class. With this technique, it can achieve Copy-On-Write(COR) mechanism at background and have a reference mechanism which much more flexible then the one C++ has.
- class Usage

  , usage document, argc, argv
- struct PairToPair

  .from.first, .from.second, .to.first, .to.second

Typedefs

- typedef PairToPair< size_t,
  size_t, size_t, size_t > FeaturePointIndexPair
- typedef std::vector
  < FeaturePointIndexPair > FeaturePointIndexPairs
- typedef ObjType< int,
  ReaderWriter_int > ObjInt
- typedef ObjType< size_t,
  ReaderWriter_size_t > ObjSizeT
- typedef ObjType< double,
  ReaderWriter_double > ObjDouble
- typedef ObjType< std::string,
  ReaderWriter_string > ObjString

Enumerations

- enum SceneInfoFlags { CAN_OFFSET = 0x01, CAN_ROTATE = 0x02, CAN_ZOOM = 0x04 }

Functions

- template<class RGB_T , class HSL_T >
  void RGB_to_HSL (RGB< RGB_T > const &rgb, HSL< HSL_T > *hsl)
- template<class HSL_T , class RGB_T >
  void HSL_to_RGB (HSL< HSL_T > const &hsl, RGB< RGB_T > *rgb)
- template<class YUV_T , class HSL_T >
  void YUV_to_HSL (YUV< YUV_T > const &yuv, HSL< HSL_T > *hsl)
- template<class HSL_T , class YUV_T >
  void HSL_to_YUV (HSL< HSL_T > const &hsl, YUV< YUV_T > *yuv)
- void colorTransformate (RGBf_Space const &rgb, HSLf_Space *hsl)

  RGBf_Space to HSLf_Space
- void colorTransformate (YUVf_Space const &yuv, HSLf_Space *hsl)

  YUVf_Space to HSLf_Space
- void colorTransformate (HSLf_Space const &hsl, RGBf_Space *rgb)

  HSLf_Space to RGBf_Space
- void colorTransformate (HSLf_Space const &hsl, YUVf_Space ∗yuv)

  HSLf_Space to YUVf_Space
- void colorTransformate (HSLf_Space const &hsl, RGBi_Space ∗rgb)

  HSLf_Space to RGBi_Space
- void colorTransformate (RGBi_Space const &rgb, HSLf_Space ∗hsl)

  RGBi_Space to HSLf_Space
- template<class RGB_T , class HSV_T >
  void RGB_to_HSV (RGB< RGB_T > const &rgb, HSV< HSV_T > ∗hsv)
- template<class HSV_T , class RGB_T >
  void HSV_to_RGB (HSV< HSV_T > const &hsv, RGB< RGB_T > ∗rgb)
- template<class YUV_T , class HSV_T >
  void YUV_to_HSV (YUV< YUV_T > const &yuv, HSV< HSV_T > ∗hsv)
- template<class HSV_T , class YUV_T >
  void HSV_to_YUV (HSV< HSV_T > const &hsv, YUV< YUV_T > ∗yuv)
- template<class HSL_T , class HSV_T >
  void HSL_to_HSV (HSL< HSL_T > const &hsl, HSV< HSV_T > ∗hsv)
- template<class HSV_T , class HSL_T >
  void HSV_to_HSL (HSV< HSV_T > const &hsv, HSL< HSL_T > ∗hsl)
- void colorTransformate (RGBf_Space const &rgb, HSVf_Space ∗hsv)

  RGBf_Space to HSVf_Space
- void colorTransformate (YUVf_Space const &yuv, HSVf_Space ∗hsv)

  YUVf_Space to HSVf_Space
- void colorTransformate (HSLf_Space const &hsl, HSVf_Space ∗hsv)

  HSLf_Space to HSVf_Space
- void colorTransformate (HSVf_Space const &hsv, RGBf_Space ∗rgb)

  HSVf_Space to RGBf_Space
- void colorTransformate (HSVf_Space const &hsv, YUVf_Space ∗yuv)

  HSVf_Space to YUVf_Space
- void colorTransformate (HSVf_Space const &hsv, HSLf_Space ∗hsl)

  HSVf_Space to HSLf_Space
- void colorTransformate (HSVf_Space const &hsv, RGBi_Space ∗rgb)

  HSVf_Space to RGBi_Space
- void colorTransformate (RGBi_Space const &rgb, HSVf_Space ∗hsv)

  RGBi_Space to HSVf_Space
- void colorTransformate (RGBi_Space const &a, RGBf_Space ∗b)

  RGBi_Space to RGBf_Space
- void colorTransformate (RGBf_Space const &a, RGBi_Space ∗b)

  RGBf_Space to RGBi_Space
- template<class RGB_T , class YUV_T >
  void RGB_to_YUV (RGB< RGB_T > const &rgb, YUV< YUV_T > ∗yuv)
- template<class YUV_T , class RGB_T >
  void YUV_to_RGB (YUV< YUV_T > const &yuv, RGB< RGB_T > ∗rgb)
- void colorTransformate (RGBf_Space const &rgb, YUVf_Space ∗yuv)

  RGBf_Space to YUVf_Space
- void colorTransformate (YUVf_Space const &yuv, RGBf_Space ∗rgb)

  YUVf_Space to RGBf_Space
- void colorTransformate (RGBi_Space const &rgb, YUVf_Space ∗yuv)

  RGBi_Space to YUVf_Space
- void colorTransformate (YUVf_Space const &yuv, RGBi_Space ∗rgb)

  YUVf_Space to RGBi_Space

- template<class Data , class WeightingClass >
  std::vector< Data > ransac (std::vector< Data > const &data, WeightingClass const &w, size_t N, double p0, double P)

  *Run the **RANSAC** method to approach the best solution.*
- template<class Scalar , class Function >
  Vector< Scalar > levenbergMarquardt (Function const &f, Vector< Scalar > const &init, int counter=-1)
- template<class Scalar , class Function >
  Vector< Scalar > levenbergMarquardtTraining (Function &f, Vector< Scalar > const &init, Scalar const &init_mu, Scalar const &mu_pow, Scalar const &er_max, int retry_number, int counter)
- template<class T >
  T noEPS (T value, T eps=1e-9)

  abs() < eps, 0,
- template<class T >
  T normalize (T lower, T upper, T value)

  (value-lower)/(upper-lower)
- template<class T >
  T denormalize (T lower, T upper, T _ratio)

  (lower+_ratio∗(upper-lower))
- template<class T >
  T ratioMapping (T l1, T u1, T m1, T l2, T u2)

  `denormalize(l2,u2,normalize(l1,u1,m1))`
- template<class T >
  T inRange (T const &mn, T const &mx, T const &v)

  `std::min(mx,std::max(mn,v))`
- template<class T >
  T isInRange (T const &mn, T const &mx, T const &x)

  (mn <= x && x <= mx)
- template<class T >
  T squ (T const &x)

  `x*x`
- template<class T >
  T cub (T const &x)

  `x*x*x`
- template<class T >
  double average (T const &beg, T const &end, double sigs)

  `sigs`
- template<class T >
  double average (T const &beg, T const &end, T const &p, double sigs)

  `sigs, p`
- template<class T >
  T tAbs (T const &t)

- std::string stringPrintf (char const ∗fmt,...)

  Cprintf, `std::string`
- std::string stringReplace (std::string str, std::string const &from, std::string const &to)

  patternpattern
- bool cstringEndWith (char const ∗str, int n,...)

  patterns
- void debugPrintf_ (char const ∗file, char const ∗func, size_t line, char const ∗msg)
- void messagePrintf (int level_change, char const ∗fmt,...)

- bool filenameCompare (std::string const &f1, std::string const &f2)

- void debugPrintf_ (char const ∗file, char const ∗func, int32_t line, char const ∗msg)
- void messagePrintf (int32_t level_change, char const ∗fmt,...)
- double noEPS (double value, double eps)
- double normalize (double lower, double upper, double value)
- double denormalize (double lower, double upper, double ratio)
- double ratioMapping (double l1, double u1, double m1, double l2, double u2)

Variables

- static const double PI = 3.1415926535897932384626433832795028

  ...
- static const size_t kGlobalSeletorID = 0

5.1.1 Typedef Documentation

typedef PairToPair<size_t, size_t, size_t, size_t> meow::FeaturePointIndexPair

Definition at line 13 of file FeaturePointsMatch.h.

typedef std::vector<FeaturePointIndexPair> meow::FeaturePointIndexPairs

Definition at line 14 of file FeaturePointsMatch.h.

typedef ObjType<double , ReaderWriter_double> meow::ObjDouble

Definition at line 196 of file ObjTypes.h.

typedef ObjType<int , ReaderWriter_int > meow::ObjInt

Definition at line 194 of file ObjTypes.h.

typedef ObjType<size_t , ReaderWriter_size_t> meow::ObjSizeT

Definition at line 195 of file ObjTypes.h.

typedef ObjType<std::string, ReaderWriter_string> meow::ObjString

Definition at line 197 of file ObjTypes.h.

5.1.2 Enumeration Type Documentation

enum meow::SceneInfoFlags

Enumerator

CAN_OFFSET

CAN_ROTATE

CAN_ZOOM

Definition at line 10 of file BundleAdjustment.h.

5.1.3 Function Documentation

template<class T > double meow::average (T const & *beg*, T const & *end*, double *sigs*) `[inline]`

`sigs`

Definition at line 83 of file utility.h.

template<class T > double meow::average (T const & *beg*, T const & *end*, T const & *p*, double *sigs*) `[inline]`

`sigs`, `p`
Definition at line 110 of file utility.h.

void meow::colorTransformate (RGBf_Space const & *rgb*, YUVf_Space ∗ *yuv*) `[inline]`

`RGBf_Space` to `YUVf_Space`
Definition at line 84 of file YUV_Space.h.

void meow::colorTransformate (RGBf_Space const & *rgb*, HSLf_Space ∗ *hsl*) `[inline]`

`RGBf_Space` to `HSLf_Space`
Definition at line 85 of file HSL_Space.h.

void meow::colorTransformate (RGBf_Space const & *rgb*, HSVf_Space ∗ *hsv*) `[inline]`

`RGBf_Space` to `HSVf_Space`
Definition at line 86 of file HSV_Space.h.

void meow::colorTransformate (YUVf_Space const & *yuv*, RGBf_Space ∗ *rgb*) `[inline]`

`YUVf_Space` to `RGBf_Space`
Definition at line 99 of file YUV_Space.h.

void meow::colorTransformate (YUVf_Space const & *yuv*, HSVf_Space ∗ *hsv*) `[inline]`

`YUVf_Space` to `HSVf_Space`
Definition at line 109 of file HSV_Space.h.

void meow::colorTransformate (YUVf_Space const & *yuv*, HSLf_Space ∗ *hsl*) `[inline]`

`YUVf_Space` to `HSLf_Space`
Definition at line 109 of file HSL_Space.h.

void meow::colorTransformate (RGBi_Space const & *rgb*, YUVf_Space ∗ *yuv*) `[inline]`

`RGBi_Space` to `YUVf_Space`
Definition at line 114 of file YUV_Space.h.

void meow::colorTransformate (HSLf_Space const & *hsl*, HSVf_Space ∗ *hsv*) `[inline]`

`HSLf_Space` to `HSVf_Space`
Definition at line 118 of file HSV_Space.h.

void meow::colorTransformate (HSLf_Space const & *hsl*, RGBf_Space ∗ *rgb*) `[inline]`

`HSLf_Space` to `RGBf_Space`
Definition at line 118 of file HSL_Space.h.

void meow::colorTransformate (YUVf_Space const & *yuv*, RGBi_Space ∗ *rgb*) `[inline]`

`YUVf_Space` to `RGBi_Space`
Definition at line 123 of file YUV_Space.h.

void meow::colorTransformate (HSVf_Space const & *hsv*, RGBf_Space ∗ *rgb*) `[inline]`

`HSVf_Space` to `RGBf_Space`
Definition at line 127 of file HSV_Space.h.

void meow::colorTransformate (RGBi_Space const & *a*, RGBf_Space ∗ *b*) `[inline]`

`RGBi_Space` to `RGBf_Space`

Definition at line 149 of file RGB_Space.h.

void meow::colorTransformate (HSVf_Space const & *hsv*, YUVf_Space ∗ *yuv*) `[inline]`

`HSVf_Space` to `YUVf_Space`

Definition at line 151 of file HSV_Space.h.

void meow::colorTransformate (RGBf_Space const & *a*, RGBi_Space ∗ *b*) `[inline]`

`RGBf_Space` to `RGBi_Space`

Definition at line 159 of file RGB_Space.h.

void meow::colorTransformate (HSLf_Space const & *hsl*, YUVf_Space ∗ *yuv*) `[inline]`

`HSLf_Space` to `YUVf_Space`

Definition at line 160 of file HSL_Space.h.

void meow::colorTransformate (HSVf_Space const & *hsv*, HSLf_Space ∗ *hsl*) `[inline]`

`HSVf_Space` to `HSLf_Space`

Definition at line 160 of file HSV_Space.h.

void meow::colorTransformate (HSLf_Space const & *hsl*, RGBi_Space ∗ *rgb*) `[inline]`

`HSLf_Space` to `RGBi_Space`

Definition at line 169 of file HSL_Space.h.

void meow::colorTransformate (HSVf_Space const & *hsv*, RGBi_Space ∗ *rgb*) `[inline]`

`HSVf_Space` to `RGBi_Space`

Definition at line 169 of file HSV_Space.h.

void meow::colorTransformate (RGBi_Space const & *rgb*, HSLf_Space ∗ *hsl*) `[inline]`

`RGBi_Space` to `HSLf_Space`

Definition at line 179 of file HSL_Space.h.

void meow::colorTransformate (RGBi_Space const & *rgb*, HSVf_Space ∗ *hsv*) `[inline]`

`RGBi_Space` to `HSVf_Space`

Definition at line 179 of file HSV_Space.h.

bool meow::cstringEndWith (char const ∗ *str*, int *n*, ...) `[inline]`

patterns

Parameters

`in`	*str*	
`in`	*n*	pattern
`in`	...	pattern

Returns

`true/false`

Note

`cstring, char const*`

Definition at line 81 of file utility.h.

template<class T > T meow::cub (T const & *x*) `[inline]`

`x*x*x`

Definition at line 75 of file utility.h.

void meow::debugPrintf_ (char const * *file*, char const * *func*, int32_t *line*, char const * *msg*) `[inline]`

Definition at line 48 of file utility.hpp.

void meow::debugPrintf_ (char const * *file*, char const * *func*, size_t *line*, char const * *msg*) `[inline]`

Definition at line 109 of file utility.h.

template<class T > T meow::denormalize (T *lower*, T *upper*, T *_ratio*) `[inline]`

(lower+_ratio*(upper-lower))

Definition at line 35 of file utility.h.

double meow::denormalize (double *lower*, double *upper*, double *ratio*) `[inline]`

Definition at line 87 of file utility.hpp.

bool meow::filenameCompare (std::string const & *f1*, std::string const & *f2*) `[inline]`

a1 < a2 < a3 < a10 < a12 < a20, a1 < a10 < a12 < a2 < a20 < a3

Parameters

`in`	*f1*	
`in`	*f2*	

Returns

`true/false` **f1f2**

Definition at line 178 of file utility.h.

template<class HSL_T , class HSV_T > void meow::HSL_to_HSV (HSL< HSL_T > const & *hsl*, HSV< HSV_T > * *hsv*) `[inline]`

Definition at line 110 of file HSV.hpp.

template<class HSL_T , class RGB_T > void meow::HSL_to_RGB (HSL< HSL_T > const & *hsl*, RGB< RGB_T > * *rgb*) `[inline]`

Definition at line 74 of file HSL.hpp.

template<class HSL_T , class YUV_T > void meow::HSL_to_YUV (HSL< HSL_T > const & *hsl*, YUV< YUV_T > * *yuv*) `[inline]`

Definition at line 121 of file HSL.hpp.

template<class HSV_T , class HSL_T > void meow::HSV_to_HSL (HSV< HSV_T > const & *hsv*, HSL< HSL_T > * *hsl*) `[inline]`

Definition at line 117 of file HSV.hpp.

template<class HSV_T , class RGB_T > void meow::HSV_to_RGB (HSV< HSV_T > const & *hsv*, RGB< RGB_T > * *rgb*) `[inline]`

Definition at line 74 of file HSV.hpp.

template<class HSV_T , class YUV_T > void meow::HSV_to_YUV (HSV< HSV_T > const & *hsv,* YUV< YUV_T > ∗ *yuv*) `[inline]`

Definition at line 103 of file HSV.hpp.

template<class T > T meow::inRange (T const & *mn,* T const & *mx,* T const & *v*) `[inline]`

`std::min(mx,std::max(mn,v))`

Definition at line 51 of file utility.h.

template<class T > T meow::isInRange (T const & *mn,* T const & *mx,* T const & *x*) `[inline]`

(mn <= x && x <= mx)

Definition at line 59 of file utility.h.

template<class Scalar , class Function > Vector<Scalar> meow::levenbergMarquardt (Function const & *f,* Vector< Scalar > const & *init,* int *counter = −1*) `[inline]`

Definition at line 163 of file methods.h.

template<class Scalar , class Function > Vector<Scalar> meow::levenbergMarquardtTraining (Function & *f,* Vector< Scalar > const & *init,* Scalar const & *init_mu,* Scalar const & *mu_pow,* Scalar const & *er_max,* int *retry_number,* int *counter*) `[inline]`

Definition at line 183 of file methods.h.

void meow::messagePrintf (int32_t *level_change,* char const ∗ *fmt,* *...*) `[inline]`

Definition at line 57 of file utility.hpp.

void meow::messagePrintf (int *level_change,* char const ∗ *fmt,* *...*) `[inline]`

printf, , :

```
message1(level = 0)
  message2(level = 1)
    information1(level = 2)
    information2(level = 2)
  ... ok(for message2)
  message3(level = 1) ... ok
  information3(level = 1)
  message4(level = 1)
    message5(level = 2) ... ok
    message6(level = 2) ... ok
    information4(level = 2)
  ... ok(for message4)
... ok(for message5)
```

Parameters

`in`	*level_change*	: • == 0, information • == 1, message, level++ • == -1, level++message

in	*fmt,...*	printf

Returns

Definition at line 145 of file utility.h.

template<class T > T meow::noEPS (T *value*, T *eps = 1e−9*) [inline]

abs() < eps, 0,

Definition at line 18 of file utility.h.

double meow::noEPS (double *value*, double *eps*) [inline]

Definition at line 79 of file utility.hpp.

template<class T > T meow::normalize (T *lower*, T *upper*, T *value*) [inline]

(value-lower)/(upper-lower)

Definition at line 27 of file utility.h.

double meow::normalize (double *lower*, double *upper*, double *value*) [inline]

Definition at line 83 of file utility.hpp.

template<class Data , class WeightingClass > std::vector<Data> meow::ransac (std::vector< Data > const & *data*, WeightingClass const & *w*, size_t *N*, double *p0*, double *P*) [inline]

Run the **RANSAC** method to approach the best solution.

RANdom SAmple Consensus is an iterative method to estimate parameters of a mathematical model from a set of observed data which contains `outliers`.

Each iterator it will choose a subset of elements, the smallest set which can form a valid parameters, from the data set. And then calculate how many elements in the whole data set is inliers. After iterator much times, we just say the best solution is the parameters that has the much inliers elements in whole iterators.

Assume:

- We need at least N element to form a valid parameters.
- The probability of choosing a right element from data set each time is p_0.
- We want the probability of our solution actually being the best solution be P.
- We need to iterator M times.

Then we can estimate the number of iterations M :

$$(1-p_0^N)^M \le (1-P) \Rightarrow M\log(1-p_0^N) \le \log(1-P) \Rightarrow M \ge \frac{\log(1-p)}{\log(1-p_0^N)}, \quad \because (1-p_0^N < 1 \Rightarrow \log(1-p_0^N) < 0)$$

So in this function we choose $M = \lceil \frac{\log(1-P)}{\log(1-p_0^N)} \rceil$

Parameters

in	*data*	The whole data sett
in	*w*	Weight function to give a floating number for a given parameters which means how best this solution is. Negitave number means invalid parameters.

`in`	*N*	N, defined above
`in`	*p0*	p_0, defined above
`in`	*P*	P, defined above

Returns

solution.

Author

cat_leopard

Definition at line 58 of file methods.h.

template<class T > T meow::ratioMapping (T *l1,* T *u1,* T *m1,* T *l2,* T *u2*) `[inline]`

`denormalize(l2,u2,normalize(l1,u1,m1))`

Definition at line 43 of file utility.h.

double meow::ratioMapping (double *l1,* double *u1,* double *m1,* double *l2,* double *u2*) `[inline]`

Definition at line 91 of file utility.hpp.

template<class RGB_T , class HSL_T > void meow::RGB_to_HSL (RGB< RGB_T > const & *rgb,* HSL< HSL_T > * *hsl*) `[inline]`

Definition at line 52 of file HSL.hpp.

template<class RGB_T , class HSV_T > void meow::RGB_to_HSV (RGB< RGB_T > const & *rgb,* HSV< HSV_T > * *hsv*) `[inline]`

Definition at line 53 of file HSV.hpp.

template<class RGB_T , class YUV_T > void meow::RGB_to_YUV (RGB< RGB_T > const & *rgb,* YUV< YUV_T > * *yuv*) `[inline]`

Definition at line 47 of file YUV.hpp.

template<class T > T meow::squ (T const & *x*) `[inline]`

`x*x`

Definition at line 67 of file utility.h.

std::string meow::stringPrintf (char const * *fmt,* ...) `[inline]`

Cprintf, `std::string`

Parameters

`in`	*fmt,...*	printf

Returns

`std::string`

Warning

8191

Definition at line 42 of file utility.h.

std::string meow::stringReplace (std::string *str,* std::string const & *from,* std::string const & *to*) `[inline]`

patternpattern

Parameters

in	*str*	
in	*from*	pattern
in	*to*	pattern

Returns

Warning

,

Definition at line 60 of file utility.h.

template< class T > T meow::tAbs (T const & *t*) `[inline]`

Definition at line 141 of file utility.h.

template< class YUV_T , class HSL_T > void meow::YUV_to_HSL (YUV< YUV_T > const & *yuv*, HSL< HSL_T > * *hsl*) `[inline]`

Definition at line 114 of file HSL.hpp.

template< class YUV_T , class HSV_T > void meow::YUV_to_HSV (YUV< YUV_T > const & *yuv*, HSV< HSV_T > * *hsv*) `[inline]`

Definition at line 96 of file HSV.hpp.

template< class YUV_T , class RGB_T > void meow::YUV_to_RGB (YUV< YUV_T > const & *yuv*, RGB< RGB_T > * *rgb*) `[inline]`

Definition at line 60 of file YUV.hpp.

5.1.4 Variable Documentation

const size_t meow::kGlobalSeletorID = 0 `[static]`

Definition at line 209 of file ObjSelector.h.

const double meow::PI = 3.14159265358979323846264338327950288 `[static]`

...

Definition at line 12 of file utility.h.

Chapter 6

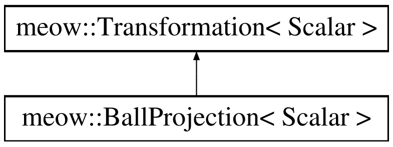
Class Documentation

6.1 meow::BallProjection< Scalar > Class Template Reference

A ball projection is to project the given vector to a hyper-sphere.

`#include "Transformations.h"`

Inheritance diagram for meow::BallProjection< Scalar >:


meow::Transformation< Scalar >
meow::BallProjection< Scalar >

Public Member Functions

- BallProjection (BallProjection const &b)
- BallProjection (size_t d)
- BallProjection (size_t d, Scalar const &r)
- BallProjection & copyFrom (BallProjection const &b)

  Copy settings from another one.
- BallProjection & referenceFrom (BallProjection const &b)

  Reference settings from another one.
- Scalar parameter (size_t i) const

  same as `radius()`
- Scalar parameter (size_t i, Scalar const &s)

  same as `radius(s)`
- Scalar radius () const

  Return the value of the radius.
- Scalar radius (Scalar const &r)

  Setup the radius.
- size_t dimension () const

  Get the dimension of this projection.
- Matrix< Scalar > transformate (Matrix< Scalar > const &x) const

  Project the input vector(s) onto the hyper-sphere and return it.
- Matrix< Scalar > jacobian (Matrix< Scalar > const &x) const

  Return the jacobian matrix (derivate by the input vector) of this projection.
- Matrix< Scalar > jacobian (Matrix< Scalar > const &x, size_t i) const

  Return the jacobian matrix (derivate by radius) of this projection.
- BallProjection & operator= (BallProjection const &b)

  Same as `copyFrom(b)`

- Matrix< Scalar > operator() (Matrix< Scalar > const &v) const

  Same as `transformate(v)`

Additional Inherited Members

6.1.1 Detailed Description

template<class Scalar>class meow::BallProjection< Scalar >

A ball projection is to project the given vector to a hyper-sphere.
Assume:

- The dimension of a ball projection is N
- The radius of the hyper-sphere is R

Then the transformation is like below:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ . \\ . \\ . \\ x_N \end{bmatrix} \overset{transformate}{\rightarrow} \begin{bmatrix} \frac{x_1 \times R}{L} \\ \frac{x_2 \times R}{L} \\ \frac{x_3 \times R}{L} \\ . \\ . \\ . \\ \frac{x_N \times R}{L} \end{bmatrix}$$

where $L = \sqrt{x_1^2 + x_2^2 + x_3^2 + ... + x_N^2}$

Author

cat_leopard

Definition at line 50 of file Transformations.h.

6.1.2 Constructor & Destructor Documentation

template<class Scalar> meow::BallProjection< Scalar >::BallProjection (BallProjection< Scalar > const & *b*) `[inline]`

Constructor, copy settings from given BallProjection
Parameters

in	*b*	another ball projection class

Definition at line 70 of file Transformations.h.

template<class Scalar> meow::BallProjection< Scalar >::BallProjection (size_t *d*) `[inline]`

Constructor and setup, radius = 1
Parameters

in	*d*	Dimension of the input/output vector

Definition at line 78 of file Transformations.h.

template<class Scalar> meow::BallProjection< Scalar >::BallProjection (size_t *d*, Scalar const & *r*) `[inline]`

Constructor and setup

Parameters

`in`	*d*	Dimension of the input/output vector
`in`	*r*	Radius of the hyper-sphere

Definition at line 88 of file Transformations.h.

6.1.3 Member Function Documentation

template<class Scalar> BallProjection& meow::BallProjection< Scalar >::copyFrom (BallProjection< Scalar > const & *b*) `[inline]`

Copy settings from another one.

Parameters

`in`	*b*	Another one

Returns

`*this`

Definition at line 98 of file Transformations.h.

template<class Scalar> size_t meow::BallProjection< Scalar >::dimension ( ) const `[inline]`

Get the dimension of this projection.

Definition at line 150 of file Transformations.h.

template<class Scalar> Matrix<Scalar> meow::BallProjection< Scalar >::jacobian (Matrix< Scalar > const & *x*) const `[inline]`, `[virtual]`

Return the jacobian matrix (derivate by the input vector) of this projection.

This method only allow a vector-like matrix be input. Assume:

- The dimension of a ball projection is N
- The length of the input vector is $L = \sqrt{x_1^2 + x_2^2 + ... + x_N^2}$
- The radius of the hyper-sphere is R

Then the jacobian matrix is like below:

$$\frac{R}{L^3} \times \begin{bmatrix} L^2 - x_1^2 & -x_1x_2 & -x_1x_3 & \dots & -x_1x_N \\ -x_2x_1 & L^2 - x_2^2 & -x_2x_3 & \dots & -x_2x_N \\ -x_3x_1 & -x_3x_2 & L^2 - x_3^2 & \dots & -x_3x_N \\ \vdots & \vdots & \vdots & & \vdots \\ -x_Nx_1 & -x_Nx_2 & -x_Nx_3 & \dots & L^2 - x_N^2 \end{bmatrix}$$

Parameters

`in`	*x*	The input matrix.

Returns

The output matrix.

Reimplemented from meow::Transformation< Scalar >.

Definition at line 213 of file Transformations.h.

template<class Scalar> Matrix<Scalar> meow::BallProjection< Scalar >::jacobian (Matrix< Scalar > const & *x*, size_t *i*) const `[inline], [virtual]`

Return the jacobian matrix (derivate by radius) of this projection.

This method only allow a vector-like matrix be input. Assume:

- The dimension of a ball projection is N
- The length of the input vector is $L=\sqrt{x_1^2+x_2^2+...+x_N^2}$
- The radius of the hyper-sphere is R

Then the jacobian matrix is like below:

$$R\times\begin{bmatrix}\frac{x_1}{L}\\\frac{x_2}{L}\\\frac{x_3}{L}\\.\\.\\.\\\frac{x_N}{L}\end{bmatrix}$$

Parameters

in	*x*	The input matrix.
in	*i*	Useless parameter

Returns

The output matrix.

Reimplemented from meow::Transformation< Scalar >.

Definition at line 258 of file Transformations.h.

template<class Scalar> Matrix<Scalar> meow::BallProjection< Scalar >::operator() (Matrix< Scalar > const & *v*) const `[inline]`

Same as `transformate(v)`

Definition at line 277 of file Transformations.h.

template<class Scalar> BallProjection& meow::BallProjection< Scalar >::operator= (BallProjection< Scalar > const & *b*) `[inline]`

Same as `copyFrom(b)`

Definition at line 270 of file Transformations.h.

template<class Scalar> Scalar meow::BallProjection< Scalar >::parameter (size_t *i*) const `[inline], [virtual]`

same as `radius()`

Implements meow::Transformation< Scalar >.

Definition at line 118 of file Transformations.h.

template<class Scalar> Scalar meow::BallProjection< Scalar >::parameter (size_t *i*, Scalar const & *s*) `[inline], [virtual]`

same as `radius(s)`

Implements meow::Transformation< Scalar >.

Definition at line 125 of file Transformations.h.

template<class Scalar> Scalar meow::BallProjection< Scalar >::radius ( ) const `[inline]`

Return the value of the radius.

Definition at line 132 of file Transformations.h.

template<class Scalar> Scalar meow::BallProjection< Scalar >::radius (Scalar const & *r*) `[inline]`

Setup the radius.

Parameters

in	*r*	New value of the radius

Returns

New radius

Definition at line 142 of file Transformations.h.

template<class Scalar> BallProjection& meow::BallProjection< Scalar >::referenceFrom (BallProjection< Scalar > const & *b*) `[inline]`

Reference settings from another one.

Parameters

in	*b*	Another one

Returns

`*this`

Definition at line 109 of file Transformations.h.

template<class Scalar> Matrix<Scalar> meow::BallProjection< Scalar >::transformate (Matrix< Scalar > const & *x*) const `[inline]`, `[virtual]`

Project the input vector(s) onto the hyper-sphere and return it.

If the number of columns of the input matrix is larger than 1, this method will think that you want to transform multiple vector once and the number of columns of the output matrix will be the same of the number of columns of the input one.

Parameters

in	*x*	The input matrix.

Returns

The output matrix.

Note

Take into account that too much safty checking will lead to inefficient, this method will not checking whether the dimension of the input vector/matrix is right. So be sure the data is valid before you call this method.

Implements meow::Transformation< Scalar >.

Definition at line 170 of file Transformations.h.

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

- meowpp/math/Transformations.h

6.2 meow::BinaryIndexTree< Value > Class Template Reference

`SegmentTree`

`#include "BinaryIndexTree.h"`

Public Member Functions

- BinaryIndexTree ()

  constructor
- BinaryIndexTree (size_t size, Value const &value)

  constructor

- BinaryIndexTree (BinaryIndexTree const &tree2)

  constructor
- void reset (size_t size, Value const &init)

  ,
- void update (size_t index, Value const &value)

  array index ()element
- Value query (ssize_t index) const

  0∼index

6.2.1 Detailed Description

template<class Value>class meow::BinaryIndexTree< Value >

`SegmentTree`

, , , **update()** . , *Value* `operator+` `std::max`(...)

Author

cat_leopard

Definition at line 21 of file BinaryIndexTree.h.

6.2.2 Constructor & Destructor Documentation

template<class Value > meow::BinaryIndexTree< Value >::BinaryIndexTree ( ) `[inline]`

constructor

Definition at line 28 of file BinaryIndexTree.h.

template<class Value > meow::BinaryIndexTree< Value >::BinaryIndexTree (size_t *size*, Value const & *value*) `[inline]`

constructor

Parameters

in	*size*	[0,size)
in	*value*	

Definition at line 37 of file BinaryIndexTree.h.

template<class Value > meow::BinaryIndexTree< Value >::BinaryIndexTree (BinaryIndexTree< Value > const & *tree2*) `[inline]`

constructor

`BinaryIndexTree`

Parameters

in	*tree2*	`BinaryIndexTree`

Definition at line 47 of file BinaryIndexTree.h.

6.2.3 Member Function Documentation

template<class Value > Value meow::BinaryIndexTree< Value >::query (ssize_t *index*) const `[inline]`

0∼index

O(logN)

Parameters

in	*index*	index

Returns

Definition at line 90 of file BinaryIndexTree.h.

template< class Value > void meow::BinaryIndexTree< Value >::reset (size_t *size*, Value const & *init*) `[inline]`

,

O(N)

Parameters

in	*size*	[0,size)
in	*init*	

Returns

Definition at line 60 of file BinaryIndexTree.h.

template< class Value > void meow::BinaryIndexTree< Value >::update (size_t *index*, Value const & *value*) `[inline]`

array *index* ()element

O(logN)

Parameters

in	*index*	index
in	*value*	

Returns

Definition at line 74 of file BinaryIndexTree.h.

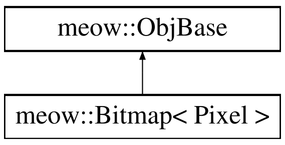
The documentation for this class was generated from the following file:

- meowpp/dsa/BinaryIndexTree.h

6.3 meow::Bitmap< Pixel > Class Template Reference

`#include "Bitmap.h"`

Inheritance diagram for meow::Bitmap< Pixel >:



Public Member Functions

- Bitmap ()

    constructor, Bitmap
- Bitmap (Bitmap const &b)

    constructor, bitmap
- Bitmap (size_t h, size_t w, Pixel const &p)

    constructor, , `Pixel`
- ∼Bitmap ()

    destructor
- Bitmap & copyFrom (Bitmap const &b)

- Bitmap & referenceFrom (Bitmap const &b)

    reference
- void reset (size_t h, size_t w, Pixel const &p)

- void clear ()

    ,
- size_t height () const

- size_t width () const

- size_t size () const

- size_t height (size_t h2, Pixel const &p)

- size_t width (size_t w2, Pixel const &p)

- size_t size (size_t h2, size_t w2, Pixel const &p)

- Pixel pixel (size_t y, size_t x) const

    (y, x) pixel
- Pixel pixel (size_t y, size_t x, Pixel const &p)

    (y, x) pixel
- void pixels (ssize_t yFirst, ssize_t yLast, ssize_t xFirst, ssize_t xLast, Pixel const &p)

- Matrix< Pixel > const & matrix () const

- Matrix< Pixel > & matrixGet ()

    (non-constant form)
- Matrix< Pixel > const & matrix (Matrix< Pixel > const &p)

- Bitmap gaussian (double radiusY, double radiusX) const

- Bitmap< Pixel > & gaussianed (double radiusY, double radiusX)

- Bitmap< Pixel > gradianceX (double radiusY, double radiusX) const

    x
- Bitmap< Pixel > & gradiancedX (double radiusY, double radiusX)

    x
- Bitmap< Pixel > gradianceY (double radiusY, double radiusX) const

y

- Bitmap< Pixel > & gradiancedY (double radiusY, double radiusX)

    y
- Bitmap & operator= (Bitmap const &b)

    same as `copyFrom(b)`
- Pixel operator() (size_t y, size_t x) const

    same as `pixel(y, x)`
- Pixel const & operator() (size_t y, size_t x, Pixel const &p) const

    same as `pixel(y, x, p)`
- bool write (FILE ∗f, bool bin, unsigned int fg) const
- bool read (FILE ∗f, bool bin, unsigned int fg)
- ObjBase ∗ create () const

    new
- ObjBase ∗ copyFrom (ObjBase const ∗b)
- char const ∗ ctype () const

    classtype
- std::string type () const

    classtype

Additional Inherited Members

6.3.1 Detailed Description

template<class Pixel>class meow::Bitmap< Pixel >

Author

cat_leopard

Definition at line 23 of file Bitmap.h.

6.3.2 Constructor & Destructor Documentation

template<class Pixel> meow::Bitmap< Pixel >::Bitmap ( ) `[inline]`

constructor, Bitmap
Definition at line 81 of file Bitmap.h.

template<class Pixel> meow::Bitmap< Pixel >::Bitmap (Bitmap< Pixel > const & *b*) `[inline]`

constructor, bitmap
Definition at line 87 of file Bitmap.h.

template<class Pixel> meow::Bitmap< Pixel >::Bitmap (size_t *h*, size_t *w*, Pixel const & *p*) `[inline]`

constructor, , `Pixel`

Parameters

in	*h*	
in	*w*	

in	*p*	pixel

Definition at line 97 of file Bitmap.h.

template<class Pixel> meow::Bitmap< Pixel >::~Bitmap ( ) `[inline]`

destructor

Definition at line 103 of file Bitmap.h.

6.3.3 Member Function Documentation

template<class Pixel> void meow::Bitmap< Pixel >::clear ( ) `[inline]`

,

Definition at line 137 of file Bitmap.h.

template<class Pixel> Bitmap& meow::Bitmap< Pixel >::copyFrom (Bitmap< Pixel > const & *b*) `[inline]`

Definition at line 109 of file Bitmap.h.

template<class Pixel> ObjBase∗ meow::Bitmap< Pixel >::copyFrom (ObjBase const ∗ *b*) `[inline]`, `[virtual]`

`ObjBase const* Bitmap`. method`copyFrom`

Parameters

in	*b*	

Returns

this

Reimplemented from meow::ObjBase.

Definition at line 405 of file Bitmap.h.

template<class Pixel> ObjBase∗ meow::Bitmap< Pixel >::create ( ) const `[inline]`, `[virtual]`

new

Returns

newBitmap<Pixel>

Reimplemented from meow::ObjBase.

Definition at line 392 of file Bitmap.h.

template<class Pixel> char const∗ meow::Bitmap< Pixel >::ctype ( ) const `[inline]`, `[virtual]`

classtype

Returns

`char const* typename`

Reimplemented from meow::ObjBase.

Definition at line 413 of file Bitmap.h.

template<class Pixel> Bitmap meow::Bitmap< Pixel >::gaussian (double *radiusY*, double *radiusX*) const `[inline]`

Parameters

`in`	*radiusY*	Ysigma
`in`	*radiusX*	Xsigma

Returns

`Bitmap`,

Definition at line 266 of file Bitmap.h.

template<class Pixel> Bitmap<Pixel>& meow::Bitmap< Pixel >::gaussianed (double *radiusY*, double *radiusX*) `[inline]`

Parameters

`in`	*radiusY*	Ysigma
`in`	*radiusX*	Xsigma

Returns

∗this

Definition at line 278 of file Bitmap.h.

template<class Pixel> Bitmap<Pixel>& meow::Bitmap< Pixel >::gradiancedX (double *radiusY*, double *radiusX*) `[inline]`

x

Parameters

`in`	*radiusY*	Ysigma
`in`	*radiusX*	Xsigma

Returns

∗this

Definition at line 302 of file Bitmap.h.

template<class Pixel> Bitmap<Pixel>& meow::Bitmap< Pixel >::gradiancedY (double *radiusY*, double *radiusX*) `[inline]`

y

Parameters

`in`	*radiusY*	Ysigma
`in`	*radiusX*	Xsigma

Returns

∗this

Definition at line 325 of file Bitmap.h.

template<class Pixel> Bitmap<Pixel> meow::Bitmap< Pixel >::gradianceX (double *radiusY*, double *radiusX*) const `[inline]`

x

Parameters

in	*radiusY*	Ysigma
in	*radiusX*	Xsigma

Returns

`Bitmap`,

Definition at line 290 of file Bitmap.h.

template<class Pixel> Bitmap<Pixel> meow::Bitmap< Pixel >::gradianceY (double *radiusY*, double *radiusX*) const [inline]

y

Parameters

in	*radiusY*	Ysigma
in	*radiusX*	Xsigma

Returns

`Bitmap`,

Definition at line 313 of file Bitmap.h.

template<class Pixel> size_t meow::Bitmap< Pixel >::height ( ) const [inline]

Definition at line 144 of file Bitmap.h.

template<class Pixel> size_t meow::Bitmap< Pixel >::height (size_t *h2*, Pixel const & *p*) [inline]

Parameters

in	*h2*	
in	*p*	, pixel

Returns

Definition at line 169 of file Bitmap.h.

template<class Pixel> Matrix<Pixel> const& meow::Bitmap< Pixel >::matrix ( ) const [inline]

Definition at line 240 of file Bitmap.h.

template<class Pixel> Matrix<Pixel> const& meow::Bitmap< Pixel >::matrix (Matrix< Pixel > const & *p*) [inline]

Definition at line 254 of file Bitmap.h.

template<class Pixel> Matrix<Pixel>& meow::Bitmap< Pixel >::matrixGet ( ) [inline]

(non-constant form)
Definition at line 247 of file Bitmap.h.

template<class Pixel> Pixel meow::Bitmap< Pixel >::operator() (size_t *y*, size_t *x*) const [inline]

same as `pixel(y, x)`
Definition at line 339 of file Bitmap.h.

template<class Pixel> Pixel const& meow::Bitmap< Pixel >::operator() (size_t *y*, size_t *x*, Pixel const & *p*) const `[inline]`

same as `pixel(y, x, p)`

Definition at line 346 of file Bitmap.h.

template<class Pixel> Bitmap& meow::Bitmap< Pixel >::operator= (Bitmap< Pixel > const & *b*) `[inline]`

same as `copyFrom(b)`

Definition at line 332 of file Bitmap.h.

template<class Pixel> Pixel meow::Bitmap< Pixel >::pixel (size_t *y*, size_t *x*) const `[inline]`

(y, x) pixel

Parameters

`in`	*y*	
`in`	*x*	

Returns

pixel constant reference

Definition at line 203 of file Bitmap.h.

template<class Pixel> Pixel meow::Bitmap< Pixel >::pixel (size_t *y*, size_t *x*, Pixel const & *p*) `[inline]`

(y, x) pixel

Parameters

`in`	*y*	
`in`	*x*	
`in`	*p*	

Returns

pixel constant reference

Definition at line 215 of file Bitmap.h.

template<class Pixel> void meow::Bitmap< Pixel >::pixels (ssize_t *yFirst*, ssize_t *yLast*, ssize_t *xFirst*, ssize_t *xLast*, Pixel const & *p*) `[inline]`

Parameters

`in`	*yFirst*	y()
`in`	*yLast*	y()
`in`	*xFirst*	x()
`in`	*xLast*	x()
`in`	*p*	

Returns

Definition at line 231 of file Bitmap.h.

template<class Pixel> bool meow::Bitmap< Pixel >::read (FILE ∗ *f*, bool *bin*, unsigned int *fg*) `[inline]`, `[virtual]`

Note

, fg

Reimplemented from meow::ObjBase.
Definition at line 373 of file Bitmap.h.

template<class Pixel> Bitmap& meow::Bitmap< Pixel >::referenceFrom (Bitmap< Pixel > const & *b*) `[inline]`

reference
Definition at line 117 of file Bitmap.h.

template<class Pixel> void meow::Bitmap< Pixel >::reset (size_t *h*, size_t *w*, Pixel const & *p*) `[inline]`

Parameters

`in`	*h*	
`in`	*w*	
`in`	*p*	pixel

Returns

Definition at line 130 of file Bitmap.h.

template<class Pixel> size_t meow::Bitmap< Pixel >::size ( ) const `[inline]`

Definition at line 158 of file Bitmap.h.

template<class Pixel> size_t meow::Bitmap< Pixel >::size (size_t *h2*, size_t *w2*, Pixel const & *p*) `[inline]`

Parameters

`in`	*h2*	
`in`	*w2*	
`in`	*p*	or, pixel

Returns

size

Definition at line 192 of file Bitmap.h.

template<class Pixel> std::string meow::Bitmap< Pixel >::type ( ) const `[inline]`, `[virtual]`

classtype
Returns

`std::string` typename

Reimplemented from meow::ObjBase.
Definition at line 421 of file Bitmap.h.

template<class Pixel> size_t meow::Bitmap< Pixel >::width ( ) const `[inline]`

Definition at line 151 of file Bitmap.h.

template<class Pixel> size_t meow::Bitmap< Pixel >::width (size_t *w2*, Pixel const & *p*) `[inline]`

Parameters

in	*w2*	
in	*p*	, pixel

Returns

Definition at line 180 of file Bitmap.h.

template<class Pixel> bool meow::Bitmap< Pixel >::write (FILE * *f*, bool *bin*, unsigned int *fg*) const `[inline]`, `[virtual]`

Note

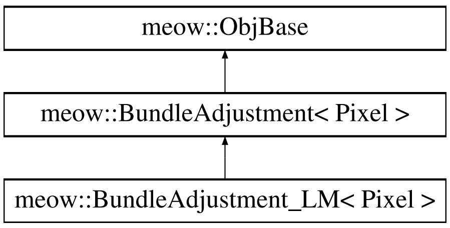
, fg

Reimplemented from meow::ObjBase.
Definition at line 354 of file Bitmap.h.
The documentation for this class was generated from the following file:

- meowpp/gra/Bitmap.h

6.4 meow::BundleAdjustment< Pixel > Class Template Reference

`#include "BundleAdjustment.h"`
Inheritance diagram for meow::BundleAdjustment< Pixel >:


meow::ObjBase
meow::BundleAdjustment< Pixel >
meow::BundleAdjustment_LM< Pixel >

Public Member Functions

- virtual ~BundleAdjustment ()
- virtual bool adjustEye (std::vector< SceneInfo< Pixel > > *seq) const
- virtual bool adjustFixedPoint (std::vector< SceneInfo< Pixel > > *seq) const

Protected Member Functions

- BundleAdjustment ()

Additional Inherited Members

6.4.1 Detailed Description

template<class Pixel>class meow::BundleAdjustment< Pixel >

Definition at line 35 of file BundleAdjustment.h.

6.4.2 Constructor & Destructor Documentation

template<class Pixel > meow::BundleAdjustment< Pixel >::BundleAdjustment ( ) `[inline]`, `[protected]`

Definition at line 37 of file BundleAdjustment.h.

template<class Pixel > virtual meow::BundleAdjustment< Pixel >::~BundleAdjustment ( ) [inline], [virtual]

Definition at line 40 of file BundleAdjustment.h.

6.4.3 Member Function Documentation

template<class Pixel > virtual bool meow::BundleAdjustment< Pixel >::adjustEye (std::vector< SceneInfo< Pixel > > ∗ *seq*) const [inline], [virtual]

Reimplemented in meow::BundleAdjustment_LM< Pixel >.

Definition at line 43 of file BundleAdjustment.h.

template<class Pixel > virtual bool meow::BundleAdjustment< Pixel >::adjustFixedPoint (std::vector< SceneInfo< Pixel > > ∗ *seq*) const [inline], [virtual]

Reimplemented in meow::BundleAdjustment_LM< Pixel >.

Definition at line 47 of file BundleAdjustment.h.

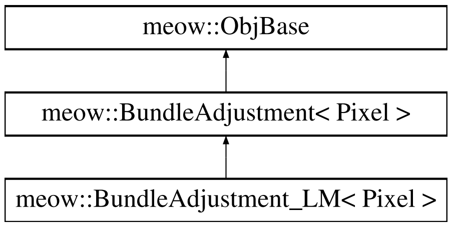
The documentation for this class was generated from the following file:

- meowpp/gra/BundleAdjustment.h

6.5 meow::BundleAdjustment_LM< Pixel > Class Template Reference

`#include "BundleAdjustment_LM.h"`

Inheritance diagram for meow::BundleAdjustment_LM< Pixel >:


meow::ObjBase
meow::BundleAdjustment< Pixel >
meow::BundleAdjustment_LM< Pixel >

Public Member Functions

- BundleAdjustment_LM ()
- BundleAdjustment_LM (BundleAdjustment_LM const &b)
- ~BundleAdjustment_LM ()
- BundleAdjustment_LM & copyFrom (BundleAdjustment_LM const &b)
- BundleAdjustment_LM & referenceFrom (BundleAdjustment_LM const &b)
- double threshold () const
- double threshold (double t)
- bool adjustEye (std::vector< SceneInfo< Pixel > > ∗seq) const
- bool adjustFixedPoint (std::vector< SceneInfo< Pixel > > ∗seq) const
- bool write (FILE ∗f, bool bin, unsigned int fg) const

  , implement `false`
- bool read (FILE ∗f, bool bin, unsigned int fg) const
- ObjBase ∗ create () const

  new, implement `NULL`
- ObjBase ∗ copyFrom (ObjBase const ∗o)

  , operator=
- char const ∗ ctype () const

  C-style stringclasstype name
- std::string type () const

  std::stringclasstype name

Additional Inherited Members

6.5.1 Detailed Description

template<class Pixel>class meow::BundleAdjustment_LM< Pixel >

Definition at line 20 of file BundleAdjustment_LM.h.

6.5.2 Constructor & Destructor Documentation

template<class Pixel > meow::BundleAdjustment_LM< Pixel >::BundleAdjustment_LM ( ) [inline]

Definition at line 177 of file BundleAdjustment_LM.h.

template<class Pixel > meow::BundleAdjustment_LM< Pixel >::BundleAdjustment_LM (BundleAdjustment_LM< Pixel > const & *b*) [inline]

Definition at line 180 of file BundleAdjustment_LM.h.

template<class Pixel > meow::BundleAdjustment_LM< Pixel >::~BundleAdjustment_LM ( ) [inline]

Definition at line 184 of file BundleAdjustment_LM.h.

6.5.3 Member Function Documentation

template<class Pixel > bool meow::BundleAdjustment_LM< Pixel >::adjustEye (std::vector< SceneInfo< Pixel > > ∗ *seq*) const [inline], [virtual]

Reimplemented from meow::BundleAdjustment< Pixel >.
Definition at line 206 of file BundleAdjustment_LM.h.

template<class Pixel > bool meow::BundleAdjustment_LM< Pixel >::adjustFixedPoint (std::vector< SceneInfo< Pixel > > ∗ *seq*) const [inline], [virtual]

Reimplemented from meow::BundleAdjustment< Pixel >.
Definition at line 344 of file BundleAdjustment_LM.h.

template<class Pixel > BundleAdjustment_LM& meow::BundleAdjustment_LM< Pixel >::copyFrom (BundleAdjustment_LM< Pixel > const & *b*) [inline]

Definition at line 187 of file BundleAdjustment_LM.h.

template<class Pixel > ObjBase∗ meow::BundleAdjustment_LM< Pixel >::copyFrom (ObjBase const ∗ *b*) [inline], [virtual]

, operator=

Parameters

`in`	*b*	

Returns

`this`

Reimplemented from meow::ObjBase.
Definition at line 360 of file BundleAdjustment_LM.h.

template<class Pixel > ObjBase∗ meow::BundleAdjustment_LM< Pixel >::create ( ) const [inline], [virtual]

new, implement `NULL`
Reimplemented from meow::ObjBase.
Definition at line 356 of file BundleAdjustment_LM.h.

template<class Pixel > char const∗ meow::BundleAdjustment_LM< Pixel >::ctype ( ) const `[inline]`, `[virtual]`

C-style stringclasstype name

Reimplemented from meow::ObjBase.

Definition at line 364 of file BundleAdjustment_LM.h.

template<class Pixel > bool meow::BundleAdjustment_LM< Pixel >::read (FILE ∗ *f*, bool *bin*, unsigned int *fg*) const `[inline]`

Definition at line 352 of file BundleAdjustment_LM.h.

template<class Pixel > BundleAdjustment_LM& meow::BundleAdjustment_LM< Pixel >::referenceFrom (BundleAdjustment_LM< Pixel > const & *b*) `[inline]`

Definition at line 192 of file BundleAdjustment_LM.h.

template<class Pixel > double meow::BundleAdjustment_LM< Pixel >::threshold ( ) const `[inline]`

Definition at line 197 of file BundleAdjustment_LM.h.

template<class Pixel > double meow::BundleAdjustment_LM< Pixel >::threshold (double *t*) `[inline]`

Definition at line 201 of file BundleAdjustment_LM.h.

template<class Pixel > std::string meow::BundleAdjustment_LM< Pixel >::type ( ) const `[inline]`, `[virtual]`

std::stringclasstype name

Reimplemented from meow::ObjBase.

Definition at line 368 of file BundleAdjustment_LM.h.

template<class Pixel > bool meow::BundleAdjustment_LM< Pixel >::write (FILE ∗ *f*, bool *bin*, unsigned int *fg*) const `[inline]`, `[virtual]`

, implement `false`

Parameters

`in`	*f*	
`in`	*bin*	binary
`in`	*fg*	argument

Returns

Reimplemented from meow::ObjBase.

Definition at line 348 of file BundleAdjustment_LM.h.

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

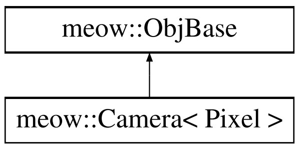
- meowpp/gra/BundleAdjustment_LM.h

6.6 meow::Camera< Pixel > Class Template Reference

Camera.

`#include "Camera.h"`

Inheritance diagram for meow::Camera< Pixel >:



Public Types

- typedef IdentityPoints< int,
  double > FixedPoints2D

Public Member Functions

- Camera ()

  constructor
- Camera (Camera const &b)

  copy constructor
- ∼Camera ()

  destructor
- Camera & copyFrom (Camera const &b)
- Camera & referenceFrom (Camera const &b)
- Photo< Pixel > const & photo () const

  photo
- Photo< Pixel > & photoGet ()

  photo (non-constant)
- Photo< Pixel > const & photo (Photo< Pixel > const &pho)

  photo
- Rotation3D< double > const & rotation () const

  rotation
- Rotation3D< double > & rotationGet ()

  rotation (non-constant)
- Rotation3D< double > const & rotation (Rotation3D< double > const &rot)

  rotation
- FixedPoints2D const & fixedPoints2D () const

  FixedPoint
- FixedPoints2D & fixedPoints2DGet () const

  FixedPoint(non-constant reference)
- FixedPoints2D const & fixedPoints2D (FixedPoints2D const &fps2d) const

  FixedPoint
- Vector< double > fixedPoint2D (int i)

  ifixed points 2d
- bool inside (Vector3D< double > p) const
- Pixel color (Vector3D< double > p) const

  color
- Camera & operator= (Camera const &b)

  same as `copyFrom(b)`
- bool write (FILE ∗f, bool bin, unsigned int fg) const
- bool read (FILE ∗f, bool bin, unsigned int fg)

- ObjBase ∗ create () const

  new
- ObjBase ∗ copyFrom (ObjBase const ∗b)
- char const ∗ ctype () const

  classtype
- std::string type () const

  classtype

Additional Inherited Members

6.6.1 Detailed Description

template<class Pixel>class meow::Camera< Pixel >

Camera.

`Photo Rotation3D`. fixedPoint,

Author

cat_leopard

Definition at line 23 of file Camera.h.

6.6.2 Member Typedef Documentation

template<class Pixel> typedef IdentityPoints<int, double> meow::Camera< Pixel >::FixedPoints2D

Definition at line 25 of file Camera.h.

6.6.3 Constructor & Destructor Documentation

template<class Pixel> meow::Camera< Pixel >::Camera ( ) `[inline]`

constructor

Definition at line 47 of file Camera.h.

template<class Pixel> meow::Camera< Pixel >::Camera (Camera< Pixel > const & *b*) `[inline]`

copy constructor

Definition at line 53 of file Camera.h.

template<class Pixel> meow::Camera< Pixel >::∼Camera ( ) `[inline]`

destructor

Definition at line 59 of file Camera.h.

6.6.4 Member Function Documentation

template<class Pixel> Pixel meow::Camera< Pixel >::color (Vector3D< double > *p*) const `[inline]`

color

Definition at line 164 of file Camera.h.

template<class Pixel> Camera& meow::Camera< Pixel >::copyFrom (Camera< Pixel > const & *b*) `[inline]`

Definition at line 65 of file Camera.h.

template<class Pixel> ObjBase* meow::Camera< Pixel >::copyFrom (ObjBase const * *b*) `[inline]`, `[virtual]`

`ObjBase const*` method`copyFrom`

Parameters

`in`	*b*	

Returns

this

Reimplemented from meow::ObjBase.
Definition at line 237 of file Camera.h.

template<class Pixel> ObjBase∗ meow::Camera< Pixel >::create ( ) const `[inline]`, `[virtual]`

new

Returns

newpointer

Reimplemented from meow::ObjBase.
Definition at line 225 of file Camera.h.

template<class Pixel> char const∗ meow::Camera< Pixel >::ctype ( ) const `[inline]`, `[virtual]`

classtype

Returns

`char const* typename`

Reimplemented from meow::ObjBase.
Definition at line 245 of file Camera.h.

template<class Pixel> Vector<double> meow::Camera< Pixel >::fixedPoint2D (int *i*) `[inline]`

ifixed points 2d
Definition at line 149 of file Camera.h.

template<class Pixel> FixedPoints2D const& meow::Camera< Pixel >::fixedPoints2D ( ) const `[inline]`

FixedPoint
Definition at line 125 of file Camera.h.

template<class Pixel> FixedPoints2D const& meow::Camera< Pixel >::fixedPoints2D (FixedPoints2D const & *fps2d*) const `[inline]`

FixedPoint
Definition at line 139 of file Camera.h.

template<class Pixel> FixedPoints2D& meow::Camera< Pixel >::fixedPoints2DGet ( ) const `[inline]`

FixedPoint(non-constant reference)
Definition at line 132 of file Camera.h.

template<class Pixel> bool meow::Camera< Pixel >::inside (Vector3D< double > *p*) const `[inline]`

Definition at line 156 of file Camera.h.

template<class Pixel> Camera& meow::Camera< Pixel >::operator= (Camera< Pixel > const & *b*) `[inline]`

same as `copyFrom(b)`

Definition at line 172 of file Camera.h.

template<class Pixel> Photo<Pixel> const& meow::Camera< Pixel >::photo ( ) const `[inline]`

photo

Definition at line 81 of file Camera.h.

template<class Pixel> Photo<Pixel> const& meow::Camera< Pixel >::photo (Photo< Pixel > const & *pho*) `[inline]`

photo

Definition at line 95 of file Camera.h.

template<class Pixel> Photo<Pixel>& meow::Camera< Pixel >::photoGet ( ) `[inline]`

photo (non-constant)

Definition at line 88 of file Camera.h.

template<class Pixel> bool meow::Camera< Pixel >::read (FILE ∗ *f,* bool *bin,* unsigned int *fg*) `[inline]`, `[virtual]`

Note

Reimplemented from meow::ObjBase.

Definition at line 201 of file Camera.h.

template<class Pixel> Camera& meow::Camera< Pixel >::referenceFrom (Camera< Pixel > const & *b*) `[inline]`

Definition at line 73 of file Camera.h.

template<class Pixel> Rotation3D<double> const& meow::Camera< Pixel >::rotation ( ) const `[inline]`

rotation

Definition at line 103 of file Camera.h.

template<class Pixel> Rotation3D<double> const& meow::Camera< Pixel >::rotation (Rotation3D< double > const & *rot*) `[inline]`

rotation

Definition at line 117 of file Camera.h.

template<class Pixel> Rotation3D<double>& meow::Camera< Pixel >::rotationGet ( ) `[inline]`

rotation (non-constant)

Definition at line 110 of file Camera.h.

template<class Pixel> std::string meow::Camera< Pixel >::type ( ) const `[inline]`, `[virtual]`

classtype

Returns

`std::string` typename

Reimplemented from meow::ObjBase.

Definition at line 254 of file Camera.h.

template<class Pixel> bool meow::Camera< Pixel >::write (FILE * *f*, bool *bin*, unsigned int *fg*) const `[inline]`, `[virtual]`

Note

Reimplemented from meow::ObjBase.

Definition at line 180 of file Camera.h.

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

- meowpp/gra/Camera.h

6.7 meow::Color3_Space< T > Class Template Reference

channel

```
#include "Color3_Space.h"
```

Public Member Functions

- virtual ∼Color3_Space ()
- Vector3D< T > const & minV () const
- Vector3D< T > const & maxV () const
- Vector3D< T > const & valV () const
- Vector3D< T > const & valV (Vector3D< T > const &vv)
- Vector3D< T > & valVGet ()
- T const & min (size_t id) const
- T const & max (size_t id) const
- T const & val (size_t id) const
- T const & val (size_t i, T const &c)
- T & valGet (size_t id)
- Matrix< T > matrix () const

Protected Member Functions

- Color3_Space (Vector3D< T > const &min_bound, Vector3D< T > const &max_bound, Vector3D< T > const &init_value)
- Color3_Space (Color3_Space const &b)
- Color3_Space< T > & copyFrom (Color3_Space< T > const &b)

Protected Attributes

- Vector3D< T > min_
- Vector3D< T > max_
- Vector3D< T > val_

6.7.1 Detailed Description

template<class T>class meow::Color3_Space< T >

channel

Author

cat_leopard

Definition at line 18 of file Color3_Space.h.

6.7.2 Constructor & Destructor Documentation

template<class T> meow::Color3_Space< T >::Color3_Space (Vector3D< T > const & *min_bound*, Vector3D< T > const & *max_bound*, Vector3D< T > const & *init_value*) `[inline]`, `[protected]`

Definition at line 23 of file Color3_Space.h.

template<class T> meow::Color3_Space< T >::Color3_Space (Color3_Space< T > const & *b*) `[inline]`, `[protected]`

Definition at line 30 of file Color3_Space.h.

template<class T> virtual meow::Color3_Space< T >::~Color3_Space ( ) `[inline]`, `[virtual]`

Definition at line 40 of file Color3_Space.h.

6.7.3 Member Function Documentation

template<class T> Color3_Space<T>& meow::Color3_Space< T >::copyFrom (Color3_Space< T > const & *b*) `[inline]`, `[protected]`

Definition at line 35 of file Color3_Space.h.

template<class T> Matrix<T> meow::Color3_Space< T >::matrix ( ) const `[inline]`

Definition at line 60 of file Color3_Space.h.

template<class T> T const& meow::Color3_Space< T >::max (size_t *id*) const `[inline]`

Definition at line 47 of file Color3_Space.h.

template<class T> Vector3D<T> const& meow::Color3_Space< T >::maxV ( ) const `[inline]`

Definition at line 42 of file Color3_Space.h.

template<class T> T const& meow::Color3_Space< T >::min (size_t *id*) const `[inline]`

Definition at line 46 of file Color3_Space.h.

template<class T> Vector3D<T> const& meow::Color3_Space< T >::minV ( ) const `[inline]`

Definition at line 41 of file Color3_Space.h.

template<class T> T const& meow::Color3_Space< T >::val (size_t *id*) const `[inline]`

Definition at line 48 of file Color3_Space.h.

template<class T> T const& meow::Color3_Space< T >::val (size_t *i*, T const & *c*) `[inline]`

Definition at line 49 of file Color3_Space.h.

template<class T> T& meow::Color3_Space< T >::valGet (size_t *id*) `[inline]`

Definition at line 55 of file Color3_Space.h.

template<class T> Vector3D<T> const& meow::Color3_Space< T >::valV ( ) const `[inline]`

Definition at line 43 of file Color3_Space.h.

template<class T> Vector3D<T> const& meow::Color3_Space< T >::valV (Vector3D< T > const & *vv*) `[inline]`

Definition at line 44 of file Color3_Space.h.

template<class T> Vector3D<T>& meow::Color3_Space< T >::valVGet ( ) `[inline]`

Definition at line 45 of file Color3_Space.h.

6.7.4 Member Data Documentation

template<class T> Vector3D<T> meow::Color3_Space< T >::max_ `[protected]`

Definition at line 21 of file Color3_Space.h.

template<class T> Vector3D<T> meow::Color3_Space< T >::min_ `[protected]`

Definition at line 20 of file Color3_Space.h.

template<class T> Vector3D<T> meow::Color3_Space< T >::val_ `[protected]`

Definition at line 22 of file Color3_Space.h.

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

- meowpp/colors/Color3_Space.h

6.8 meow::DisjointSet Class Reference

`#include "DisjointSet.h"`

Public Member Functions

- DisjointSet ()

  constructor
- DisjointSet (size_t n)

  constructor
- DisjointSet (DisjointSet const &dsj)

  constructor
- size_t root (size_t a) const

  number
- size_t size () const

  element
- void reset (size_t n)

- size_t merge (size_t a, size_t b)

6.8.1 Detailed Description

DisjointSet **Data Dtructure**, .

Note

- ,
- *number number* , set

Author

cat_leopard

Definition at line 25 of file DisjointSet.h.

6.8.2 Constructor & Destructor Documentation

meow::DisjointSet::DisjointSet ( ) `[inline]`

constructor

Definition at line 54 of file DisjointSet.h.

meow::DisjointSet::DisjointSet (size_t *n*) `[inline]`

constructor

Parameters

in	*n*	elements

Definition at line 62 of file DisjointSet.h.

meow::DisjointSet::DisjointSet (DisjointSet const & *dsj*) `[inline]`

constructor

`DisjointSet`

Parameters

in	*dsj*	`DisjointSet`

Definition at line 73 of file DisjointSet.h.

6.8.3 Member Function Documentation

size_t meow::DisjointSet::merge (size_t *a*, size_t *b*) `[inline]`

number1 **number2** , .

Parameters

in	*a*	*number1*
in	*b*	*number2*

Returns

Definition at line 128 of file DisjointSet.h.

void meow::DisjointSet::reset (size_t *n*) `[inline]`

, *n*

Parameters

in	*n*	*n*

Returns

Definition at line 107 of file DisjointSet.h.

size_t meow::DisjointSet::root (size_t *a*) const `[inline]`

number

Parameters

`in`	*a*	number

Returns

Definition at line 85 of file DisjointSet.h.

size_t meow::DisjointSet::size ( ) const `[inline]`

element

Returns

element

Definition at line 95 of file DisjointSet.h.
The documentation for this class was generated from the following file:

- meowpp/dsa/DisjointSet.h

6.9 meow::SplayTree_Range< Key, Value >::Element Class Reference

```
stl iterator ,Element
  #include "SplayTree.h"
```

Public Member Functions

- Element ()
- Element (Node ∗node)
- Element (Element const &element2)
- ∼Element ()
- Element & copyFrom (Element const &e)
- bool same (Element const &e2) const

  Entry
- Element & operator= (Element const &e2)

  same as copyFrom
- Entry ∗ operator-> ()

  `std::pair<Key const&,Value&>∗`
- Entry & operator∗ ()

  `std::pair<Key const&,Value&>&`
- bool operator== (Element const &e2) const

  same as `same(e2)`
- bool operator!= (Element const &e2) const

  same as `!same`(e2)

6.9.1 Detailed Description

template<class Key, class Value>class meow::SplayTree_Range< Key, Value >::Element

`stl iterator ,Element`

Definition at line 750 of file SplayTree.h.

6.9.2 Constructor & Destructor Documentation

template<class Key , class Value > meow::SplayTree_Range< Key, Value >::Element::Element ( ) `[inline]`

Definition at line 762 of file SplayTree.h.

template<class Key , class Value > meow::SplayTree_Range< Key, Value >::Element::Element (Node ∗ *node*) `[inline]`

Definition at line 764 of file SplayTree.h.

template<class Key , class Value > meow::SplayTree_Range< Key, Value >::Element::Element (Element const & *element2*) `[inline]`

Definition at line 767 of file SplayTree.h.

template<class Key , class Value > meow::SplayTree_Range< Key, Value >::Element::∼Element ( ) `[inline]`

Definition at line 770 of file SplayTree.h.

6.9.3 Member Function Documentation

template<class Key , class Value > Element& meow::SplayTree_Range< Key, Value >::Element::copyFrom (Element const & *e*) `[inline]`

Definition at line 775 of file SplayTree.h.

template<class Key , class Value > bool meow::SplayTree_Range< Key, Value >::Element::operator!= (Element const & *e2*) const `[inline]`

same as `!same`(e2)

Definition at line 806 of file SplayTree.h.

template<class Key , class Value > Entry& meow::SplayTree_Range< Key, Value >::Element::operator∗ ( ) `[inline]`

`std::pair<Key const&,Value&>&`

Definition at line 796 of file SplayTree.h.

template<class Key , class Value > Entry∗ meow::SplayTree_Range< Key, Value >::Element::operator-> ( ) `[inline]`

`std::pair<Key const&,Value&>*`

Definition at line 791 of file SplayTree.h.

template<class Key , class Value > Element& meow::SplayTree_Range< Key, Value >::Element::operator= (Element const & *e2*) `[inline]`

same as copyFrom

Definition at line 786 of file SplayTree.h.

template<class Key , class Value > bool meow::SplayTree_Range< Key, Value >::Element::operator== (Element const & *e2*) const `[inline]`

same as `same(e2)`

Definition at line 801 of file SplayTree.h.

template<class Key , class Value > bool meow::SplayTree_Range< Key, Value >::Element::same (Element const & *e2*) const `[inline]`

Entry

Definition at line 781 of file SplayTree.h.

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

- meowpp/dsa/SplayTree.h

6.10 meow::SplayTree< Key, Value >::Element Class Reference

`stl iterator ,Element`

`#include "SplayTree.h"`

Public Member Functions

- Element ()
- Element (Node ∗node)
- Element (Element const &element2)
- ∼Element ()
- Element & copyFrom (Element const &e)

- bool same (Element const &e2) const

  Entry
- Element & operator= (Element const &e2)

  same as copyFrom
- Entry ∗ operator-> ()

  `std::pair<Key const&,Value&>*`
- Entry & operator∗ ()

  `std::pair<Key const&,Value&>&`
- bool operator== (Element const &e2) const

  same as `same(e2)`
- bool operator!= (Element const &e2) const

  same as `!same`(e2)

6.10.1 Detailed Description

template<class Key, class Value>class meow::SplayTree< Key, Value >::Element

`stl iterator ,Element`

Definition at line 191 of file SplayTree.h.

6.10.2 Constructor & Destructor Documentation

template<class Key , class Value > meow::SplayTree< Key, Value >::Element::Element ( ) `[inline]`

Definition at line 203 of file SplayTree.h.

template<class Key , class Value > meow::SplayTree< Key, Value >::Element::Element (Node ∗ *node*) `[inline]`

Definition at line 205 of file SplayTree.h.

template<class Key , class Value > meow::SplayTree< Key, Value >::Element::Element (Element const & *element2*) `[inline]`

Definition at line 208 of file SplayTree.h.

template<class Key , class Value > meow::SplayTree< Key, Value >::Element::∼Element ( ) `[inline]`

Definition at line 211 of file SplayTree.h.

6.10.3 Member Function Documentation

template<class Key , class Value > Element& meow::SplayTree< Key, Value >::Element::copyFrom (Element const & *e*) `[inline]`

Definition at line 216 of file SplayTree.h.

template<class Key , class Value > bool meow::SplayTree< Key, Value >::Element::operator!= (Element const & *e2*) const `[inline]`

same as !`same`(e2)

Definition at line 247 of file SplayTree.h.

template<class Key , class Value > Entry& meow::SplayTree< Key, Value >::Element::operator∗ ( ) `[inline]`

`std::pair<Key const&,Value&>&`

Definition at line 237 of file SplayTree.h.

template<class Key , class Value > Entry∗ meow::SplayTree< Key, Value >::Element::operator-> ( ) `[inline]`

`std::pair<Key const&,Value&>*`

Definition at line 232 of file SplayTree.h.

template<class Key , class Value > Element& meow::SplayTree< Key, Value >::Element::operator= (Element const & *e2*) `[inline]`

same as copyFrom

Definition at line 227 of file SplayTree.h.

template<class Key , class Value > bool meow::SplayTree< Key, Value >::Element::operator== (Element const & *e2*) const `[inline]`

same as `same(e2)`

Definition at line 242 of file SplayTree.h.

template<class Key , class Value > bool meow::SplayTree< Key, Value >::Element::same (Element const & *e2*) const `[inline]`

Entry

Definition at line 222 of file SplayTree.h.

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

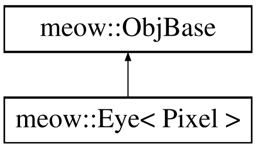
- meowpp/dsa/SplayTree.h

6.11 meow::Eye< Pixel > Class Template Reference

`Camera` offset transformation

`#include "Eye.h"`

Inheritance diagram for meow::Eye< Pixel >:



Public Member Functions

- Eye ()
- Eye (Eye const &b)
- Eye (Camera< Pixel > const &c, Vector3D< double > const &o)
- ∼Eye ()
- Eye & copyFrom (Eye const &e)
- Eye & referenceFrom (Eye const &e)
- Camera< Pixel > const & camera () const
- Camera< Pixel > & cameraGet ()
- Camera< Pixel > const & camera (Camera< Pixel > const &c)
- Vector3D< double > const & offset () const
- Vector3D< double > & offsetGet ()
- Vector3D< double > const & offset (Vector3D< double > const &ofs)
- bool inside (Vector3D< double > const &v) const
- Eye & operator= (Eye const &e)
- bool write (FILE ∗f, bool bin, unsigned int fg) const
- bool read (FILE ∗f, bool bin, unsigned int fg)
- ObjBase ∗ create () const

  new
- ObjBase ∗ copyFrom (ObjBase const ∗b)
- char const ∗ ctype () const

  classtype
- std::string type () const

  classtype

Additional Inherited Members

6.11.1 Detailed Description

template<class Pixel>class meow::Eye< Pixel >

`Camera` offset transformation

Author

cat_leopard

Definition at line 17 of file Eye.h.

6.11.2 Constructor & Destructor Documentation

template<class Pixel> meow::Eye< Pixel >::Eye ( ) `[inline]`

Definition at line 38 of file Eye.h.

template<class Pixel> meow::Eye< Pixel >::Eye (Eye< Pixel > const & *b*) `[inline]`

Definition at line 41 of file Eye.h.

template<class Pixel> meow::Eye< Pixel >::Eye (Camera< Pixel > const & *c*, Vector3D< double > const & *o*) `[inline]`

Definition at line 44 of file Eye.h.

template<class Pixel> meow::Eye< Pixel >::~Eye ( ) `[inline]`

Definition at line 47 of file Eye.h.

6.11.3 Member Function Documentation

template<class Pixel> Camera<Pixel> const& meow::Eye< Pixel >::camera ( ) const `[inline]`

Definition at line 60 of file Eye.h.

template<class Pixel> Camera<Pixel> const& meow::Eye< Pixel >::camera (Camera< Pixel > const & *c*) `[inline]`

Definition at line 68 of file Eye.h.

template<class Pixel> Camera<Pixel>& meow::Eye< Pixel >::cameraGet ( ) `[inline]`

Definition at line 64 of file Eye.h.

template<class Pixel> Eye& meow::Eye< Pixel >::copyFrom (Eye< Pixel > const & *e*) `[inline]`

Definition at line 50 of file Eye.h.

template<class Pixel> ObjBase* meow::Eye< Pixel >::copyFrom (ObjBase const * *b*) `[inline]`, `[virtual]`

`ObjBase` `const*` method`copyFrom`

Parameters

`in`	*b*	

Returns

this

Reimplemented from meow::ObjBase.
Definition at line 151 of file Eye.h.

template<class Pixel> ObjBase* meow::Eye< Pixel >::create ( ) const `[inline]`, `[virtual]`

new

Returns

newpointer

Reimplemented from meow::ObjBase.
Definition at line 139 of file Eye.h.

template<class Pixel> char const* meow::Eye< Pixel >::ctype ( ) const `[inline]`, `[virtual]`

classtype

Returns

`char const* typename`

Reimplemented from meow::ObjBase.
Definition at line 159 of file Eye.h.

template<class Pixel> bool meow::Eye< Pixel >::inside (Vector3D< double > const & *v*) const `[inline]`

Definition at line 86 of file Eye.h.

template<class Pixel> Vector3D<double> const& meow::Eye< Pixel >::offset ( ) const `[inline]`

Definition at line 73 of file Eye.h.

template<class Pixel> Vector3D<double> const& meow::Eye< Pixel >::offset (Vector3D< double > const & *ofs*) `[inline]`

Definition at line 81 of file Eye.h.

template<class Pixel> Vector3D<double>& meow::Eye< Pixel >::offsetGet ( ) `[inline]`

Definition at line 77 of file Eye.h.

template<class Pixel> Eye& meow::Eye< Pixel >::operator= (Eye< Pixel > const & *e*) `[inline]`

Definition at line 90 of file Eye.h.

template<class Pixel> bool meow::Eye< Pixel >::read (FILE ∗ *f*, bool *bin*, unsigned int *fg*) `[inline]`, `[virtual]`

Note

Reimplemented from meow::ObjBase.
Definition at line 119 of file Eye.h.

template<class Pixel> Eye& meow::Eye< Pixel >::referenceFrom (Eye< Pixel > const & *e*) `[inline]`

Definition at line 55 of file Eye.h.

template<class Pixel> std::string meow::Eye< Pixel >::type ( ) const `[inline]`, `[virtual]`

classtype

Returns

`std::string` typename

Reimplemented from meow::ObjBase.
Definition at line 167 of file Eye.h.

template<class Pixel> bool meow::Eye< Pixel >::write (FILE ∗ *f*, bool *bin*, unsigned int *fg*) const `[inline]`, `[virtual]`

Note

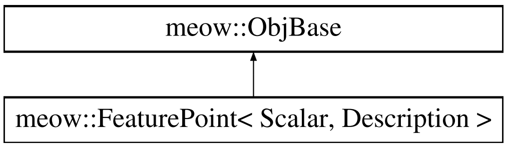
Reimplemented from meow::ObjBase.
Definition at line 98 of file Eye.h.
The documentation for this class was generated from the following file:

- meowpp/gra/Eye.h

6.12 meow::FeaturePoint< Scalar, Description > Class Template Reference

`#include "FeaturePoint.h"`

Inheritance diagram for meow::FeaturePoint< Scalar, Description >:



Public Member Functions

- FeaturePoint ()

  constructor
- FeaturePoint (size_t pDim, size_t dDim)

  constructor
- FeaturePoint (FeaturePoint const &fp)

  constructor
- ∼FeaturePoint ()

  destructor
- FeaturePoint & copyFrom (FeaturePoint const &fp)
- FeaturePoint & referenceFrom (FeaturePoint const &fp)
- Vector< Scalar > const & position () const

  position
- Vector< Scalar > & positionGet ()

  position (non-const reference)
- Vector< Description > const & description () const

  description
- Vector< Description > & descriptionGet ()

  description (non-const reference)
- Vector< Scalar > const & position (Vector< Scalar > const &p) const

  position
- Vector< Description > const & description (Vector< Description > const &d)

  description
- Scalar position (size_t index) const

  positioniscalar
- Description description (size_t i) const

  descriptioniDescription
- Scalar position (size_t i, Scalar const &s)

  positioniscalar
- Description description (size_t i, Description const &d)

  descriptioniDescription
- FeaturePoint & operator= (FeaturePoint const &fp)

  same as copyFrom(fp)
- Scalar const & operator() (size_t i) const

  same as position(i)
- Description operator[] (size_t i) const

same as description(i)

- bool write (FILE ∗f, bool bin, unsigned int fg) const

  , implement `false`
- bool read (FILE ∗f, bool bin, unsigned int fg)

  , implement `false`
- ObjBase ∗ create () const

  new, implement `NULL`
- ObjBase ∗ copyFrom (ObjBase const &b)
- char const ∗ ctype () const

  C-style stringclasstype name
- std::string type () const

  std::stringclasstype name

Additional Inherited Members

6.12.1 Detailed Description

template<class Scalar, class Description>class meow::FeaturePoint< Scalar, Description >

Author

cat_leopard

Definition at line 21 of file FeaturePoint.h.

6.12.2 Constructor & Destructor Documentation

template<class Scalar , class Description > meow::FeaturePoint< Scalar, Description >::FeaturePoint ( ) `[inline]`

constructor

Definition at line 29 of file FeaturePoint.h.

template<class Scalar , class Description > meow::FeaturePoint< Scalar, Description >::FeaturePoint (size_t *pDim,* size_t *dDim*) `[inline]`

constructor

Definition at line 35 of file FeaturePoint.h.

template<class Scalar , class Description > meow::FeaturePoint< Scalar, Description >::FeaturePoint (FeaturePoint< Scalar, Description > const & *fp*) `[inline]`

constructor

Definition at line 42 of file FeaturePoint.h.

template<class Scalar , class Description > meow::FeaturePoint< Scalar, Description >::∼FeaturePoint ( ) `[inline]`

destructor

Definition at line 49 of file FeaturePoint.h.

6.12.3 Member Function Documentation

template<class Scalar , class Description > FeaturePoint& meow::FeaturePoint< Scalar, Description >::copyFrom (FeaturePoint< Scalar, Description > const & *fp*) `[inline]`

Definition at line 55 of file FeaturePoint.h.

template<class Scalar , class Description > ObjBase* meow::FeaturePoint< Scalar, Description >::copyFrom (ObjBase const & *b*) `[inline]`

Definition at line 219 of file FeaturePoint.h.

template<class Scalar , class Description > ObjBase* meow::FeaturePoint< Scalar, Description >::create ( ) const `[inline]`, `[virtual]`

new, implement `NULL`

Reimplemented from meow::ObjBase.

Definition at line 215 of file FeaturePoint.h.

template<class Scalar , class Description > char const* meow::FeaturePoint< Scalar, Description >::ctype ( ) const `[inline]`, `[virtual]`

C-style stringclasstype name

Reimplemented from meow::ObjBase.

Definition at line 223 of file FeaturePoint.h.

template<class Scalar , class Description > Vector<Description> const& meow::FeaturePoint< Scalar, Description >::description ( ) const `[inline]`

description

Definition at line 87 of file FeaturePoint.h.

template<class Scalar , class Description > Vector<Description> const& meow::FeaturePoint< Scalar, Description >::description (Vector< Description > const & *d*) `[inline]`

description

Definition at line 109 of file FeaturePoint.h.

template<class Scalar , class Description > Description meow::FeaturePoint< Scalar, Description >::description (size_t *i*) const `[inline]`

descriptioniDescription

Definition at line 124 of file FeaturePoint.h.

template<class Scalar , class Description > Description meow::FeaturePoint< Scalar, Description >::description (size_t *i*, Description const & *d*) `[inline]`

descriptioniDescription

Definition at line 139 of file FeaturePoint.h.

template<class Scalar , class Description > Vector<Description>& meow::FeaturePoint< Scalar, Description >::descriptionGet ( ) `[inline]`

description (non-const reference)

Definition at line 94 of file FeaturePoint.h.

template<class Scalar , class Description > Scalar const& meow::FeaturePoint< Scalar, Description >::operator() (size_t *i*) const `[inline]`

same as position(i)

Definition at line 154 of file FeaturePoint.h.

template<class Scalar , class Description > FeaturePoint& meow::FeaturePoint< Scalar, Description >::operator= (FeaturePoint< Scalar, Description > const & *fp*) `[inline]`

same as copyFrom(fp)

Definition at line 147 of file FeaturePoint.h.

template<class Scalar , class Description > Description meow::FeaturePoint< Scalar, Description >::operator[] (size_t *i*) const `[inline]`

same as description(i)

Definition at line 161 of file FeaturePoint.h.

template<class Scalar , class Description > Vector<Scalar> const& meow::FeaturePoint< Scalar, Description >::position ( ) const `[inline]`

position

Definition at line 73 of file FeaturePoint.h.

template<class Scalar , class Description > Vector<Scalar> const& meow::FeaturePoint< Scalar, Description >::position (Vector< Scalar > const & *p*) const `[inline]`

position

Definition at line 101 of file FeaturePoint.h.

template<class Scalar , class Description > Scalar meow::FeaturePoint< Scalar, Description >::position (size_t *index*) const `[inline]`

positioniscalar

Definition at line 117 of file FeaturePoint.h.

template<class Scalar , class Description > Scalar meow::FeaturePoint< Scalar, Description >::position (size_t *i*, Scalar const & *s*) `[inline]`

positioniscalar

Definition at line 131 of file FeaturePoint.h.

template<class Scalar , class Description > Vector<Scalar>& meow::FeaturePoint< Scalar, Description >::positionGet ( ) `[inline]`

position (non-const reference)

Definition at line 80 of file FeaturePoint.h.

template<class Scalar , class Description > bool meow::FeaturePoint< Scalar, Description >::read (FILE ∗ *f*, bool *bin*, unsigned int *fg*) `[inline]`, `[virtual]`

, implement `false`

Parameters

`in`	*f*	
`in`	*bin*	binary
`in`	*fg*	argument

Returns

Reimplemented from meow::ObjBase.

Definition at line 189 of file FeaturePoint.h.

template<class Scalar , class Description > FeaturePoint& meow::FeaturePoint< Scalar, Description >::referenceFrom (FeaturePoint< Scalar, Description > const & *fp*) `[inline]`

Definition at line 64 of file FeaturePoint.h.

template<class Scalar , class Description > std::string meow::FeaturePoint< Scalar, Description >::type ( ) const `[inline]`, `[virtual]`

std::stringclasstype name

Reimplemented from meow::ObjBase.

Definition at line 227 of file FeaturePoint.h.

template<class Scalar , class Description > bool meow::FeaturePoint< Scalar, Description >::write (FILE ∗ *f*, bool *bin*, unsigned int *fg*) const `[inline]`, `[virtual]`

, implement `false`

Parameters

`in`	*f*	
`in`	*bin*	binary
`in`	*fg*	argument

Returns

Reimplemented from meow::ObjBase.

Definition at line 165 of file FeaturePoint.h.

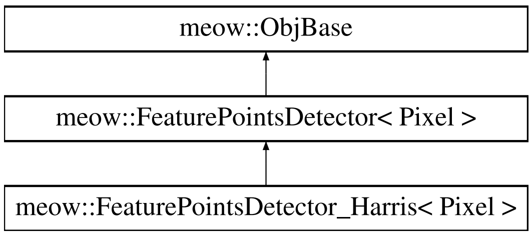
The documentation for this class was generated from the following file:

- meowpp/gra/FeaturePoint.h

6.13 meow::FeaturePointsDetector< Pixel > Class Template Reference

`#include "FeaturePointsDetector.h"`

Inheritance diagram for meow::FeaturePointsDetector< Pixel >:


meow::ObjBase
meow::FeaturePointsDetector< Pixel >
meow::FeaturePointsDetector_Harris< Pixel >

Public Member Functions

- virtual ∼FeaturePointsDetector ()
- virtual std::vector
  < FeaturePoint< double, double > > detect (Bitmap< Pixel > const &bitmap) const =0

Protected Member Functions

- FeaturePointsDetector ()

Additional Inherited Members

6.13.1 Detailed Description

template<class Pixel>class meow::FeaturePointsDetector< Pixel >

Definition at line 14 of file FeaturePointsDetector.h.

6.13.2 Constructor & Destructor Documentation

template<class Pixel > meow::FeaturePointsDetector< Pixel >::FeaturePointsDetector ( ) `[inline]`, `[protected]`

Definition at line 16 of file FeaturePointsDetector.h.

template<class Pixel > virtual meow::FeaturePointsDetector< Pixel >::~FeaturePointsDetector ( ) `[inline]`, `[virtual]`

Definition at line 18 of file FeaturePointsDetector.h.

6.13.3 Member Function Documentation

template<class Pixel > virtual std::vector<FeaturePoint<double, double> > meow::-FeaturePointsDetector< Pixel >::detect (Bitmap< Pixel > const & *bitmap*) const `[pure virtual]`

Implemented in meow::FeaturePointsDetector_Harris< Pixel >.

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

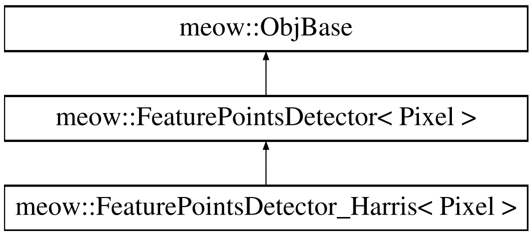
- meowpp/gra/FeaturePointsDetector.h

6.14 meow::FeaturePointsDetector_Harris< Pixel > Class Template Reference

Harris corner detect.

`#include "FeaturePointsDetector_Harris.h"`

Inheritance diagram for meow::FeaturePointsDetector_Harris< Pixel >:


meow::ObjBase
meow::FeaturePointsDetector< Pixel >
meow::FeaturePointsDetector_Harris< Pixel >

Public Types

- typedef FeaturePoint< double, double > MyFeaturePoint
- typedef std::vector < MyFeaturePoint > MyFeaturePoints

Public Member Functions

- FPD_Harris ()

  constructor
- FPD_Harris (FPD_Harris const &fps)

  constructor FeaturePointsDetector_Harris
- ~FPD_Harris ()
- FPD_Harris & copyFrom (FPD_Harris const &fps)
- FPD_Harris & referenceFrom (FPD_Harris const &fps)

- double paramK () const

  K.
- double paramR () const

  R.
- double paramW () const

  W.
- double paramN () const

  N.
- double paramG () const

  G.
- double paramL () const

  L.
- size_t paramB () const

  bound
- double paramK (double k)

  K.
- double paramR (double r)

  R.
- double paramW (double w)

  W.
- double paramN (double n)

  N.
- double paramL (double l)

  L.
- double paramG (double g)

  G.
- size_t paramB (size_t b)

  B.
- MyFeaturePoints detect (Bitmap< Pixel > const &bmp) const

- FPD_Harris & operator= (FPD_Harris const &fps)

  same as `copyFrom(fps)`
- MyFeaturePoints operator() (Bitmap< Pixel > const &bmp) const

  same as `detect(bmp)`
- bool write (FILE ∗f, bool bin, unsigned int fg) const

- bool read (FILE ∗f, bool bin, unsigned int fg)

- ObjBase ∗ create () const

  new
- ObjBase ∗ copyFrom (ObjBase const ∗b)

- char const ∗ ctype () const

  classtype
- std::string type () const

  classtype

Additional Inherited Members

6.14.1 Detailed Description

template<class Pixel>class meow::FeaturePointsDetector_Harris< Pixel >

Harris corner detect.

Author

cat_leopard

Definition at line 24 of file FeaturePointsDetector_Harris.h.

6.14.2 Member Typedef Documentation

template<class Pixel > typedef FeaturePoint<double, double> meow::FeaturePointsDetector_Harris< Pixel >::MyFeaturePoint

Definition at line 60 of file FeaturePointsDetector_Harris.h.

template<class Pixel > typedef std::vector<MyFeaturePoint> meow::FeaturePointsDetector_Harris< Pixel >::MyFeaturePoints

Definition at line 61 of file FeaturePointsDetector_Harris.h.

6.14.3 Constructor & Destructor Documentation

template<class Pixel > meow::FeaturePointsDetector_Harris< Pixel >::~FPD_Harris ( ) `[inline]`

Definition at line 71 of file FeaturePointsDetector_Harris.h.

6.14.4 Member Function Documentation

template<class Pixel > FPD_Harris& meow::FeaturePointsDetector_Harris< Pixel >::copyFrom (FPD_Harris const & *fps*) `[inline]`

Definition at line 75 of file FeaturePointsDetector_Harris.h.

template<class Pixel > ObjBase∗ meow::FeaturePointsDetector_Harris< Pixel >::copyFrom (ObjBase const ∗ *b*) `[inline]`, `[virtual]`

`ObjBase const* FeaturePointsDetector_Harris`. method`copyFrom`

Parameters

`in`	*b*	

Returns

this

Reimplemented from meow::ObjBase.
Definition at line 329 of file FeaturePointsDetector_Harris.h.

template<class Pixel > ObjBase∗ meow::FeaturePointsDetector_Harris< Pixel >::create ( ) const `[inline]`, `[virtual]`

new

Returns

newFeaturePointsDetector_Harris<Pixel>

Reimplemented from meow::ObjBase.
Definition at line 316 of file FeaturePointsDetector_Harris.h.

template<class Pixel > char const* meow::FeaturePointsDetector_Harris< Pixel >::ctype ( ) const `[inline]`, `[virtual]`

classtype

Returns

`char const* typename`

Reimplemented from meow::ObjBase.
Definition at line 337 of file FeaturePointsDetector_Harris.h.

template<class Pixel > MyFeaturePoints meow::FeaturePointsDetector_Harris< Pixel >::detect (Bitmap< Pixel > const & *bmp*) const `[inline]`, `[virtual]`

Parameters

in	*bmp*	

Returns

`std::vector<FeaturePoint<double,double>>`

Implements meow::FeaturePointsDetector< Pixel >.
Definition at line 168 of file FeaturePointsDetector_Harris.h.

template<class Pixel > meow::FeaturePointsDetector_Harris< Pixel >::FPD_Harris ( ) `[inline]`

constructor
Definition at line 63 of file FeaturePointsDetector_Harris.h.

template<class Pixel > meow::FeaturePointsDetector_Harris< Pixel >::FPD_Harris (FPD_Harris const & *fps*) `[inline]`

constructor FeaturePointsDetector_Harris
Definition at line 67 of file FeaturePointsDetector_Harris.h.

template<class Pixel > MyFeaturePoints meow::FeaturePointsDetector_Harris< Pixel >::operator() (Bitmap< Pixel > const & *bmp*) const `[inline]`

same as `detect(bmp)`
Definition at line 290 of file FeaturePointsDetector_Harris.h.

template<class Pixel > FPD_Harris& meow::FeaturePointsDetector_Harris< Pixel >::operator= (FPD_Harris const & *fps*) `[inline]`

same as `copyFrom(fps)`
Definition at line 285 of file FeaturePointsDetector_Harris.h.

template<class Pixel > size_t meow::FeaturePointsDetector_Harris< Pixel >::paramB ( ) const `[inline]`

bound
Definition at line 117 of file FeaturePointsDetector_Harris.h.

template<class Pixel > size_t meow::FeaturePointsDetector_Harris< Pixel >::paramB (size_t *b*) `[inline]`

B.
Definition at line 158 of file FeaturePointsDetector_Harris.h.

template< class Pixel > double meow::FeaturePointsDetector_Harris< Pixel >::paramG ( ) const `[inline]`

G.

Definition at line 107 of file FeaturePointsDetector_Harris.h.

template< class Pixel > double meow::FeaturePointsDetector_Harris< Pixel >::paramG (double *g*) `[inline]`

G.

Definition at line 152 of file FeaturePointsDetector_Harris.h.

template< class Pixel > double meow::FeaturePointsDetector_Harris< Pixel >::paramK ( ) const `[inline]`

K.

Definition at line 87 of file FeaturePointsDetector_Harris.h.

template< class Pixel > double meow::FeaturePointsDetector_Harris< Pixel >::paramK (double *k*) `[inline]`

K.

Definition at line 122 of file FeaturePointsDetector_Harris.h.

template< class Pixel > double meow::FeaturePointsDetector_Harris< Pixel >::paramL ( ) const `[inline]`

L.

Definition at line 112 of file FeaturePointsDetector_Harris.h.

template< class Pixel > double meow::FeaturePointsDetector_Harris< Pixel >::paramL (double *l*) `[inline]`

L.

Definition at line 146 of file FeaturePointsDetector_Harris.h.

template< class Pixel > double meow::FeaturePointsDetector_Harris< Pixel >::paramN ( ) const `[inline]`

N.

Definition at line 102 of file FeaturePointsDetector_Harris.h.

template< class Pixel > double meow::FeaturePointsDetector_Harris< Pixel >::paramN (double *n*) `[inline]`

N.

Definition at line 140 of file FeaturePointsDetector_Harris.h.

template< class Pixel > double meow::FeaturePointsDetector_Harris< Pixel >::paramR ( ) const `[inline]`

R.

Definition at line 92 of file FeaturePointsDetector_Harris.h.

template< class Pixel > double meow::FeaturePointsDetector_Harris< Pixel >::paramR (double *r*) `[inline]`

R.

Definition at line 128 of file FeaturePointsDetector_Harris.h.

template<class Pixel > double meow::FeaturePointsDetector_Harris< Pixel >::paramW ( ) const `[inline]`

W.

Definition at line 97 of file FeaturePointsDetector_Harris.h.

template<class Pixel > double meow::FeaturePointsDetector_Harris< Pixel >::paramW (double *w*) `[inline]`

W.

Definition at line 134 of file FeaturePointsDetector_Harris.h.

template<class Pixel > bool meow::FeaturePointsDetector_Harris< Pixel >::read (FILE ∗ *f*, bool *bin*, unsigned int *fg*) `[inline]`, `[virtual]`

Reimplemented from meow::ObjBase.

Definition at line 307 of file FeaturePointsDetector_Harris.h.

template<class Pixel > FPD_Harris& meow::FeaturePointsDetector_Harris< Pixel >::referenceFrom (FPD_Harris const & *fps*) `[inline]`

Definition at line 81 of file FeaturePointsDetector_Harris.h.

template<class Pixel > std::string meow::FeaturePointsDetector_Harris< Pixel >::type ( ) const `[inline]`, `[virtual]`

classtype

Returns

`std::string` typename

Reimplemented from meow::ObjBase.

Definition at line 345 of file FeaturePointsDetector_Harris.h.

template<class Pixel > bool meow::FeaturePointsDetector_Harris< Pixel >::write (FILE ∗ *f*, bool *bin*, unsigned int *fg*) const `[inline]`, `[virtual]`

Reimplemented from meow::ObjBase.

Definition at line 298 of file FeaturePointsDetector_Harris.h.

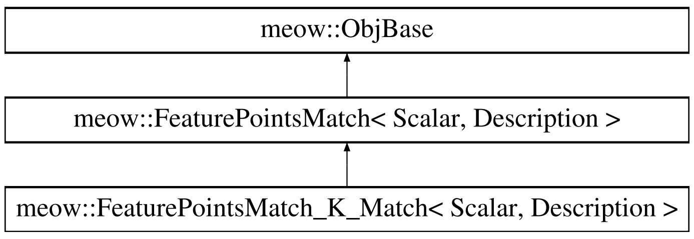
The documentation for this class was generated from the following file:

- meowpp/gra/FeaturePointsDetector_Harris.h

6.15 meow::FeaturePointsMatch< Scalar, Description > Class Template Reference

`#include "FeaturePointsMatch.h"`

Inheritance diagram for meow::FeaturePointsMatch< Scalar, Description >:


meow::ObjBase
meow::FeaturePointsMatch< Scalar, Description >
meow::FeaturePointsMatch_K_Match< Scalar, Description >

Public Types

- typedef std::vector
  < FeaturePoint< Scalar,
  Description > > FeaturePoints
- typedef std::vector
  < FeaturePoints > FeaturePointss

Public Member Functions

- virtual ∼FeaturePointsMatch ()
- virtual FeaturePointIndexPairs match (size_t dimension, FeaturePoints const &from, FeaturePoints const &to) const =0
- virtual FeaturePointIndexPairs match (size_t dimension, FeaturePoints const &from, FeaturePointss const &to) const =0
- virtual FeaturePointIndexPairs match (size_t dimension, FeaturePointss const &from, FeaturePointss const &to) const =0
- virtual FeaturePointIndexPairs match (size_t dimension, FeaturePointss const &fpss) const =0

Protected Member Functions

- FeaturePointsMatch ()

Additional Inherited Members

6.15.1 Detailed Description

template<class Scalar, class Description>class meow::FeaturePointsMatch< Scalar, Description >

Definition at line 17 of file FeaturePointsMatch.h.

6.15.2 Member Typedef Documentation

template<class Scalar , class Description > typedef std::vector<FeaturePoint<Scalar, Description> > meow::FeaturePointsMatch< Scalar, Description >::FeaturePoints

Definition at line 22 of file FeaturePointsMatch.h.

template<class Scalar , class Description > typedef std::vector<FeaturePoints > meow::FeaturePointsMatch< Scalar, Description >::FeaturePointss

Definition at line 23 of file FeaturePointsMatch.h.

6.15.3 Constructor & Destructor Documentation

template<class Scalar , class Description > meow::FeaturePointsMatch< Scalar, Description >::FeaturePointsMatch ( ) `[inline], [protected]`

Definition at line 19 of file FeaturePointsMatch.h.

template<class Scalar , class Description > virtual meow::FeaturePointsMatch< Scalar, Description >::∼FeaturePointsMatch ( ) `[inline], [virtual]`

Definition at line 25 of file FeaturePointsMatch.h.

6.15.4 Member Function Documentation

template< class Scalar , class Description > virtual FeaturePointIndexPairs meow::FeaturePointsMatch< Scalar, Description >::match (size_t *dimension*, FeaturePoints const & *from*, FeaturePoints const & *to*) const `[pure virtual]`

Implemented in meow::FeaturePointsMatch_K_Match< Scalar, Description >.

template< class Scalar , class Description > virtual FeaturePointIndexPairs meow::FeaturePointsMatch< Scalar, Description >::match (size_t *dimension*, FeaturePoints const & *from*, FeaturePointss const & *to*) const `[pure virtual]`

Implemented in meow::FeaturePointsMatch_K_Match< Scalar, Description >.

template< class Scalar , class Description > virtual FeaturePointIndexPairs meow::FeaturePointsMatch< Scalar, Description >::match (size_t *dimension*, FeaturePointss const & *from*, FeaturePointss const & *to*) const `[pure virtual]`

Implemented in meow::FeaturePointsMatch_K_Match< Scalar, Description >.

template< class Scalar , class Description > virtual FeaturePointIndexPairs meow::FeaturePointsMatch< Scalar, Description >::match (size_t *dimension*, FeaturePointss const & *fpss*) const `[pure virtual]`

Implemented in meow::FeaturePointsMatch_K_Match< Scalar, Description >.

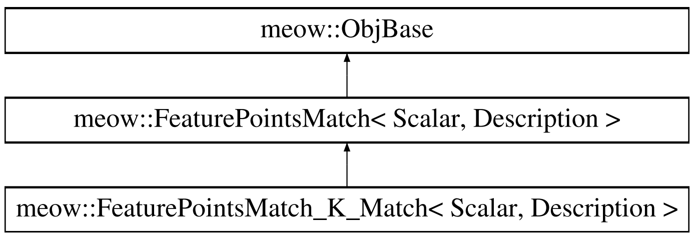
The documentation for this class was generated from the following file:

- meowpp/gra/FeaturePointsMatch.h

6.16 meow::FeaturePointsMatch_K_Match< Scalar, Description > Class Template Reference

`#include "FeaturePointsMatch_K_Match.h"`

Inheritance diagram for meow::FeaturePointsMatch_K_Match< Scalar, Description >:


meow::ObjBase
meow::FeaturePointsMatch< Scalar, Description >
meow::FeaturePointsMatch_K_Match< Scalar, Description >

Public Types

- typedef std::vector
  < FeaturePoint< Scalar,
  Description > > FeaturePoints
- typedef std::vector
  < FeaturePoints > FeaturePointss

Public Member Functions

- FPMKM ()
- FPMKM (FPMKM const &m)
- FPMKM (size_t k)
- ∼FPMKM ()

- FPMKM & copyFrom (FPMKM const &m)
- FPMKM & referenceFrom (FPMKM const &m)
- size_t paramK () const
- size_t paramK (size_t k)
- FeaturePointIndexPairs match (size_t dimension, FeaturePoints const &from, FeaturePoints const &to) const
- FeaturePointIndexPairs match (size_t dimension, FeaturePoints const &from, FeaturePointss const &to) const
- FeaturePointIndexPairs match (size_t dimension, FeaturePointss const &from, FeaturePointss const &to) const
- FeaturePointIndexPairs match (size_t dimension, FeaturePointss const &fpss) const
- FPMKM & operator= (FPMKM const &b)
- bool write (FILE ∗f, bool bin, unsigned int fg) const

  , implement `false`
- bool read (FILE ∗f, bool bin, unsigned int fg)

  , implement `false`
- ObjBase ∗ create () const

  new, implement `NULL`
- ObjBase ∗ copyFrom (ObjBase const ∗ptr)

  , operator=
- char const ∗ ctype () const

  C-style stringclasstype name
- std::string type () const

  std::stringclasstype name

Additional Inherited Members

6.16.1 Detailed Description

template<class Scalar, class Description>class meow::FeaturePointsMatch_K_Match< Scalar, Description >

Definition at line 15 of file FeaturePointsMatch_K_Match.h.

6.16.2 Member Typedef Documentation

template<class Scalar , class Description > typedef std::vector<FeaturePoint<Scalar, Description> > meow::FeaturePointsMatch_K_Match< Scalar, Description >::FeaturePoints

Definition at line 19 of file FeaturePointsMatch_K_Match.h.

template<class Scalar , class Description > typedef std::vector<FeaturePoints > meow::FeaturePointsMatch_K_Match< Scalar, Description >::FeaturePointss

Definition at line 20 of file FeaturePointsMatch_K_Match.h.

6.16.3 Constructor & Destructor Documentation

template<class Scalar , class Description > meow::FeaturePointsMatch_K_Match< Scalar, Description >::∼FPMKM ( ) `[inline]`

Definition at line 74 of file FeaturePointsMatch_K_Match.h.

6.16.4 Member Function Documentation

template<class Scalar , class Description > FPMKM& meow::FeaturePointsMatch_K_Match< Scalar, Description >::copyFrom (FPMKM const & *m*) `[inline]`

Definition at line 77 of file FeaturePointsMatch_K_Match.h.

template<class Scalar , class Description > ObjBase∗ meow::FeaturePointsMatch_K_Match< Scalar, Description >::copyFrom (ObjBase const ∗ *b*) `[inline]`, `[virtual]`

, operator=

Parameters

in	*b*	

Returns

`this`

Reimplemented from meow::ObjBase.
Definition at line 170 of file FeaturePointsMatch_K_Match.h.

template<class Scalar , class Description > ObjBase* meow::FeaturePointsMatch_K_Match< Scalar, Description >::create ( ) const [inline], [virtual]

new, implement `NULL`
Reimplemented from meow::ObjBase.
Definition at line 166 of file FeaturePointsMatch_K_Match.h.

template<class Scalar , class Description > char const* meow::FeaturePointsMatch_K_Match< Scalar, Description >::ctype ( ) const [inline], [virtual]

C-style stringclasstype name
Reimplemented from meow::ObjBase.
Definition at line 174 of file FeaturePointsMatch_K_Match.h.

template<class Scalar , class Description > meow::FeaturePointsMatch_K_Match< Scalar, Description >::FPMKM ( ) [inline]

Definition at line 64 of file FeaturePointsMatch_K_Match.h.

template<class Scalar , class Description > meow::FeaturePointsMatch_K_Match< Scalar, Description >::FPMKM (FPMKM const & *m*) [inline]

Definition at line 67 of file FeaturePointsMatch_K_Match.h.

template<class Scalar , class Description > meow::FeaturePointsMatch_K_Match< Scalar, Description >::FPMKM (size_t *k*) [inline]

Definition at line 71 of file FeaturePointsMatch_K_Match.h.

template<class Scalar , class Description > FeaturePointIndexPairs meow::FeaturePointsMatch_K_Match< Scalar, Description >::match (size_t *dimension,* FeaturePoints const & *from,* FeaturePoints const & *to*) const [inline], [virtual]

Implements meow::FeaturePointsMatch< Scalar, Description >.
Definition at line 97 of file FeaturePointsMatch_K_Match.h.

template<class Scalar , class Description > FeaturePointIndexPairs meow::FeaturePointsMatch_K_Match< Scalar, Description >::match (size_t *dimension,* FeaturePoints const & *from,* FeaturePointss const & *to*) const [inline], [virtual]

Implements meow::FeaturePointsMatch< Scalar, Description >.
Definition at line 104 of file FeaturePointsMatch_K_Match.h.

template<class Scalar , class Description > FeaturePointIndexPairs meow::FeaturePointsMatch_K_Match< Scalar, Description >::match (size_t *dimension,* FeaturePointss const & *from,* FeaturePointss const & *to*) const [inline], [virtual]

Implements meow::FeaturePointsMatch< Scalar, Description >.
Definition at line 110 of file FeaturePointsMatch_K_Match.h.

template<class Scalar , class Description > FeaturePointIndexPairs meow::FeaturePointsMatch_K_Match< Scalar, Description >::match (size_t *dimension,* FeaturePointss const & *fpss*) const [inline], [virtual]

Implements meow::FeaturePointsMatch< Scalar, Description >.

Definition at line 134 of file FeaturePointsMatch_K_Match.h.

template<class Scalar , class Description > FPMKM& meow::FeaturePointsMatch_K_Match< Scalar, Description >::operator= (FPMKM const & *b*) [inline]

Definition at line 151 of file FeaturePointsMatch_K_Match.h.

template<class Scalar , class Description > size_t meow::FeaturePointsMatch_K_Match< Scalar, Description >::paramK ( ) const [inline]

Definition at line 87 of file FeaturePointsMatch_K_Match.h.

template<class Scalar , class Description > size_t meow::FeaturePointsMatch_K_Match< Scalar, Description >::paramK (size_t *k*) [inline]

Definition at line 91 of file FeaturePointsMatch_K_Match.h.

template<class Scalar , class Description > bool meow::FeaturePointsMatch_K_Match< Scalar, Description >::read (FILE ∗ *f,* bool *bin,* unsigned int *fg*) [inline], [virtual]

, implement `false`

Parameters

in	*f*	
in	*bin*	binary
in	*fg*	argument

Returns

Reimplemented from meow::ObjBase.

Definition at line 161 of file FeaturePointsMatch_K_Match.h.

template<class Scalar , class Description > FPMKM& meow::FeaturePointsMatch_K_Match< Scalar, Description >::referenceFrom (FPMKM const & *m*) [inline]

Definition at line 82 of file FeaturePointsMatch_K_Match.h.

template<class Scalar , class Description > std::string meow::FeaturePointsMatch_K_Match< Scalar, Description >::type ( ) const [inline], [virtual]

std::stringclasstype name

Reimplemented from meow::ObjBase.

Definition at line 178 of file FeaturePointsMatch_K_Match.h.

template<class Scalar , class Description > bool meow::FeaturePointsMatch_K_Match< Scalar, Description >::write (FILE ∗ *f,* bool *bin,* unsigned int *fg*) const [inline], [virtual]

, implement `false`

Parameters

in	*f*	
in	*bin*	binary
in	*fg*	argument

Returns

Reimplemented from meow::ObjBase.
Definition at line 156 of file FeaturePointsMatch_K_Match.h.
The documentation for this class was generated from the following file:

- meowpp/gra/FeaturePointsMatch_K_Match.h

6.17 meow::HashTableList< Data, HashFunc > Class Template Reference

keylisthash_table
```
#include "HashTable.h"
```

Public Member Functions

- HashTableList ()

  constructor
- HashTableList (size_t size, HashFunc const &func)

  constructor
- ∼HashTableList ()

  destructor
- HashTableList & copyFrom (HashTableList const &b)

  copy
- void clear ()

- void reset (size_t size, HashFunc const &func)

  , sizehash function
- size_t tableSize () const

  table size
- size_t size () const

  element
- HashFunc const & func () const

  hash function
- bool add (Data const &e)

  element
- bool add (HashTableList const &h)

  HashTableListelement
- bool del (Data const &e)

  element
- bool del (HashTableList const &h)

  HashTableListelement
- bool exist (Data const &e) const

  element
- std::vector< Data > all () const

- std::vector< Data > all (size_t index) const *keyindex*

- HashTableList & operator= (HashTableList const &h)

  same as `copyFrom(h)`

- HashTableList & operator+= (HashTableList const &h)

  same as `add(h)`

- HashTableList & operator-= (HashTableList const &h)

  same as `del(h)`

6.17.1 Detailed Description

template<class Data, class HashFunc>class meow::HashTableList< Data, HashFunc >

keylisthash_table

Author

cat_leopard

Definition at line 15 of file HashTable.h.

6.17.2 Constructor & Destructor Documentation

template<class Data , class HashFunc > meow::HashTableList< Data, HashFunc >::HashTableList ( ) `[inline]`

constructor

Definition at line 23 of file HashTable.h.

template<class Data , class HashFunc > meow::HashTableList< Data, HashFunc >::HashTableList (size_t *size*, HashFunc const & *func*) `[inline]`

constructor

table size, hash function

Definition at line 31 of file HashTable.h.

template<class Data , class HashFunc > meow::HashTableList< Data, HashFunc >::~HashTableList ( ) `[inline]`

destructor

Definition at line 37 of file HashTable.h.

6.17.3 Member Function Documentation

template<class Data , class HashFunc > bool meow::HashTableList< Data, HashFunc >::add (Data const & *e*) `[inline]`

element

Definition at line 95 of file HashTable.h.

template<class Data , class HashFunc > bool meow::HashTableList< Data, HashFunc >::add (HashTableList< Data, HashFunc > const & *h*) `[inline]`

HashTableListelement

Definition at line 104 of file HashTable.h.

template<class Data , class HashFunc > std::vector<Data> meow::HashTableList< Data, HashFunc >::all ( ) const `[inline]`

Definition at line 173 of file HashTable.h.

template<class Data , class HashFunc > std::vector<Data> meow::HashTableList< Data, HashFunc >::all (size_t *index*) const `[inline]`

keyindex

Definition at line 187 of file HashTable.h.

template<class Data , class HashFunc > void meow::HashTableList< Data, HashFunc >::clear ( ) `[inline]`

Definition at line 52 of file HashTable.h.

template<class Data , class HashFunc > HashTableList& meow::HashTableList< Data, HashFunc >::copyFrom (HashTableList< Data, HashFunc > const & *b*) `[inline]`

copy

Definition at line 43 of file HashTable.h.

template<class Data , class HashFunc > bool meow::HashTableList< Data, HashFunc >::del (Data const & *e*) `[inline]`

element

Definition at line 117 of file HashTable.h.

template<class Data , class HashFunc > bool meow::HashTableList< Data, HashFunc >::del (HashTableList< Data, HashFunc > const & *h*) `[inline]`

HashTableListelement

Definition at line 132 of file HashTable.h.

template<class Data , class HashFunc > bool meow::HashTableList< Data, HashFunc >::exist (Data const & *e*) const `[inline]`

element

Definition at line 160 of file HashTable.h.

template<class Data , class HashFunc > HashFunc const& meow::HashTableList< Data, HashFunc >::func ( ) const `[inline]`

hash function

Definition at line 88 of file HashTable.h.

template<class Data , class HashFunc > HashTableList& meow::HashTableList< Data, HashFunc >::operator+= (HashTableList< Data, HashFunc > const & *h*) `[inline]`

same as `add(h)`

Definition at line 203 of file HashTable.h.

template<class Data , class HashFunc > HashTableList& meow::HashTableList< Data, HashFunc >::operator-= (HashTableList< Data, HashFunc > const & *h*) `[inline]`

same as `del(h)`

Definition at line 209 of file HashTable.h.

template<class Data , class HashFunc > HashTableList& meow::HashTableList< Data, HashFunc >::operator= (HashTableList< Data, HashFunc > const & *h*) `[inline]`

same as `copyFrom(h)`

Definition at line 198 of file HashTable.h.

template<class Data , class HashFunc > void meow::HashTableList< Data, HashFunc >::reset (size_t *size,* HashFunc const & *func*) `[inline]`

, sizehash function

Definition at line 61 of file HashTable.h.

template<class Data , class HashFunc > size_t meow::HashTableList< Data, HashFunc >::size ( ) const `[inline]`

element

Definition at line 77 of file HashTable.h.

template<class Data , class HashFunc > size_t meow::HashTableList< Data, HashFunc >::tableSize ( ) const `[inline]`

table size

Definition at line 70 of file HashTable.h.

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

- meowpp/dsa/HashTable.h

6.18 meow::HSL< T > Class Template Reference

`#include "HSL.h"`

Public Member Functions

- virtual ∼HSL ()
- virtual T hMax () const =0
- virtual T hMin () const =0
- virtual T sMax () const =0
- virtual T sMin () const =0
- virtual T lMax () const =0
- virtual T lMin () const =0
- T h () const
- T s () const
- T l () const
- T hsl (size_t i) const
- T lsh (size_t i) const
- T h (T const &val)
- T s (T const &val)
- T l (T const &val)
- T hsl (size_t i, T const &val)
- T lsh (size_t i, T const &val)

Protected Member Functions

- HSL ()
- HSL (T const &h, T const &s, T const &l)
- HSL (T const ∗hsl)

Protected Attributes

- T hsl_ [3]

6.18.1 Detailed Description

template<class T>class meow::HSL< T >

Definition at line 8 of file HSL.h.

6.18.2 Constructor & Destructor Documentation

template<class T > meow::HSL< T >::HSL ( ) `[inline]`, `[protected]`

Definition at line 9 of file HSL.hpp.

template<class T> meow::HSL< T >::HSL (T const & *h*, T const & *s*, T const & *l*) `[inline]`, `[protected]`

Definition at line 10 of file HSL.hpp.

template<class T> meow::HSL< T >::HSL (T const * *hsl*) `[inline]`, `[protected]`

Definition at line 13 of file HSL.hpp.

template<class T> virtual meow::HSL< T >::~HSL ( ) `[inline]`, `[virtual]`

Definition at line 15 of file HSL.h.

6.18.3 Member Function Documentation

template<class T > T meow::HSL< T >::h ( ) const `[inline]`

Definition at line 17 of file HSL.hpp.

template<class T> T meow::HSL< T >::h (T const & *val*) `[inline]`

Definition at line 24 of file HSL.hpp.

template<class T> virtual T meow::HSL< T >::hMax ( ) const `[pure virtual]`

Implemented in meow::HSLf.

template<class T> virtual T meow::HSL< T >::hMin ( ) const `[pure virtual]`

Implemented in meow::HSLf.

template<class T > T meow::HSL< T >::hsl (size_t *i*) const `[inline]`

Definition at line 20 of file HSL.hpp.

template<class T> T meow::HSL< T >::hsl (size_t *i*, T const & *val*) `[inline]`

Definition at line 27 of file HSL.hpp.

template<class T > T meow::HSL< T >::l ( ) const `[inline]`

Definition at line 19 of file HSL.hpp.

template<class T> T meow::HSL< T >::l (T const & *val*) `[inline]`

Definition at line 26 of file HSL.hpp.

template<class T> virtual T meow::HSL< T >::lMax ( ) const `[pure virtual]`

Implemented in meow::HSLf.

template<class T> virtual T meow::HSL< T >::lMin ( ) const `[pure virtual]`

Implemented in meow::HSLf.

template<class T > T meow::HSL< T >::lsh (size_t *i*) const `[inline]`

Definition at line 23 of file HSL.hpp.

template<class T> T meow::HSL< T >::lsh (size_t *i*, T const & *val*) `[inline]`

Definition at line 30 of file HSL.hpp.

template<class T > T meow::HSL< T >::s ( ) const `[inline]`

Definition at line 18 of file HSL.hpp.

template<class T> T meow::HSL< T >::s (T const & *val*) `[inline]`

Definition at line 25 of file HSL.hpp.

template<class T> virtual T meow::HSL< T >::sMax ( ) const `[pure virtual]`

Implemented in meow::HSLf.

template<class T> virtual T meow::HSL< T >::sMin ( ) const `[pure virtual]`

Implemented in meow::HSLf.

6.18.4 Member Data Documentation

template<class T> T meow::HSL< T >::hsl_[3] `[protected]`

Definition at line 10 of file HSL.h.

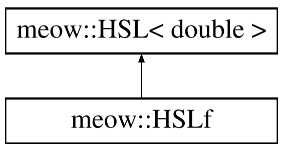
The documentation for this class was generated from the following files:

- meowpp/colors/HSL.h
- meowpp/colors/HSL.hpp

6.19 meow::HSLf Class Reference

`#include "HSL.h"`

Inheritance diagram for meow::HSLf:



Public Member Functions

- HSLf ()
- ∼HSLf ()
- HSLf (double const &h, double const &s, double const &l)
- HSLf (double const ∗hsl)
- double hMin () const
- double hMax () const
- double sMin () const

- double sMax () const
- double lMin () const
- double lMax () const

Additional Inherited Members

6.19.1 Detailed Description

Definition at line 37 of file HSL.h.

6.19.2 Constructor & Destructor Documentation

meow::HSLf::HSLf ( ) `[inline]`

Definition at line 38 of file HSL.hpp.

meow::HSLf::∼HSLf ( ) `[inline]`

Definition at line 39 of file HSL.hpp.

meow::HSLf::HSLf (double const & *h*, double const & *s*, double const & *l*) `[inline]`

Definition at line 40 of file HSL.hpp.

meow::HSLf::HSLf (double const ∗ *hsl*) `[inline]`

Definition at line 41 of file HSL.hpp.

6.19.3 Member Function Documentation

double meow::HSLf::hMax ( ) const `[inline]`, `[virtual]`

Implements meow::HSL< double >.

Definition at line 43 of file HSL.hpp.

double meow::HSLf::hMin ( ) const `[inline]`, `[virtual]`

Implements meow::HSL< double >.

Definition at line 42 of file HSL.hpp.

double meow::HSLf::lMax ( ) const `[inline]`, `[virtual]`

Implements meow::HSL< double >.

Definition at line 47 of file HSL.hpp.

double meow::HSLf::lMin ( ) const `[inline]`, `[virtual]`

Implements meow::HSL< double >.

Definition at line 46 of file HSL.hpp.

double meow::HSLf::sMax ( ) const `[inline]`, `[virtual]`

Implements meow::HSL< double >.

Definition at line 45 of file HSL.hpp.

double meow::HSLf::sMin ( ) const [inline], [virtual]

Implements meow::HSL< double >.

Definition at line 44 of file HSL.hpp.

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

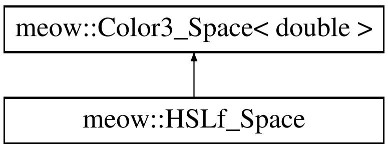
- meowpp/colors/HSL.h
- meowpp/colors/HSL.hpp

6.20 meow::HSLf_Space Class Reference

Y(), **U**(), **V**()

`#include "HSL_Space.h"`

Inheritance diagram for meow::HSLf_Space:



Public Member Functions

- HSLf_Space ()
- HSLf_Space (double c)
- HSLf_Space (Vector3D< double > const &v)
- HSLf_Space (HSL_Space const &b)
- ∼HSLf_Space ()
- double const & hslMin (size_t i) const
- double const & hMin () const
- double const & sMin () const
- double const & lMin () const
- double const & hslMax (size_t i) const
- double const & hMax () const
- double const & sMax () const
- double const & lMax () const
- double const & hsl (size_t i) const
- double const & h () const
- double const & s () const
- double const & l () const
- double const & hsl (size_t i, double c)
- double const & h (double c)
- double const & s (double c)
- double const & l (double c)
- double & hslGet (size_t i)
- double & hGet ()
- double & sGet ()
- double & lGet ()
- HSLf_Space & operator= (HSLf_Space const &b)
- HSLf_Space operator+ (HSLf_Space const &b) const
- HSLf_Space operator- (HSLf_Space const &b) const
- HSLf_Space operator∗ (double const &c) const
- HSLf_Space operator/ (double const &c) const
- double operator∗ (HSLf_Space const &b) const

Additional Inherited Members

6.20.1 Detailed Description

Y(), **U**(), **V**()
0.0∼1.0

Author

cat_leopard

Definition at line 22 of file HSL_Space.h.

6.20.2 Constructor & Destructor Documentation

meow::HSLf_Space::HSLf_Space ( ) `[inline]`

Definition at line 24 of file HSL_Space.h.

meow::HSLf_Space::HSLf_Space (double *c*) `[inline]`

Definition at line 28 of file HSL_Space.h.

meow::HSLf_Space::HSLf_Space (Vector3D< double > const & *v*) `[inline]`

Definition at line 32 of file HSL_Space.h.

meow::HSLf_Space::HSLf_Space (HSL_Space const & *b*) `[inline]`

Definition at line 37 of file HSL_Space.h.

meow::HSLf_Space::∼HSLf_Space ( ) `[inline]`

Definition at line 39 of file HSL_Space.h.

6.20.3 Member Function Documentation

double const& meow::HSLf_Space::h ( ) const `[inline]`

Definition at line 50 of file HSL_Space.h.

double const& meow::HSLf_Space::h (double *c*) `[inline]`

Definition at line 54 of file HSL_Space.h.

double& meow::HSLf_Space::hGet ( ) `[inline]`

Definition at line 58 of file HSL_Space.h.

double const& meow::HSLf_Space::hMax ( ) const `[inline]`

Definition at line 46 of file HSL_Space.h.

double const& meow::HSLf_Space::hMin ( ) const `[inline]`

Definition at line 42 of file HSL_Space.h.

double const& meow::HSLf_Space::hsl (size_t *i*) const `[inline]`

Definition at line 49 of file HSL_Space.h.

double const& meow::HSLf_Space::hsl (size_t *i*, double *c*) `[inline]`

Definition at line 53 of file HSL_Space.h.

double& meow::HSLf_Space::hslGet (size_t *i*) `[inline]`

Definition at line 57 of file HSL_Space.h.

double const& meow::HSLf_Space::hslMax (size_t *i*) const `[inline]`

Definition at line 45 of file HSL_Space.h.

double const& meow::HSLf_Space::hslMin (size_t *i*) const `[inline]`

Definition at line 41 of file HSL_Space.h.

double const& meow::HSLf_Space::l ( ) const `[inline]`

Definition at line 52 of file HSL_Space.h.

double const& meow::HSLf_Space::l (double *c*) `[inline]`

Definition at line 56 of file HSL_Space.h.

double& meow::HSLf_Space::lGet ( ) `[inline]`

Definition at line 60 of file HSL_Space.h.

double const& meow::HSLf_Space::lMax ( ) const `[inline]`

Definition at line 48 of file HSL_Space.h.

double const& meow::HSLf_Space::lMin ( ) const `[inline]`

Definition at line 44 of file HSL_Space.h.

HSLf_Space meow::HSLf_Space::operator* (double const & *c*) const `[inline]`

Definition at line 71 of file HSL_Space.h.

double meow::HSLf_Space::operator* (HSLf_Space const & *b*) const `[inline]`

Definition at line 77 of file HSL_Space.h.

HSLf_Space meow::HSLf_Space::operator+ (HSLf_Space const & *b*) const `[inline]`

Definition at line 65 of file HSL_Space.h.

HSLf_Space meow::HSLf_Space::operator- (HSLf_Space const & *b*) const `[inline]`

Definition at line 68 of file HSL_Space.h.

HSLf_Space meow::HSLf_Space::operator/ (double const & *c*) const `[inline]`

Definition at line 74 of file HSL_Space.h.

HSLf_Space& meow::HSLf_Space::operator= (HSLf_Space const & *b*) `[inline]`

Definition at line 61 of file HSL_Space.h.

double const& meow::HSLf_Space::s ( ) const [inline]

Definition at line 51 of file HSL_Space.h.

double const& meow::HSLf_Space::s (double *c*) [inline]

Definition at line 55 of file HSL_Space.h.

double& meow::HSLf_Space::sGet ( ) [inline]

Definition at line 59 of file HSL_Space.h.

double const& meow::HSLf_Space::sMax ( ) const [inline]

Definition at line 47 of file HSL_Space.h.

double const& meow::HSLf_Space::sMin ( ) const [inline]

Definition at line 43 of file HSL_Space.h.

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

- meowpp/colors/HSL_Space.h

6.21 meow::HSV< T > Class Template Reference

`#include "HSV.h"`

Public Member Functions

- virtual ∼HSV ()
- virtual T hMax () const =0
- virtual T hMin () const =0
- virtual T sMax () const =0
- virtual T sMin () const =0
- virtual T vMax () const =0
- virtual T vMin () const =0
- T h () const
- T s () const
- T v () const
- T hsv (size_t i) const
- T vsh (size_t i) const
- T h (T const &val)
- T s (T const &val)
- T v (T const &val)
- T hsv (size_t i, T const &val)
- T vsh (size_t i, T const &val)

Protected Member Functions

- HSV ()
- HSV (T const &h, T const &s, T const &v)
- HSV (T const ∗hsv)

Protected Attributes

- T hsv_ [3]

6.21.1 Detailed Description

template<class T>class meow::HSV< T >

Definition at line 9 of file HSV.h.

6.21.2 Constructor & Destructor Documentation

template<class T > meow::HSV< T >::HSV ( ) `[inline], [protected]`

Definition at line 10 of file HSV.hpp.

template<class T> meow::HSV< T >::HSV (T const & *h*, T const & *s*, T const & *v*) `[inline], [protected]`

Definition at line 11 of file HSV.hpp.

template<class T> meow::HSV< T >::HSV (T const ∗ *hsv*) `[inline], [protected]`

Definition at line 14 of file HSV.hpp.

template<class T> virtual meow::HSV< T >::~HSV ( ) `[inline], [virtual]`

Definition at line 16 of file HSV.h.

6.21.3 Member Function Documentation

template<class T > T meow::HSV< T >::h ( ) const `[inline]`

Definition at line 18 of file HSV.hpp.

template<class T> T meow::HSV< T >::h (T const & *val*) `[inline]`

Definition at line 25 of file HSV.hpp.

template<class T> virtual T meow::HSV< T >::hMax ( ) const `[pure virtual]`

Implemented in meow::HSVf.

template<class T> virtual T meow::HSV< T >::hMin ( ) const `[pure virtual]`

Implemented in meow::HSVf.

template<class T > T meow::HSV< T >::hsv (size_t *i*) const `[inline]`

Definition at line 21 of file HSV.hpp.

template<class T> T meow::HSV< T >::hsv (size_t *i*, T const & *val*) `[inline]`

Definition at line 28 of file HSV.hpp.

template<class T > T meow::HSV< T >::s ( ) const `[inline]`

Definition at line 19 of file HSV.hpp.

template<class T> T meow::HSV< T >::s (T const & *val*) `[inline]`

Definition at line 26 of file HSV.hpp.

template<class T> virtual T meow::HSV< T >::sMax ( ) const `[pure virtual]`

Implemented in meow::HSVf.

template<class T> virtual T meow::HSV< T >::sMin ( ) const `[pure virtual]`

Implemented in meow::HSVf.

template<class T > T meow::HSV< T >::v ( ) const `[inline]`

Definition at line 20 of file HSV.hpp.

template<class T> T meow::HSV< T >::v (T const & *val*) `[inline]`

Definition at line 27 of file HSV.hpp.

template<class T> virtual T meow::HSV< T >::vMax ( ) const `[pure virtual]`

Implemented in meow::HSVf.

template<class T> virtual T meow::HSV< T >::vMin ( ) const `[pure virtual]`

Implemented in meow::HSVf.

template<class T > T meow::HSV< T >::vsh (size_t *i*) const `[inline]`

Definition at line 24 of file HSV.hpp.

template<class T> T meow::HSV< T >::vsh (size_t *i*, T const & *val*) `[inline]`

Definition at line 31 of file HSV.hpp.

6.21.4 Member Data Documentation

template<class T> T meow::HSV< T >::hsv_[3] `[protected]`

Definition at line 11 of file HSV.h.

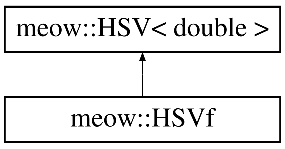
The documentation for this class was generated from the following files:

- meowpp/colors/HSV.h
- meowpp/colors/HSV.hpp

6.22 meow::HSVf Class Reference

```
#include "HSV.h"
```

Inheritance diagram for meow::HSVf:



Public Member Functions

- HSVf ()
- ∼HSVf ()
- HSVf (double const &h, double const &s, double const &v)
- HSVf (double const ∗hsv)
- double hMin () const
- double hMax () const
- double sMin () const

- double sMax () const
- double vMin () const
- double vMax () const

Additional Inherited Members

6.22.1 Detailed Description

Definition at line 38 of file HSV.h.

6.22.2 Constructor & Destructor Documentation

meow::HSVf::HSVf ( ) `[inline]`

Definition at line 39 of file HSV.hpp.

meow::HSVf::∼HSVf ( ) `[inline]`

Definition at line 40 of file HSV.hpp.

meow::HSVf::HSVf (double const & *h*, double const & *s*, double const & *v*) `[inline]`

Definition at line 41 of file HSV.hpp.

meow::HSVf::HSVf (double const ∗ *hsv*) `[inline]`

Definition at line 42 of file HSV.hpp.

6.22.3 Member Function Documentation

double meow::HSVf::hMax ( ) const `[inline]`, `[virtual]`

Implements meow::HSV< double >.
Definition at line 44 of file HSV.hpp.

double meow::HSVf::hMin ( ) const `[inline]`, `[virtual]`

Implements meow::HSV< double >.
Definition at line 43 of file HSV.hpp.

double meow::HSVf::sMax ( ) const `[inline]`, `[virtual]`

Implements meow::HSV< double >.
Definition at line 46 of file HSV.hpp.

double meow::HSVf::sMin ( ) const `[inline]`, `[virtual]`

Implements meow::HSV< double >.
Definition at line 45 of file HSV.hpp.

double meow::HSVf::vMax ( ) const `[inline]`, `[virtual]`

Implements meow::HSV< double >.
Definition at line 48 of file HSV.hpp.

double meow::HSVf::vMin ( ) const [inline], [virtual]

Implements meow::HSV< double >.

Definition at line 47 of file HSV.hpp.

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

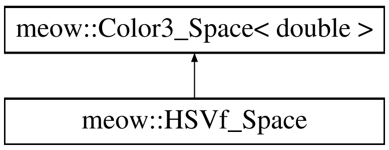
- meowpp/colors/HSV.h
- meowpp/colors/HSV.hpp

6.23 meow::HSVf_Space Class Reference

Y(), **U**(), **V**()

`#include "HSV_Space.h"`

Inheritance diagram for meow::HSVf_Space:



Public Member Functions

- HSVf_Space ()
- HSVf_Space (double c)
- HSVf_Space (Vector3D< double > const &v)
- HSVf_Space (HSV_Space const &b)
- ∼HSVf_Space ()
- double const & hsvMin (size_t i) const
- double const & hMin () const
- double const & sMin () const
- double const & vMin () const
- double const & hsvMax (size_t i) const
- double const & hMax () const
- double const & sMax () const
- double const & vMax () const
- double const & hsv (size_t i) const
- double const & h () const
- double const & s () const
- double const & v () const
- double const & hsv (size_t i, double c)
- double const & h (double c)
- double const & s (double c)
- double const & v (double c)
- double & hsvGet (size_t i)
- double & hGet ()
- double & sGet ()
- double & vGet ()
- HSVf_Space & operator= (HSVf_Space const &b)
- HSVf_Space operator+ (HSVf_Space const &b) const
- HSVf_Space operator- (HSVf_Space const &b) const
- HSVf_Space operator∗ (double const &c) const
- HSVf_Space operator/ (double const &c) const
- double operator∗ (HSVf_Space const &b) const

Additional Inherited Members

6.23.1 Detailed Description

Y(), **U**(), **V**()
0.0∼1.0

Author

cat_leopard

Definition at line 23 of file HSV_Space.h.

6.23.2 Constructor & Destructor Documentation

meow::HSVf_Space::HSVf_Space ( ) `[inline]`

Definition at line 25 of file HSV_Space.h.

meow::HSVf_Space::HSVf_Space (double *c*) `[inline]`

Definition at line 29 of file HSV_Space.h.

meow::HSVf_Space::HSVf_Space (Vector3D< double > const & *v*) `[inline]`

Definition at line 33 of file HSV_Space.h.

meow::HSVf_Space::HSVf_Space (HSV_Space const & *b*) `[inline]`

Definition at line 38 of file HSV_Space.h.

meow::HSVf_Space::∼HSVf_Space ( ) `[inline]`

Definition at line 40 of file HSV_Space.h.

6.23.3 Member Function Documentation

double const& meow::HSVf_Space::h ( ) const `[inline]`

Definition at line 51 of file HSV_Space.h.

double const& meow::HSVf_Space::h (double *c*) `[inline]`

Definition at line 55 of file HSV_Space.h.

double& meow::HSVf_Space::hGet ( ) `[inline]`

Definition at line 59 of file HSV_Space.h.

double const& meow::HSVf_Space::hMax ( ) const `[inline]`

Definition at line 47 of file HSV_Space.h.

double const& meow::HSVf_Space::hMin ( ) const `[inline]`

Definition at line 43 of file HSV_Space.h.

double const& meow::HSVf_Space::hsv (size_t *i*) const `[inline]`

Definition at line 50 of file HSV_Space.h.

double const& meow::HSVf_Space::hsv (size_t *i*, double *c*) `[inline]`

Definition at line 54 of file HSV_Space.h.

double& meow::HSVf_Space::hsvGet (size_t *i*) `[inline]`

Definition at line 58 of file HSV_Space.h.

double const& meow::HSVf_Space::hsvMax (size_t *i*) const `[inline]`

Definition at line 46 of file HSV_Space.h.

double const& meow::HSVf_Space::hsvMin (size_t *i*) const `[inline]`

Definition at line 42 of file HSV_Space.h.

HSVf_Space meow::HSVf_Space::operator∗ (double const & *c*) const `[inline]`

Definition at line 72 of file HSV_Space.h.

double meow::HSVf_Space::operator∗ (HSVf_Space const & *b*) const `[inline]`

Definition at line 78 of file HSV_Space.h.

HSVf_Space meow::HSVf_Space::operator+ (HSVf_Space const & *b*) const `[inline]`

Definition at line 66 of file HSV_Space.h.

HSVf_Space meow::HSVf_Space::operator- (HSVf_Space const & *b*) const `[inline]`

Definition at line 69 of file HSV_Space.h.

HSVf_Space meow::HSVf_Space::operator/ (double const & *c*) const `[inline]`

Definition at line 75 of file HSV_Space.h.

HSVf_Space& meow::HSVf_Space::operator= (HSVf_Space const & *b*) `[inline]`

Definition at line 62 of file HSV_Space.h.

double const& meow::HSVf_Space::s ( ) const `[inline]`

Definition at line 52 of file HSV_Space.h.

double const& meow::HSVf_Space::s (double *c*) `[inline]`

Definition at line 56 of file HSV_Space.h.

double& meow::HSVf_Space::sGet ( ) `[inline]`

Definition at line 60 of file HSV_Space.h.

double const& meow::HSVf_Space::sMax ( ) const `[inline]`

Definition at line 48 of file HSV_Space.h.

double const& meow::HSVf_Space::sMin ( ) const `[inline]`

Definition at line 44 of file HSV_Space.h.

double const& meow::HSVf_Space::v ( ) const [inline]

Definition at line 53 of file HSV_Space.h.

double const& meow::HSVf_Space::v (double *c*) [inline]

Definition at line 57 of file HSV_Space.h.

double& meow::HSVf_Space::vGet ( ) [inline]

Definition at line 61 of file HSV_Space.h.

double const& meow::HSVf_Space::vMax ( ) const [inline]

Definition at line 49 of file HSV_Space.h.

double const& meow::HSVf_Space::vMin ( ) const [inline]

Definition at line 45 of file HSV_Space.h.

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

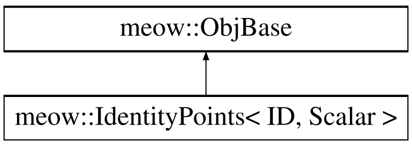
- meowpp/colors/HSV_Space.h

6.24 meow::IdentityPoints< ID, Scalar > Class Template Reference

`std::map<ID,`Vector<Scalar> >

`#include "IdentityPoints.h"`

Inheritance diagram for meow::IdentityPoints< ID, Scalar >:



Public Types

- typedef std::map< ID, Vector
  < Scalar > > IdentityPointsMap
- typedef IdentityPointsMap::iterator IdentityPointsMapIter
- typedef
  IdentityPointsMap::const_iterator IdentityPointsMapIterK

Public Member Functions

- IdentityPoints ()

  constructor
- IdentityPoints (IdentityPoints const &b)

  constructor,
- ∼IdentityPoints ()

  destructor
- IdentityPoints & copyFrom (IdentityPoints const &b)
- IdentityPoints & referenceFrom (IdentityPoints const &b)
- void clear ()

  identity points
- size_t size () const

  identity points
- bool empty () const

  identity points
- bool exist (ID const &id) const

  id
- size_t dimension () const

  dimension
- size_t dimension (size_t dim)

  dimension,
- size_t dimension (size_t dim, Scalar const &init_value)

  dimension, identity pointdimension
- IdentityPointsMap const & identityPoints () const

  identity points
- IdentityPointsMap const & identityPoints (IdentityPointsMap const &points)

  identity points
- IdentityPointsMap const & identityPointsAdd (IdentityPointsMap const &points)

  identity Points
- IdentityPointsMap const & identityPointsDel (std::set< ID > const &ids)

  identity Points
- Vector< Scalar > identityPoint (ID const &id) const

  identity point
- Vector< Scalar > identityPoint (ID const &id, Vector< Scalar > const &b)

  identity point
- Vector< Scalar > identityPointAdd (ID const &id, Vector< Scalar > const &b)

  identity point
- void identityPointDel (ID const &id)

  identity point
- Vector< Scalar > & identityPointGet (ID const &id)

  identity point, non-constant reference
- IdentityPoints & operator= (IdentityPoints const &b)

  same as `copyFrom(b)`
- bool write (FILE *f, bool bin, unsigned int fg) const

- bool read (FILE *f, bool bin, unsigned int fg)

- ObjBase * create () const

  new
- ObjBase * copyFrom (ObjBase const *b)

- char const * ctype () const

  classtype
- std::string type () const

  classtype

Additional Inherited Members

6.24.1 Detailed Description

template<class ID, class Scalar>class meow::IdentityPoints< ID, Scalar >

`std::map<ID`,Vector<Scalar> >

Author

cat_leopard

Definition at line 21 of file IdentityPoints.h.

6.24.2 Member Typedef Documentation

template<class ID, class Scalar> typedef std::map<ID, Vector<Scalar> > meow::IdentityPoints< ID, Scalar >::IdentityPointsMap

Definition at line 23 of file IdentityPoints.h.

template<class ID, class Scalar> typedef IdentityPointsMap:: iterator meow::IdentityPoints< ID, Scalar >::IdentityPointsMapIter

Definition at line 24 of file IdentityPoints.h.

template<class ID, class Scalar> typedef IdentityPointsMap::const_iterator meow::IdentityPoints< ID, Scalar >::IdentityPointsMapIterK

Definition at line 25 of file IdentityPoints.h.

6.24.3 Constructor & Destructor Documentation

template<class ID, class Scalar> meow::IdentityPoints< ID, Scalar >::IdentityPoints ( ) `[inline]`

constructor
Definition at line 46 of file IdentityPoints.h.

template<class ID, class Scalar> meow::IdentityPoints< ID, Scalar >::IdentityPoints (IdentityPoints< ID, Scalar > const & *b*) `[inline]`

constructor,
Definition at line 52 of file IdentityPoints.h.

template<class ID, class Scalar> meow::IdentityPoints< ID, Scalar >::∼IdentityPoints ( ) `[inline]`

destructor
Definition at line 59 of file IdentityPoints.h.

6.24.4 Member Function Documentation

template<class ID, class Scalar> void meow::IdentityPoints< ID, Scalar >::clear ( ) `[inline]`

identity points
Definition at line 81 of file IdentityPoints.h.

template<class ID, class Scalar> IdentityPoints& meow::IdentityPoints< ID, Scalar >::copyFrom (IdentityPoints< ID, Scalar > const & *b*) `[inline]`

Definition at line 65 of file IdentityPoints.h.

template<class ID, class Scalar> ObjBase* meow::IdentityPoints< ID, Scalar >::copyFrom (ObjBase const * *b*) `[inline]`, `[virtual]`

`ObjBase` `const*` `Bitmap`. method`copyFrom`

Parameters

`in`	*b*	

Returns

this

Reimplemented from meow::ObjBase.
Definition at line 308 of file IdentityPoints.h.

template<class ID, class Scalar> ObjBase∗ meow::IdentityPoints< ID, Scalar >::create ( ) const `[inline]`, `[virtual]`

new

Returns

newBitmap<Pixel>

Reimplemented from meow::ObjBase.
Definition at line 295 of file IdentityPoints.h.

template<class ID, class Scalar> char const∗ meow::IdentityPoints< ID, Scalar >::ctype ( ) const `[inline]`, `[virtual]`

classtype

Returns

`char const∗ typename`

Reimplemented from meow::ObjBase.
Definition at line 316 of file IdentityPoints.h.

template<class ID, class Scalar> size_t meow::IdentityPoints< ID, Scalar >::dimension ( ) const `[inline]`

dimension
Definition at line 109 of file IdentityPoints.h.

template<class ID, class Scalar> size_t meow::IdentityPoints< ID, Scalar >::dimension (size_t *dim*) `[inline]`

dimension,
Definition at line 116 of file IdentityPoints.h.

template<class ID, class Scalar> size_t meow::IdentityPoints< ID, Scalar >::dimension (size_t *dim,* Scalar const & *init_value*) `[inline]`

dimension, identity pointdimension
Definition at line 125 of file IdentityPoints.h.

template<class ID, class Scalar> bool meow::IdentityPoints< ID, Scalar >::empty ( ) const `[inline]`

identity points
Definition at line 95 of file IdentityPoints.h.

template<class ID, class Scalar> bool meow::IdentityPoints< ID, Scalar >::exist (ID const & *id*) const `[inline]`

id
Definition at line 102 of file IdentityPoints.h.

template<class ID, class Scalar> Vector<Scalar> meow::IdentityPoints< ID, Scalar >::identityPoint (ID const & *id*) const `[inline]`

identity point

Definition at line 173 of file IdentityPoints.h.

template<class ID, class Scalar> Vector<Scalar> meow::IdentityPoints< ID, Scalar >::identityPoint (ID const & *id*, Vector< Scalar > const & *b*) `[inline]`

identity point

Definition at line 180 of file IdentityPoints.h.

template<class ID, class Scalar> Vector<Scalar> meow::IdentityPoints< ID, Scalar >::identityPointAdd (ID const & *id*, Vector< Scalar > const & *b*) `[inline]`

identity point

Definition at line 190 of file IdentityPoints.h.

template<class ID, class Scalar> void meow::IdentityPoints< ID, Scalar >::identityPointDel (ID const & *id*) `[inline]`

identity point

Definition at line 200 of file IdentityPoints.h.

template<class ID, class Scalar> Vector<Scalar>& meow::IdentityPoints< ID, Scalar >::identityPointGet (ID const & *id*) `[inline]`

identity point, non-constant reference

Definition at line 207 of file IdentityPoints.h.

template<class ID, class Scalar> IdentityPointsMap const& meow::IdentityPoints< ID, Scalar >::identityPoints ( ) const `[inline]`

identity points

Definition at line 137 of file IdentityPoints.h.

template<class ID, class Scalar> IdentityPointsMap const& meow::IdentityPoints< ID, Scalar >::identityPoints (IdentityPointsMap const & *points*) `[inline]`

identity points

Definition at line 144 of file IdentityPoints.h.

template<class ID, class Scalar> IdentityPointsMap const& meow::IdentityPoints< ID, Scalar >::identityPointsAdd (IdentityPointsMap const & *points*) `[inline]`

identity Points

Definition at line 152 of file IdentityPoints.h.

template<class ID, class Scalar> IdentityPointsMap const& meow::IdentityPoints< ID, Scalar >::identityPointsDel (std::set< ID > const & *ids*) `[inline]`

identity Points

Definition at line 162 of file IdentityPoints.h.

template<class ID, class Scalar> IdentityPoints& meow::IdentityPoints< ID, Scalar >::operator= (IdentityPoints< ID, Scalar > const & *b*) `[inline]`

same as `copyFrom(b)`

Definition at line 214 of file IdentityPoints.h.

template<class ID, class Scalar> bool meow::IdentityPoints< ID, Scalar >::read (FILE ∗ *f*, bool *bin*, unsigned int *fg*) [inline], [virtual]

Note

Reimplemented from meow::ObjBase.
Definition at line 257 of file IdentityPoints.h.

template<class ID, class Scalar> IdentityPoints& meow::IdentityPoints< ID, Scalar >::referenceFrom (IdentityPoints< ID, Scalar > const & *b*) [inline]

Definition at line 73 of file IdentityPoints.h.

template<class ID, class Scalar> size_t meow::IdentityPoints< ID, Scalar >::size ( ) const [inline]

identity points
Definition at line 88 of file IdentityPoints.h.

template<class ID, class Scalar> std::string meow::IdentityPoints< ID, Scalar >::type ( ) const [inline], [virtual]

classtype

Returns

`std::string` typename

Reimplemented from meow::ObjBase.
Definition at line 324 of file IdentityPoints.h.

template<class ID, class Scalar> bool meow::IdentityPoints< ID, Scalar >::write (FILE ∗ *f*, bool *bin*, unsigned int *fg*) const [inline], [virtual]

Note

Reimplemented from meow::ObjBase.
Definition at line 222 of file IdentityPoints.h.
The documentation for this class was generated from the following file:

- meowpp/gra/IdentityPoints.h

6.25 meow::ImplementInterface< T > Class Template Reference

`#include "Register_Implement.h"`

Public Member Functions

- T const & identify () const
- virtual ∼ImplementInterface ()

Protected Member Functions

- ImplementInterface (T const &id)

6.25.1 Detailed Description

template<class T>class meow::ImplementInterface< T >

Definition at line 7 of file Register_Implement.h.

6.25.2 Constructor & Destructor Documentation

template<class T> meow::ImplementInterface< T >::ImplementInterface (T const & *id*) `[inline]`, `[protected]`

Definition at line 11 of file Register_Implement.h.

template<class T> virtual meow::ImplementInterface< T >::~ImplementInterface ( ) `[inline]`, `[virtual]`

Definition at line 14 of file Register_Implement.h.

6.25.3 Member Function Documentation

template<class T> T const& meow::ImplementInterface< T >::identify ( ) const `[inline]`

Definition at line 13 of file Register_Implement.h.
The documentation for this class was generated from the following file:

- meowpp/oo/Register_Implement.h

6.26 meow::KD_Tree< Vector, Scalar > Class Template Reference

`k-dimension` tree
`#include "KD_Tree.h"`

Public Types

- typedef std::vector< Vector > Vectors

  Custom Type: Vectors is `std::vector<Vector>`

Public Member Functions

- KD_Tree ()

  constructor, with dimension = 1
- KD_Tree (size_t dimension)

  constructor, given dimension
- ~KD_Tree ()

  destructor
- void insert (Vector const &v)

  Vectorset
- bool erase (Vector const &v)

  Vectorset
- void build ()

  insert/erase `rebuild()`
- void forceBuild ()

- Vectors query (Vector const &v, size_t nearestNumber, bool compareWholeVector) const

- void clear ()

- void reset (size_t dimension)

6.26.1 Detailed Description

template<class Vector, class Scalar>class meow::KD_Tree< Vector, Scalar >

`k-dimension` tree
 k-dimension tree, **NK**, set **i**

Template Class Operators Request

const?	Typename	Operator	Parameters	Return Type	Description
const	Vector	operator[]	(size_t n)	Scalar	n
const	Vector	operator<	(Vector v)	bool	
const	Scalar	operator*	(Scalar s)	Scalar	
const	Scalar	operator+	(Scalar s)	Scalar	
const	Scalar	operator-	(Scalar s)	Scalar	
const	Scalar	operator<	(Scalar s)	bool	

Note

: N >> 2^K , K ,

Author

cat_leopard

Definition at line 40 of file KD_Tree.h.

6.26.2 Member Typedef Documentation

template<class Vector , class Scalar > typedef std::vector<Vector> meow::KD_Tree< Vector, Scalar >::Vectors

Custom Type: Vectors is `std::vector<Vector>`
 Definition at line 189 of file KD_Tree.h.

6.26.3 Constructor & Destructor Documentation

template<class Vector , class Scalar > meow::KD_Tree< Vector, Scalar >::KD_Tree ( ) `[inline]`

constructor, with dimension = 1
 Definition at line 192 of file KD_Tree.h.

template<class Vector , class Scalar > meow::KD_Tree< Vector, Scalar >::KD_Tree (size_t *dimension*) `[inline]`

constructor, given dimension
 Definition at line 196 of file KD_Tree.h.

template<class Vector , class Scalar > meow::KD_Tree< Vector, Scalar >::~KD_Tree ( ) `[inline]`

destructor
 Definition at line 201 of file KD_Tree.h.

6.26.4 Member Function Documentation

template<class Vector , class Scalar > void meow::KD_Tree< Vector, Scalar >::build ( ) `[inline]`

insert/erase `rebuild()`
 Definition at line 231 of file KD_Tree.h.

template<class Vector , class Scalar > void meow::KD_Tree< Vector, Scalar >::clear ( ) `[inline]`

Definition at line 286 of file KD_Tree.h.

template< class Vector , class Scalar > bool meow::KD_Tree< Vector, Scalar >::erase (Vector const & *v*) `[inline]`

Vectorset

Definition at line 215 of file KD_Tree.h.

template< class Vector , class Scalar > void meow::KD_Tree< Vector, Scalar >::forceBuild ( ) `[inline]`

Definition at line 240 of file KD_Tree.h.

template< class Vector , class Scalar > void meow::KD_Tree< Vector, Scalar >::insert (Vector const & *v*) `[inline]`

Vectorset

Definition at line 207 of file KD_Tree.h.

template< class Vector , class Scalar > Vectors meow::KD_Tree< Vector, Scalar >::query (Vector const & *v*, size_t *nearestNumber*, bool *compareWholeVector*) const `[inline]`

set `i , . v1`,v2 `, cmp true , v1<v2 . .`

Definition at line 263 of file KD_Tree.h.

template< class Vector , class Scalar > void meow::KD_Tree< Vector, Scalar >::reset (size_t *dimension*) `[inline]`

Definition at line 295 of file KD_Tree.h.

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

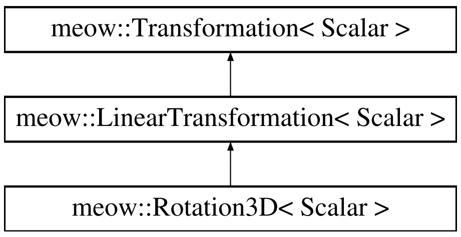
- meowpp/dsa/KD_Tree.h

6.27 meow::LinearTransformation< Scalar > Class Template Reference

A base class for implementing kinds of linear transformations.

`#include "LinearTransformation.h"`

Inheritance diagram for meow::LinearTransformation< Scalar >:


meow::Transformation< Scalar >
meow::LinearTransformation< Scalar >
meow::Rotation3D< Scalar >

Public Member Functions

- virtual ~LinearTransformation ()
- virtual Matrix< Scalar > const & matrix () const

  Return the matrix form of this transformation.
- virtual Matrix< Scalar > matrixInv () const

  Return the inverse of the matrix form of this transformate.

Protected Member Functions

- LinearTransformation (size_t inputRows, size_t outputRows, size_t psize)
- LinearTransformation (size_t inputRows, size_t outputRows, size_t psize, Matrix< Scalar > const &m)
- LinearTransformation (LinearTransformation const &b)
- LinearTransformation & copyFrom (LinearTransformation const &b)

  Copy settings, matrix from another LinearTransformation.
- LinearTransformation & referenceFrom (LinearTransformation const &b)

  Reference settings, matrix from another LinearTransformation.
- virtual Matrix< Scalar > const & matrix (Matrix< Scalar > const &m)

  setup the matrix

6.27.1 Detailed Description

template<class Scalar>class meow::LinearTransformation< Scalar >

A base class for implementing kinds of linear transformations.

Because all linear transformations belong to transformations, this class inherit to Transformation.

Author

cat_leopard

Definition at line 20 of file LinearTransformation.h.

6.27.2 Constructor & Destructor Documentation

template<class Scalar> meow::LinearTransformation< Scalar >::LinearTransformation (size_t *inputRows*, size_t *outputRows*, size_t *psize*) `[inline], [protected]`

Constructor with input/output size gived

Definition at line 27 of file LinearTransformation.h.

template<class Scalar> meow::LinearTransformation< Scalar >::LinearTransformation (size_t *inputRows*, size_t *outputRows*, size_t *psize*, Matrix< Scalar > const & *m*) `[inline], [protected]`

Constructor with input/output size gived and a inital matrix

Definition at line 35 of file LinearTransformation.h.

template<class Scalar> meow::LinearTransformation< Scalar >::LinearTransformation (LinearTransformation< Scalar > const & *b*) `[inline], [protected]`

Constructor with another LinearTransformation

Parameters

in	*b*	another LinearTransformation

Definition at line 46 of file LinearTransformation.h.

template<class Scalar> virtual meow::LinearTransformation< Scalar >::~LinearTransformation ( ) `[inline], [virtual]`

Destructor

Definition at line 85 of file LinearTransformation.h.

6.27.3 Member Function Documentation

template<class Scalar> LinearTransformation& meow::LinearTransformation< Scalar >::copyFrom (LinearTransformation< Scalar > const & *b*) `[inline], [protected]`

Copy settings, matrix from another LinearTransformation.

Parameters

`in`	*b*	another LinearTransformation

Definition at line 56 of file LinearTransformation.h.

template<class Scalar> virtual Matrix<Scalar> const& meow::LinearTransformation< Scalar >::matrix (Matrix< Scalar > const & *m*) `[inline]`, `[protected]`, `[virtual]`

setup the matrix

Definition at line 76 of file LinearTransformation.h.

template<class Scalar> virtual Matrix<Scalar> const& meow::LinearTransformation< Scalar >::matrix ( ) const `[inline]`, `[virtual]`

Return the matrix form of this transformation.

Returns

A matrix

Definition at line 93 of file LinearTransformation.h.

template<class Scalar> virtual Matrix<Scalar> meow::LinearTransformation< Scalar >::matrixInv ( ) const `[inline]`, `[virtual]`

Return the inverse of the matrix form of this transformate.

Returns

A matrix (may be invalid)

Reimplemented in meow::Rotation3D< Scalar >, and meow::Rotation3D< double >.
Definition at line 102 of file LinearTransformation.h.

template<class Scalar> LinearTransformation& meow::LinearTransformation< Scalar >::referenceFrom (LinearTransformation< Scalar > const & *b*) `[inline]`, `[protected]`

Reference settings, matrix from another LinearTransformation.

Parameters

`in`	*b*	another LinearTransformation

Definition at line 67 of file LinearTransformation.h.
The documentation for this class was generated from the following file:

- meowpp/math/LinearTransformation.h

6.28 meow::Matrix< Entry > Class Template Reference

matrix

```
#include "Matrix.h"
```

Public Types

- typedef std::vector< Entry >
  ::reference EntryRef
- typedef std::vector< Entry >
  ::const_reference EntryRefK

Public Member Functions

- Matrix ()

  constructor
- Matrix (Matrix const &m)

  constructor
- Matrix (size_t r, size_t c, Entry const &e)

  constructor
- ∼Matrix ()

  destructor
- Matrix & copyFrom (Matrix const &m)

  copy
- Matrix & referenceFrom (Matrix const &m)

  reference
- void reset (size_t r, size_t c, Entry const &e)

  reset the size of the matrix to r x c with entry all be e
- bool valid () const

  Return whether it is a ***valid*** *matrix.*
- size_t rows () const

  Return number of rows.
- size_t cols () const

  Return number of cols.
- size_t size () const

  Return number of rows times number of cols.
- size_t rows (size_t r, Entry const &e)

  resize the matrix such that number of rows become r.
- size_t cols (size_t c, Entry const &e)

  resize the matrix such that number of cols become c
- size_t size (size_t r, size_t c, Entry const &e)

  resize
- Entry entry (size_t r, size_t c) const

  Access the entry at r x c.
- Entry entry (size_t r, size_t c, Entry const &e)

  Change the entry at r x c.
- EntryRef entryGet (size_t r, size_t c)

  Get the entry at r x c.
- void entries (ssize_t rFirst, ssize_t rLast, ssize_t cFirst, ssize_t cLast, Entry const &e)

  Change the entries from rFirst x cFirst to rLast x cLast.
- Matrix subMatrix (size_t rFirst, size_t rLast, size_t cFirst, size_t cLast) const

  Return a rLast-rFirst+1 x cLast-cFirst+1 matrix.
- Matrix row (size_t r) const

  Return the r -th row.
- Matrix col (size_t c) const

  Return the c -th column.
- Matrix positive () const

  return +(∗this)
- Matrix negative () const

  return -(∗this)
- Matrix add (Matrix const &m) const

  return (∗this) + m.
- Matrix sub (Matrix const &m) const

return (∗this) - m.

- Matrix mul (Matrix const &m) const

  return (∗this) times m.
- Matrix mul (Entry const &s) const

  return (∗this) times s. s is a scalar
- Matrix div (Entry const &s) const

  return (∗this) / s. s is a scalar
- Matrix identity () const

  Return a identity matrix with size equal to itself.
- Matrix & identitied ()

  Let itself be an identity matrix.
- Matrix & diagonaled ()

  Let itself be an diagonal form of original itself.
- Matrix diagonal () const

  Return a matrix which is a diangonal form of me.
- Matrix inverse () const

  Return a matrix which is an inverse matrix of (∗this)
- Matrix & inversed ()

  let itself become itself's inverse matrix
- Matrix transpose () const

  return itself's transpose matrix
- Matrix & transposed ()

  Let itself become itself's transpose matrix.
- Matrix triangular () const

  return a matrix which is the triangular form of (∗this)
- Matrix & triangulared ()

  triangluar itself
- Matrix & operator= (Matrix const &m)

  same as copyFrom
- Entry operator() (size_t r, size_t c) const

  same as entry(r,c)
- Entry operator() (size_t r, size_t c, Entry const &e)

  same as entry(r,c,e)
- Matrix operator+ () const

  same as positive()
- Matrix operator- () const

  same as negative()
- Matrix operator+ (Matrix const &m) const

  same as add(m)
- Matrix operator- (Matrix const &m) const

  same as sub(m)
- Matrix operator∗ (Matrix const &m) const

  same as mul(m)
- Matrix operator∗ (Entry const &s) const

  same as mul(m)
- Matrix operator/ (Entry const &s) const

  same as div(s)

6.28.1 Detailed Description

template<class Entry>class meow::Matrix< Entry >

matrix

Author

cat_leopard

Definition at line 18 of file Matrix.h.

6.28.2 Member Typedef Documentation

template<class Entry> typedef std::vector<Entry>::reference meow::Matrix< Entry >::EntryRef

Definition at line 20 of file Matrix.h.

template<class Entry> typedef std::vector<Entry>::const_reference meow::Matrix< Entry >::EntryRefK

Definition at line 21 of file Matrix.h.

6.28.3 Constructor & Destructor Documentation

template<class Entry> meow::Matrix< Entry >::Matrix ( ) `[inline]`

constructor

Create an empty matrix with size **0x0**. In other world, create an **invalid** matrix

Definition at line 53 of file Matrix.h.

template<class Entry> meow::Matrix< Entry >::Matrix (Matrix< Entry > const & *m*) `[inline]`

constructor

Copy data from another one

Parameters

`in`	*m*	another matrix

Definition at line 62 of file Matrix.h.

template<class Entry> meow::Matrix< Entry >::Matrix (size_t *r*, size_t *c*, Entry const & *e*) `[inline]`

constructor

Create an *r* x *c* matrix with all entry be *e*

Parameters

`in`	*r*	number of rows
`in`	*c*	number of columns
`in`	*e*	inital entry

Definition at line 74 of file Matrix.h.

template<class Entry> meow::Matrix< Entry >::~Matrix ( ) `[inline]`

destructor

Definition at line 78 of file Matrix.h.

6.28.4 Member Function Documentation

template<class Entry> Matrix meow::Matrix< Entry >::add (Matrix< Entry > const & *m*) const `[inline]`

return (∗this) + *m*.

If the size not match, it will return an invalid matrix

Definition at line 282 of file Matrix.h.

template<class Entry> Matrix meow::Matrix< Entry >::col (size_t *c*) const `[inline]`

Return the *c* -th column.

Definition at line 260 of file Matrix.h.

template<class Entry> size_t meow::Matrix< Entry >::cols ( ) const `[inline]`

Return number of cols.

Definition at line 125 of file Matrix.h.

template<class Entry> size_t meow::Matrix< Entry >::cols (size_t *c*, Entry const & *e*) `[inline]`

resize the matrix such that number of cols become *c*

New created entry will be *e*

Parameters

`in`	*c*	new number of columns
`in`	*e*	inital entry

Returns

new number of columns

Definition at line 160 of file Matrix.h.

template<class Entry> Matrix& meow::Matrix< Entry >::copyFrom (Matrix< Entry > const & *m*) `[inline]`

copy

Copy data from another matrix

Parameters

`in`	*m*	matrix

Returns

$*$this

Definition at line 88 of file Matrix.h.

template<class Entry> Matrix meow::Matrix< Entry >::diagonal ( ) const `[inline]`

Return a matrix which is a diangonal form of me.

Definition at line 371 of file Matrix.h.

template<class Entry> Matrix& meow::Matrix< Entry >::diagonaled ( ) `[inline]`

Let itself be an diagonal form of original itself.

Definition at line 358 of file Matrix.h.

template<class Entry> Matrix meow::Matrix< Entry >::div (Entry const & *s*) const `[inline]`

return ($*$this) / *s*. *s* is a scalar

Definition at line 328 of file Matrix.h.

template<class Entry> void meow::Matrix< Entry >::entries (ssize_t *rFirst*, ssize_t *rLast*, ssize_t *cFirst*, ssize_t *cLast*, Entry const & *e*) `[inline]`

Change the entries from *rFirst* x *cFirst* to *rLast* x *cLast*.

Parameters

`in`	*rFirst*	
`in`	*rLast*	
`in`	*cFirst*	
`in`	*cLast*	
`in`	*e*	value

Returns

void

Definition at line 218 of file Matrix.h.

template<class Entry> Entry meow::Matrix< Entry >::entry (size_t *r*, size_t *c*) const `[inline]`

Access the entry at *r* x *c*.

Definition at line 193 of file Matrix.h.

template<class Entry> Entry meow::Matrix< Entry >::entry (size_t *r*, size_t *c*, Entry const & *e*) `[inline]`

Change the entry at *r* x *c*.

Definition at line 198 of file Matrix.h.

template<class Entry> EntryRef meow::Matrix< Entry >::entryGet (size_t *r*, size_t *c*) `[inline]`

Get the entry at *r* x *c*.

Definition at line 204 of file Matrix.h.

template<class Entry> Matrix& meow::Matrix< Entry >::identitied ( ) `[inline]`

Let itself be an identity matrix.

Our definition of Identity matrix is 1 for entry(i, i) and 0 otherwise.

Definition at line 348 of file Matrix.h.

template<class Entry> Matrix meow::Matrix< Entry >::identity ( ) const `[inline]`

Return a identity matrix with size equal to itself.

Definition at line 337 of file Matrix.h.

template<class Entry> Matrix meow::Matrix< Entry >::inverse ( ) const `[inline]`

Return a matrix which is an inverse matrix of (∗this)

If inverse matrix doesn't exist, it will return a invalid matrix

Definition at line 382 of file Matrix.h.

template<class Entry> Matrix& meow::Matrix< Entry >::inversed ( ) `[inline]`

let itself become itself's inverse matrix

Definition at line 410 of file Matrix.h.

template<class Entry> Matrix meow::Matrix< Entry >::mul (Matrix< Entry > const & *m*) const `[inline]`

return (∗this) times *m*.

If the size not match, it will return an invalid matrix

Definition at line 308 of file Matrix.h.

template<class Entry> Matrix meow::Matrix< Entry >::mul (Entry const & *s*) const `[inline]`

return (∗this) times *s*. *s* is a scalar
Definition at line 319 of file Matrix.h.

template<class Entry> Matrix meow::Matrix< Entry >::negative ( ) const `[inline]`

return -(∗this)
Definition at line 270 of file Matrix.h.

template<class Entry> Entry meow::Matrix< Entry >::operator() (size_t *r*, size_t *c*) const `[inline]`

same as *entry(r,c)*
Definition at line 470 of file Matrix.h.

template<class Entry> Entry meow::Matrix< Entry >::operator() (size_t *r*, size_t *c*, Entry const & *e*) `[inline]`

same as *entry(r,c,e)*
Definition at line 475 of file Matrix.h.

template<class Entry> Matrix meow::Matrix< Entry >::operator∗ (Matrix< Entry > const & *m*) const `[inline]`

same as *mul(m)*
Definition at line 500 of file Matrix.h.

template<class Entry> Matrix meow::Matrix< Entry >::operator∗ (Entry const & *s*) const `[inline]`

same as *mul(m)*
Definition at line 505 of file Matrix.h.

template<class Entry> Matrix meow::Matrix< Entry >::operator+ ( ) const `[inline]`

same as *positive()*
Definition at line 480 of file Matrix.h.

template<class Entry> Matrix meow::Matrix< Entry >::operator+ (Matrix< Entry > const & *m*) const `[inline]`

same as *add(m)*
Definition at line 490 of file Matrix.h.

template<class Entry> Matrix meow::Matrix< Entry >::operator- ( ) const `[inline]`

same as *negative()*
Definition at line 485 of file Matrix.h.

template<class Entry> Matrix meow::Matrix< Entry >::operator- (Matrix< Entry > const & *m*) const `[inline]`

same as *sub(m)*
Definition at line 495 of file Matrix.h.

template<class Entry> Matrix meow::Matrix< Entry >::operator/ (Entry const & *s*) const `[inline]`

same as *div(s)*
Definition at line 510 of file Matrix.h.

template<class Entry> Matrix& meow::Matrix< Entry >::operator= (Matrix< Entry > const & *m*) `[inline]`

same as *copyFrom*

Definition at line 465 of file Matrix.h.

template<class Entry> Matrix meow::Matrix< Entry >::positive ( ) const `[inline]`

return +(∗this)

Definition at line 265 of file Matrix.h.

template<class Entry> Matrix& meow::Matrix< Entry >::referenceFrom (Matrix< Entry > const & *m*) `[inline]`

reference

Reference itself to another matrix

Parameters

`in`	*m*	matrix

Returns

∗this

Definition at line 101 of file Matrix.h.

template<class Entry> void meow::Matrix< Entry >::reset (size_t *r*, size_t *c*, Entry const & *e*) `[inline]`

reset the size of the matrix to *r* x *c* with entry all be *e*

Definition at line 107 of file Matrix.h.

template<class Entry> Matrix meow::Matrix< Entry >::row (size_t *r*) const `[inline]`

Return the *r*-th row.

Definition at line 255 of file Matrix.h.

template<class Entry> size_t meow::Matrix< Entry >::rows ( ) const `[inline]`

Return number of rows.

Definition at line 120 of file Matrix.h.

template<class Entry> size_t meow::Matrix< Entry >::rows (size_t *r*, Entry const & *e*) `[inline]`

resize the matrix such that number of rows become *r*.

New created entry will be *e*

Parameters

`in`	*r*	new number of rows
`in`	*e*	inital entry

Returns

new number of rows

Definition at line 143 of file Matrix.h.

template<class Entry> size_t meow::Matrix< Entry >::size ( ) const `[inline]`

Return number of rows times number of cols.

Definition at line 130 of file Matrix.h.

template< class Entry > size_t meow::Matrix< Entry >::size (size_t *r*, size_t *c*, Entry const & *e*) `[inline]`

resize

Resize to *r* x *c*, with new created entry be *e*

Parameters

in	*r*	number of rows
in	*c*	number of rows
in	*e*	inital entry

Returns

$r * c$

Definition at line 186 of file Matrix.h.

template< class Entry > Matrix meow::Matrix< Entry >::sub (Matrix< Entry > const & *m*) const `[inline]`

return (∗this) - *m*.

If the size not match, it will return an invalid matrix

Definition at line 295 of file Matrix.h.

template< class Entry > Matrix meow::Matrix< Entry >::subMatrix (size_t *rFirst*, size_t *rLast*, size_t *cFirst*, size_t *cLast*) const `[inline]`

Return a *rLast-rFirst+1* x *cLast-cFirst+1* matrix.

With value be the entries from *rFirst* x *cFirst* to *rLast* x *cLast*

Parameters

in	*rFirst*	
in	*rLast*	
in	*cFirst*	
in	*cLast*	

Returns

a matrix

Definition at line 239 of file Matrix.h.

template< class Entry > Matrix meow::Matrix< Entry >::transpose ( ) const `[inline]`

return itself's transpose matrix

Definition at line 416 of file Matrix.h.

template< class Entry > Matrix& meow::Matrix< Entry >::transposed ( ) `[inline]`

Let itself become itself's transpose matrix.

Definition at line 425 of file Matrix.h.

template< class Entry > Matrix meow::Matrix< Entry >::triangular ( ) const `[inline]`

return a matrix which is the triangular form of (∗this)

Definition at line 431 of file Matrix.h.

template< class Entry > Matrix& meow::Matrix< Entry >::triangulared ( ) `[inline]`

triangluar itself

Definition at line 438 of file Matrix.h.

template<class Entry> bool meow::Matrix< Entry >::valid ( ) const `[inline]`

Return whether it is a **valid** matrix.

Definition at line 115 of file Matrix.h.

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

- meowpp/math/Matrix.h

6.29 meow::MergeableHeap< Element > Class Template Reference

`Maximum-Heap` , heap, `merge`

`#include "MergeableHeap.h"`

Public Member Functions

- MergeableHeap ()

  constructor
- MergeableHeap (MergeableHeap const &heap2)

  constructor,
- ∼MergeableHeap ()

  destructor
- MergeableHeap & copyFrom (MergeableHeap const &heap2)
- void moveTo (MergeableHeap ∗heap2)

  heap,
- Element const & top () const

  Element
- size_t size () const
- bool empty () const
- void push (Element const &value)

  element
- void pop ()

  element
- void clear ()
- void merge (MergeableHeap ∗heap2)
- MergeableHeap & operator= (MergeableHeap const &heap2)

  same as `copyFrom(heap2)`

6.29.1 Detailed Description

template<class Element>class meow::MergeableHeap< Element >

`Maximum-Heap` , heap, `merge`

Template Class Operators Request

const?	Typename	Operator	Parameters	Return Type	Description
const	Element	operator<	(Element `b`)	bool	

Note

: MergeableHeap `A B`, :

- `A.merge(&B) B`
- `B.moveTo(&A) B,A`

Author

cat_leopard

Definition at line 30 of file MergeableHeap.h.

6.29.2 Constructor & Destructor Documentation

template<class Element > meow::MergeableHeap< Element >::MergeableHeap ( ) `[inline]`

constructor

Definition at line 78 of file MergeableHeap.h.

template<class Element > meow::MergeableHeap< Element >::MergeableHeap (MergeableHeap< Element > const & *heap2*) `[inline]`

constructor,

Definition at line 82 of file MergeableHeap.h.

template<class Element > meow::MergeableHeap< Element >::~MergeableHeap ( ) `[inline]`

destructor

Definition at line 86 of file MergeableHeap.h.

6.29.3 Member Function Documentation

template<class Element > void meow::MergeableHeap< Element >::clear ( ) `[inline]`

Definition at line 147 of file MergeableHeap.h.

template<class Element > MergeableHeap& meow::MergeableHeap< Element >::copyFrom (MergeableHeap< Element > const & *heap2*) `[inline]`

Definition at line 91 of file MergeableHeap.h.

template<class Element > bool meow::MergeableHeap< Element >::empty ( ) const `[inline]`

Definition at line 123 of file MergeableHeap.h.

template<class Element > void meow::MergeableHeap< Element >::merge (MergeableHeap< Element > ∗ *heap2*) `[inline]`

MergeableHeapheap

Definition at line 155 of file MergeableHeap.h.

template<class Element > void meow::MergeableHeap< Element >::moveTo (MergeableHeap< Element > ∗ *heap2*) `[inline]`

heap,

Definition at line 100 of file MergeableHeap.h.

template<class Element > MergeableHeap& meow::MergeableHeap< Element >::operator= (MergeableHeap< Element > const & *heap2*) `[inline]`

same as `copyFrom(heap2)`

Definition at line 161 of file MergeableHeap.h.

template<class Element > void meow::MergeableHeap< Element >::pop ( ) `[inline]`

element

Definition at line 137 of file MergeableHeap.h.

template<class Element > void meow::MergeableHeap< Element >::push (Element const & *value*) [inline]

element
Definition at line 130 of file MergeableHeap.h.

template<class Element > size_t meow::MergeableHeap< Element >::size ( ) const [inline]

Definition at line 116 of file MergeableHeap.h.

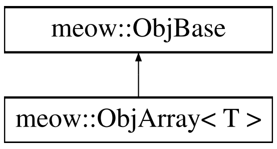
template<class Element > Element const& meow::MergeableHeap< Element >::top ( ) const [inline]

Element
Definition at line 109 of file MergeableHeap.h.
The documentation for this class was generated from the following file:

- meowpp/dsa/MergeableHeap.h

6.30 meow::ObjArray< T > Class Template Reference

`std::vector`, ObjBase
`#include "ObjArray.h"`
Inheritance diagram for meow::ObjArray< T >:



Public Member Functions

- ObjArray ()
- ObjArray (ObjArray const &a)
- ObjArray (std::vector< T > const &a)
- ObjArray (size_t sz, T const &e)
- ∼ObjArray ()
- ObjArray & copyFrom (ObjArray const &a)
- ObjArray & referenceFrom (ObjArray const &a)
- size_t size () const
- bool empty () const
- size_t size (size_t res, T const &i)
- size_t size (size_t res)
- void clear ()
- T const & entry (size_t i) const
- T const & entry (size_t i, T const &e)
- T const & putBack (T const &e)
- bool popBack ()
- ObjArray & operator= (ObjArray const &a)
- T const & operator[] (size_t i) const
- T & operator[] (size_t i)
- bool write (FILE ∗f, bool bin, unsigned int fg) const

  , implement `false`
- bool read (FILE ∗f, bool bin, unsigned int fg)

  , implement `false`

- ObjBase ∗ create () const

  new, implement `NULL`
- ObjBase ∗ copyFrom (ObjBase const ∗b)

  , operator=
- char const ∗ ctype () const

  C-style stringclasstype name
- std::string type () const

  std::stringclasstype name

Additional Inherited Members

6.30.1 Detailed Description

template<class T>class meow::ObjArray< T >

`std::vector`, ObjBase

Author

cathook

Definition at line 23 of file ObjArray.h.

6.30.2 Constructor & Destructor Documentation

template<class T > meow::ObjArray< T >::ObjArray ( ) `[inline]`

Definition at line 38 of file ObjArray.h.

template<class T > meow::ObjArray< T >::ObjArray (ObjArray< T > const & *a*) `[inline]`

Definition at line 41 of file ObjArray.h.

template<class T > meow::ObjArray< T >::ObjArray (std::vector< T > const & *a*) `[inline]`

Definition at line 45 of file ObjArray.h.

template<class T > meow::ObjArray< T >::ObjArray (size_t *sz*, T const & *e*) `[inline]`

Definition at line 49 of file ObjArray.h.

template<class T > meow::ObjArray< T >::∼ObjArray ( ) `[inline]`

Definition at line 53 of file ObjArray.h.

6.30.3 Member Function Documentation

template<class T > void meow::ObjArray< T >::clear ( ) `[inline]`

Definition at line 83 of file ObjArray.h.

template<class T > ObjArray& meow::ObjArray< T >::copyFrom (ObjArray< T > const & *a*) `[inline]`

Definition at line 56 of file ObjArray.h.

template<class T > ObjBase∗ meow::ObjArray< T >::copyFrom (ObjBase const ∗ *b*) `[inline]`, `[virtual]`

, operator=

Parameters

in	*b*	

Returns

`this`

Reimplemented from meow::ObjBase.

Definition at line 151 of file ObjArray.h.

template<class T > ObjBase* meow::ObjArray< T >::create ( ) const `[inline]`, `[virtual]`

new, implement `NULL`

Reimplemented from meow::ObjBase.

Definition at line 147 of file ObjArray.h.

template<class T > char const* meow::ObjArray< T >::ctype ( ) const `[inline]`, `[virtual]`

C-style stringclasstype name

Reimplemented from meow::ObjBase.

Definition at line 155 of file ObjArray.h.

template<class T > bool meow::ObjArray< T >::empty ( ) const `[inline]`

Definition at line 69 of file ObjArray.h.

template<class T > T const& meow::ObjArray< T >::entry (size_t *i*) const `[inline]`

Definition at line 87 of file ObjArray.h.

template<class T > T const& meow::ObjArray< T >::entry (size_t *i*, T const & *e*) `[inline]`

Definition at line 90 of file ObjArray.h.

template<class T > ObjArray& meow::ObjArray< T >::operator= (ObjArray< T > const & *a*) `[inline]`

Definition at line 106 of file ObjArray.h.

template<class T > T const& meow::ObjArray< T >::operator[] (size_t *i*) const `[inline]`

Definition at line 110 of file ObjArray.h.

template<class T > T& meow::ObjArray< T >::operator[] (size_t *i*) `[inline]`

Definition at line 114 of file ObjArray.h.

template<class T > bool meow::ObjArray< T >::popBack ( ) `[inline]`

Definition at line 100 of file ObjArray.h.

template<class T > T const& meow::ObjArray< T >::putBack (T const & *e*) `[inline]`

Definition at line 95 of file ObjArray.h.

template<class T > bool meow::ObjArray< T >::read (FILE * *f*, bool *bin*, unsigned int *fg*) `[inline]`, `[virtual]`

, implement `false`

Parameters

in	*f*	
in	*bin*	binary
in	*fg*	argument

Returns

Reimplemented from meow::ObjBase.

Definition at line 132 of file ObjArray.h.

template<class T > ObjArray& meow::ObjArray< T >::referenceFrom (ObjArray< T > const & *a*) `[inline]`

Definition at line 61 of file ObjArray.h.

template<class T > size_t meow::ObjArray< T >::size ( ) const `[inline]`

Definition at line 66 of file ObjArray.h.

template<class T > size_t meow::ObjArray< T >::size (size_t *res,* T const & *i*) `[inline]`

Definition at line 73 of file ObjArray.h.

template<class T > size_t meow::ObjArray< T >::size (size_t *res*) `[inline]`

Definition at line 78 of file ObjArray.h.

template<class T > std::string meow::ObjArray< T >::type ( ) const `[inline]`, `[virtual]`

std::stringclasstype name

Reimplemented from meow::ObjBase.

Definition at line 159 of file ObjArray.h.

template<class T > bool meow::ObjArray< T >::write (FILE ∗ *f,* bool *bin,* unsigned int *fg*) const `[inline]`, `[virtual]`

, implement `false`

Parameters

in	*f*	
in	*bin*	binary
in	*fg*	argument

Returns

Reimplemented from meow::ObjBase.

Definition at line 118 of file ObjArray.h.

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

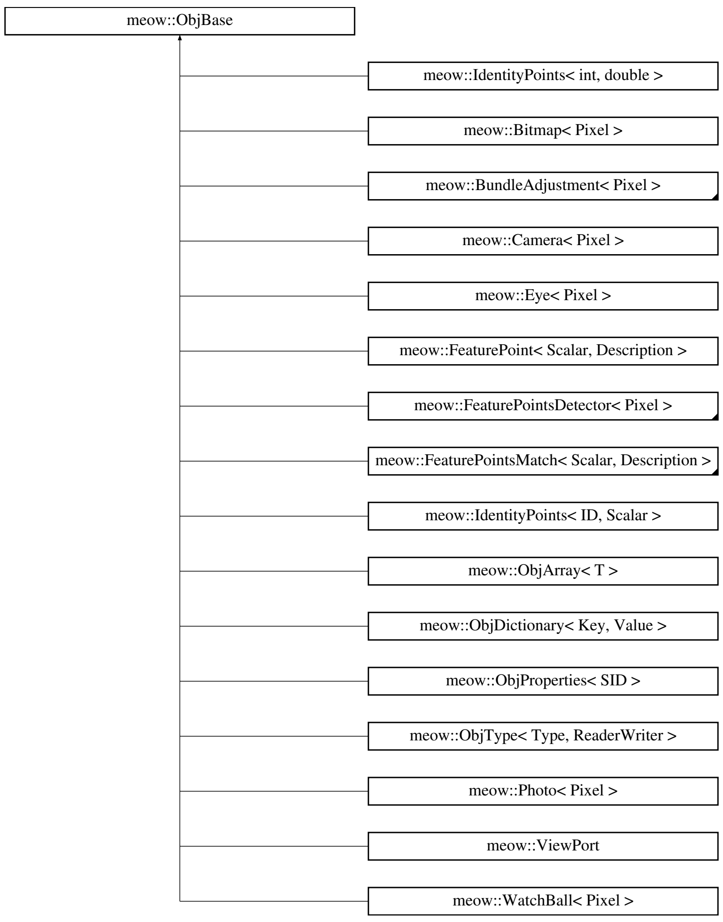
- meowpp/oo/ObjArray.h

6.31 meow::ObjBase Class Reference

Base, read, write, create, ...

`#include "ObjBase.h"`

Inheritance diagram for meow::ObjBase:



meow::ObjBase
meow::IdentityPoints< int, double >
meow::Bitmap< Pixel >
meow::BundleAdjustment< Pixel >
meow::Camera< Pixel >
meow::Eye< Pixel >
meow::FeaturePoint< Scalar, Description >
meow::FeaturePointsDetector< Pixel >
meow::FeaturePointsMatch< Scalar, Description >
meow::IdentityPoints< ID, Scalar >
meow::ObjArray< T >
meow::ObjDictionary< Key, Value >
meow::ObjProperties< SID >
meow::ObjType< Type, ReaderWriter >
meow::Photo< Pixel >
meow::ViewPort
meow::WatchBall< Pixel >

Public Member Functions

- virtual ∼ObjBase ()
- virtual bool write (FILE ∗f, bool bin, unsigned int fg) const

    , implement `false`
- virtual bool read (FILE ∗f, bool bin, unsigned int fg)

    , implement `false`
- virtual ObjBase ∗ create () const

    new, implement `NULL`
- virtual ObjBase ∗ copyFrom (ObjBase const ∗b)

    , operator=
- virtual char const ∗ ctype () const

    C-style stringclasstype name
- virtual std::string type () const

    std::stringclasstype name

Static Public Member Functions

- static char const ∗ ctypeBase ()

    C-style stringbasetype name

- static std::string typeBase ()

  std::stringbasetype name

Protected Member Functions

- ObjBase ()

6.31.1 Detailed Description

Base, read, write, create, ...

Author

cathook

Definition at line 15 of file ObjBase.h.

6.31.2 Constructor & Destructor Documentation

meow::ObjBase::ObjBase ( ) `[inline]`, `[protected]`

Definition at line 17 of file ObjBase.h.

virtual meow::ObjBase::∼ObjBase ( ) `[inline]`, `[virtual]`

Definition at line 19 of file ObjBase.h.

6.31.3 Member Function Documentation

virtual ObjBase∗ meow::ObjBase::copyFrom (ObjBase const ∗ *b*) `[inline]`, `[virtual]`

, operator=

Parameters

`in`	*b*	

Returns

`this`

Reimplemented in meow::Photo< Pixel >, meow::Bitmap< Pixel >, meow::BundleAdjustment_LM< Pixel >, meow::FeaturePointsDetector_Harris< Pixel >, meow::IdentityPoints< ID, Scalar >, meow::IdentityPoints< int, double >, meow::Camera< Pixel >, meow::WatchBall< Pixel >, meow::FeaturePointsMatch_K_Match< Scalar, Description >, meow::Eye< Pixel >, meow::ObjArray< T >, meow::ObjDictionary< Key, Value >, meow::ObjType< Type, ReaderWriter >, and meow::ObjProperties< SID >.

Definition at line 58 of file ObjBase.h.

virtual ObjBase∗ meow::ObjBase::create ( ) const `[inline]`, `[virtual]`

new, implement `NULL`

Reimplemented in meow::Photo< Pixel >, meow::Bitmap< Pixel >, meow::BundleAdjustment_LM< Pixel >, meow::FeaturePointsDetector_Harris< Pixel >, meow::IdentityPoints< ID, Scalar >, meow::IdentityPoints< int, double >, meow::Camera< Pixel >, meow::WatchBall< Pixel >, meow::FeaturePoint< Scalar, Description >, meow::FeaturePointsMatch_K_Match< Scalar, Description >, meow::ObjArray< T >, meow::Eye< Pixel >, meow::ObjDictionary< Key, Value >, meow::ObjType< Type, ReaderWriter >, and meow::ObjProperties< SID >.

Definition at line 48 of file ObjBase.h.

virtual char const∗ meow::ObjBase::ctype ( ) const [inline], [virtual]

C-style stringclasstype name

Reimplemented in meow::Photo< Pixel >, meow::Bitmap< Pixel >, meow::BundleAdjustment_LM< Pixel >, meow::FeaturePointsDetector_Harris< Pixel >, meow::IdentityPoints< ID, Scalar >, meow::IdentityPoints< int, double >, meow::Camera< Pixel >, meow::WatchBall< Pixel >, meow::FeaturePoint< Scalar, Description >, meow::FeaturePointsMatch_K_Match< Scalar, Description >, meow::Eye< Pixel >, meow::ObjArray< T >, meow::ObjDictionary< Key, Value >, meow::ObjType< Type, ReaderWriter >, and meow::ObjProperties< SID >.

Definition at line 66 of file ObjBase.h.

static char const∗ meow::ObjBase::ctypeBase ( ) [inline], [static]

C-style stringbasetype name

Definition at line 81 of file ObjBase.h.

virtual bool meow::ObjBase::read (FILE ∗ *f*, bool *bin*, unsigned int *fg*) [inline], [virtual]

, implement `false`

Parameters

`in`	*f*	
`in`	*bin*	binary
`in`	*fg*	argument

Returns

Reimplemented in meow::Photo< Pixel >, meow::Bitmap< Pixel >, meow::FeaturePointsDetector_Harris< Pixel >, meow::IdentityPoints< ID, Scalar >, meow::IdentityPoints< int, double >, meow::WatchBall< Pixel >, meow::Camera< Pixel >, meow::FeaturePoint< Scalar, Description >, meow::FeaturePointsMatch_K_Match< Scalar, Description >, meow::ObjArray< T >, meow::ObjDictionary< Key, Value >, meow::Eye< Pixel >, meow::ObjType< Type, ReaderWriter >, and meow::ObjProperties< SID >.

Definition at line 41 of file ObjBase.h.

virtual std::string meow::ObjBase::type ( ) const [inline], [virtual]

std::stringclasstype name

Reimplemented in meow::Photo< Pixel >, meow::Bitmap< Pixel >, meow::BundleAdjustment_LM< Pixel >, meow::FeaturePointsDetector_Harris< Pixel >, meow::IdentityPoints< ID, Scalar >, meow::IdentityPoints< int, double >, meow::Camera< Pixel >, meow::WatchBall< Pixel >, meow::FeaturePoint< Scalar, Description >, meow::FeaturePointsMatch_K_Match< Scalar, Description >, meow::Eye< Pixel >, meow::ObjArray< T >, meow::ObjDictionary< Key, Value >, meow::ObjType< Type, ReaderWriter >, and meow::ObjProperties< SID >.

Definition at line 73 of file ObjBase.h.

static std::string meow::ObjBase::typeBase ( ) [inline], [static]

std::stringbasetype name

Definition at line 88 of file ObjBase.h.

virtual bool meow::ObjBase::write (FILE ∗ *f*, bool *bin*, unsigned int *fg*) const [inline], [virtual]

, implement `false`

Parameters

`in`	*f*	

in	*bin*	binary
in	*fg*	argument

Returns

Reimplemented in meow::Photo< Pixel >, meow::Bitmap< Pixel >, meow::BundleAdjustment_LM< Pixel >, meow::FeaturePointsDetector_Harris< Pixel >, meow::IdentityPoints< ID, Scalar >, meow::IdentityPoints< int, double >, meow::WatchBall< Pixel >, meow::Camera< Pixel >, meow::FeaturePoint< Scalar, Description >, meow::FeaturePointsMatch_K_Match< Scalar, Description >, meow::ObjArray< T >, meow::ObjDictionary< Key, Value >, meow::Eye< Pixel >, meow::ObjType< Type, ReaderWriter >, and meow::ObjProperties< SID >.

Definition at line 29 of file ObjBase.h.

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

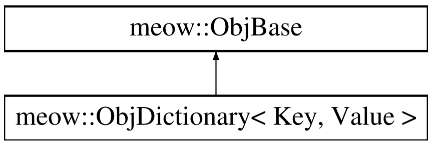
- meowpp/oo/ObjBase.h

6.32 meow::ObjDictionary< Key, Value > Class Template Reference

`std::map`, ObjBase

`#include "ObjDictionary.h"`

Inheritance diagram for meow::ObjDictionary< Key, Value >:



Public Member Functions

- ObjDictionary ()
- ObjDictionary (ObjDictionary const &d)
- ObjDictionary (std::map< Key, Value > const &d)
- ∼ObjDictionary ()
- ObjDictionary & copyFrom (ObjDictionary const &d)
- ObjDictionary & referenceFrom (ObjDictionary const &d)
- size_t size () const
- bool empty () const
- void clear ()
- std::map< Key, Value >
  ::const_iterator end () const
- std::map< Key, Value >::iterator end ()
- std::map< Key, Value >
  ::const_iterator find (Key const &k) const
- std::map< Key, Value >::iterator find (Key const &k)
- bool exist (Key const &k) const
- void insert (Key const &k, Value const &v)
- ObjDictionary & operator= (ObjDictionary const &a)
- Value & operator[] (Key const &k)
- bool write (FILE ∗f, bool bin, unsigned int fg) const

    , implement `false`
- bool read (FILE ∗f, bool bin, unsigned int fg)

    , implement `false`
- ObjBase ∗ create () const

    new, implement `NULL`

- ObjBase ∗ copyFrom (ObjBase const ∗b)

  , operator=
- char const ∗ ctype () const

  C-style stringclasstype name
- std::string type () const

  std::stringclasstype name

Additional Inherited Members

6.32.1 Detailed Description

template<class Key, class Value>class meow::ObjDictionary< Key, Value >

`std::map`, ObjBase

Author

cathook

Definition at line 23 of file ObjDictionary.h.

6.32.2 Constructor & Destructor Documentation

template<class Key , class Value > meow::ObjDictionary< Key, Value >::ObjDictionary ( ) `[inline]`

Definition at line 38 of file ObjDictionary.h.

template<class Key , class Value > meow::ObjDictionary< Key, Value >::ObjDictionary (ObjDictionary< Key, Value > const & *d*) `[inline]`

Definition at line 41 of file ObjDictionary.h.

template<class Key , class Value > meow::ObjDictionary< Key, Value >::ObjDictionary (std::map< Key, Value > const & *d*) `[inline]`

Definition at line 45 of file ObjDictionary.h.

template<class Key , class Value > meow::ObjDictionary< Key, Value >::∼ObjDictionary ( ) `[inline]`

Definition at line 49 of file ObjDictionary.h.

6.32.3 Member Function Documentation

template<class Key , class Value > void meow::ObjDictionary< Key, Value >::clear ( ) `[inline]`

Definition at line 69 of file ObjDictionary.h.

template<class Key , class Value > ObjDictionary& meow::ObjDictionary< Key, Value >::copyFrom (ObjDictionary< Key, Value > const & *d*) `[inline]`

Definition at line 52 of file ObjDictionary.h.

template<class Key , class Value > ObjBase∗ meow::ObjDictionary< Key, Value >::copyFrom (ObjBase const ∗ *b*) `[inline]`, `[virtual]`

, operator=

Parameters

in	*b*	

Returns

`this`

Reimplemented from meow::ObjBase.
Definition at line 143 of file ObjDictionary.h.

template<class Key , class Value > ObjBase∗ meow::ObjDictionary< Key, Value >::create ( ) const [inline], [virtual]

new, implement `NULL`
Reimplemented from meow::ObjBase.
Definition at line 139 of file ObjDictionary.h.

template<class Key , class Value > char const∗ meow::ObjDictionary< Key, Value >::ctype ( ) const [inline], [virtual]

C-style stringclasstype name
Reimplemented from meow::ObjBase.
Definition at line 147 of file ObjDictionary.h.

template<class Key , class Value > bool meow::ObjDictionary< Key, Value >::empty ( ) const [inline]

Definition at line 65 of file ObjDictionary.h.

template<class Key , class Value > std::map<Key, Value>::const_iterator meow::ObjDictionary< Key, Value >::end ( ) const [inline]

Definition at line 73 of file ObjDictionary.h.

template<class Key , class Value > std::map<Key, Value>::iterator meow::ObjDictionary< Key, Value >::end ( ) [inline]

Definition at line 77 of file ObjDictionary.h.

template<class Key , class Value > bool meow::ObjDictionary< Key, Value >::exist (Key const & *k*) const [inline]

Definition at line 89 of file ObjDictionary.h.

template<class Key , class Value > std::map<Key, Value>::const_iterator meow::ObjDictionary< Key, Value >::find (Key const & *k*) const [inline]

Definition at line 81 of file ObjDictionary.h.

template<class Key , class Value > std::map<Key, Value>::iterator meow::ObjDictionary< Key, Value >::find (Key const & *k*) [inline]

Definition at line 85 of file ObjDictionary.h.

template<class Key , class Value > void meow::ObjDictionary< Key, Value >::insert (Key const & *k*, Value const & *v*) [inline]

Definition at line 93 of file ObjDictionary.h.

template<class Key , class Value > ObjDictionary& meow::ObjDictionary< Key, Value >::operator= (ObjDictionary< Key, Value > const & *a*) `[inline]`

Definition at line 97 of file ObjDictionary.h.

template<class Key , class Value > Value& meow::ObjDictionary< Key, Value >::operator[] (Key const & *k*) `[inline]`

Definition at line 101 of file ObjDictionary.h.

template<class Key , class Value > bool meow::ObjDictionary< Key, Value >::read (FILE * *f*, bool *bin*, unsigned int *fg*) `[inline]`, `[virtual]`

, implement `false`

Parameters

`in`	*f*	
`in`	*bin*	binary
`in`	*fg*	argument

Returns

Reimplemented from meow::ObjBase.

Definition at line 121 of file ObjDictionary.h.

template<class Key , class Value > ObjDictionary& meow::ObjDictionary< Key, Value >::referenceFrom (ObjDictionary< Key, Value > const & *d*) `[inline]`

Definition at line 57 of file ObjDictionary.h.

template<class Key , class Value > size_t meow::ObjDictionary< Key, Value >::size ( ) const `[inline]`

Definition at line 62 of file ObjDictionary.h.

template<class Key , class Value > std::string meow::ObjDictionary< Key, Value >::type ( ) const `[inline]`, `[virtual]`

std::stringclasstype name

Reimplemented from meow::ObjBase.

Definition at line 151 of file ObjDictionary.h.

template<class Key , class Value > bool meow::ObjDictionary< Key, Value >::write (FILE * *f*, bool *bin*, unsigned int *fg*) const `[inline]`, `[virtual]`

, implement `false`

Parameters

`in`	*f*	
`in`	*bin*	binary
`in`	*fg*	argument

Returns

Reimplemented from meow::ObjBase.

Definition at line 105 of file ObjDictionary.h.

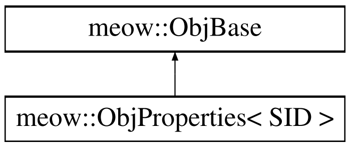
The documentation for this class was generated from the following file:

- meowpp/oo/ObjDictionary.h

6.33 meow::ObjProperties< SID > Class Template Reference

`#include "ObjProperties.h"`

Inheritance diagram for meow::ObjProperties< SID >:



Public Member Functions

- ObjProperties ()
- ObjProperties (ObjProperties const &p)
- virtual ∼ObjProperties ()
- size_t propertySize () const
- bool propertyEmpty () const
- void propertyClear ()
- ObjBase const ∗ property (std::string name) const
- ObjBase ∗ property (std::string name)
- bool propertyAdd (std::string name, ObjBase ∗obj, bool autoRemove)
- bool propertyDel (std::string name)
- ObjProperties & properties () const
- ObjProperties & properties (ObjProperties const &p)
- bool write (FILE ∗f, bool bin, unsigned int fg) const

    , implement `false`
- bool read (FILE ∗f, bool bin, unsigned int fg)

    , implement `false`
- ObjBase ∗ create () const

    new, implement `NULL`
- ObjBase ∗ copyFrom (ObjBase const ∗b)

    , operator=
- char const ∗ ctype () const

    C-style stringclasstype name
- std::string type () const

    std::stringclasstype name

Additional Inherited Members

6.33.1 Detailed Description

template<size_t SID>class meow::ObjProperties< SID >

Definition at line 13 of file ObjProperties.h.

6.33.2 Constructor & Destructor Documentation

template<size_t SID> meow::ObjProperties< SID >::ObjProperties ( )

template<size_t SID> meow::ObjProperties< SID >::ObjProperties (ObjProperties< SID > const & *p*)

template<size_t SID> virtual meow::ObjProperties< SID >::∼ObjProperties ( ) `[virtual]`

6.33.3 Member Function Documentation

template<size_t SID> ObjBase∗ meow::ObjProperties< SID >::copyFrom (ObjBase const ∗ *b*) `[virtual]`

, operator=

Parameters

`in`	*b*	

Returns

`this`

Reimplemented from meow::ObjBase.

template<size_t SID> ObjBase∗ meow::ObjProperties< SID >::create ( ) const `[virtual]`

new, implement `NULL`

Reimplemented from meow::ObjBase.

template<size_t SID> char const∗ meow::ObjProperties< SID >::ctype ( ) const `[virtual]`

C-style stringclasstype name

Reimplemented from meow::ObjBase.

template<size_t SID> ObjProperties& meow::ObjProperties< SID >::properties ( ) const

template<size_t SID> ObjProperties& meow::ObjProperties< SID >::properties (ObjProperties< SID > const & *p*)

template<size_t SID> ObjBase const∗ meow::ObjProperties< SID >::property (std::string *name*) const

template<size_t SID> ObjBase∗ meow::ObjProperties< SID >::property (std::string *name*)

template<size_t SID> bool meow::ObjProperties< SID >::propertyAdd (std::string *name,* ObjBase ∗ *obj,* bool *autoRemove*)

template<size_t SID> void meow::ObjProperties< SID >::propertyClear ( )

template<size_t SID> bool meow::ObjProperties< SID >::propertyDel (std::string *name*)

template<size_t SID> bool meow::ObjProperties< SID >::propertyEmpty ( ) const

template<size_t SID> size_t meow::ObjProperties< SID >::propertySize ( ) const

template<size_t SID> bool meow::ObjProperties< SID >::read (FILE ∗ *f,* bool *bin,* unsigned int *fg*) `[virtual]`

, implement `false`

Parameters

`in`	*f*	
`in`	*bin*	binary
`in`	*fg*	argument

Returns

Reimplemented from meow::ObjBase.

template<size_t SID> std::string meow::ObjProperties< SID >::type ( ) const `[virtual]`

std::stringclasstype name

Reimplemented from meow::ObjBase.

template<size_t SID> bool meow::ObjProperties< SID >::write (FILE ∗ *f,* bool *bin,* unsigned int *fg*) const `[virtual]`

, implement `false`

Parameters

in	*f*	
in	*bin*	binary
in	*fg*	argument

Returns

Reimplemented from meow::ObjBase.
The documentation for this class was generated from the following file:

- meowpp/oo/ObjProperties.h

6.34 meow::ObjSelector< id > Class Template Reference

register, runtimestringnewclass
```
#include "ObjSelector.h"
```

Public Member Functions

- ObjSelector (std::string name, ObjBase ∗obj, bool autoDelete)

    ObjSelector, ObjBase
- ObjSelector (ObjBase ∗obj, bool autoDelete)

    ObjSelector, ObjBase
- ∼ObjSelector ()

Static Public Member Functions

- static void add (std::string name, ObjBase ∗obj, bool autoDelete)

    () Class (`ObjBase`) Name
- static void add (ObjBase ∗obj, bool autoDelete)

    () Class (`ObjBase`) typename
- static void del (std::string name)

    nameClass
- static ObjBase const ∗ get (std::string name)

    Class
- static ObjBase ∗ create (std::string name)

    Class new
- static bool exist (ObjBase ∗obj)

    typeClass
- static std::string name (ObjBase ∗obj)

    typename
- static std::vector< std::string > names ()

    name
- static bool write (FILE ∗f, bool binary, ObjBase ∗obj, unsigned int fg)

    ()
- static ObjBase ∗ read (FILE ∗f, bool binary)

    ()

Friends

- struct Info

6.34.1 Detailed Description

template<size_t id>class meow::ObjSelector< id >

register, runtimestringnewclass

Author

cathook< `ObjSelector`

Definition at line 22 of file ObjSelector.h.

6.34.2 Constructor & Destructor Documentation

template<size_t id> meow::ObjSelector< id >::ObjSelector (std::string *name,* ObjBase ∗ *obj,* bool *autoDelete*) `[inline]`

ObjSelector, ObjBase
Definition at line 148 of file ObjSelector.h.

template<size_t id> meow::ObjSelector< id >::ObjSelector (ObjBase ∗ *obj,* bool *autoDelete*) `[inline]`

ObjSelector, ObjBase
Definition at line 156 of file ObjSelector.h.

template<size_t id> meow::ObjSelector< id >::∼ObjSelector ( ) `[inline]`

Definition at line 162 of file ObjSelector.h.

6.34.3 Member Function Documentation

template<size_t id> static void meow::ObjSelector< id >::add (std::string *name,* ObjBase ∗ *obj,* bool *autoDelete*) `[inline]`, `[static]`

() Class (`ObjBase`) Name
Definition at line 69 of file ObjSelector.h.

template<size_t id> static void meow::ObjSelector< id >::add (ObjBase ∗ *obj,* bool *autoDelete*) `[inline]`, `[static]`

() Class (`ObjBase`) typename
Definition at line 76 of file ObjSelector.h.

template<size_t id> static ObjBase∗ meow::ObjSelector< id >::create (std::string *name*) `[inline]`, `[static]`

Class new
Definition at line 101 of file ObjSelector.h.

template<size_t id> static void meow::ObjSelector< id >::del (std::string *name*) `[inline]`, `[static]`

nameClass
Definition at line 83 of file ObjSelector.h.

template<size_t id> static bool meow::ObjSelector< id >::exist (ObjBase ∗ *obj*) `[inline]`, `[static]`

typeClass
Definition at line 110 of file ObjSelector.h.

template< size_t id > static ObjBase const∗ meow::ObjSelector< id >::get (std::string *name*) [inline], [static]

Class

Definition at line 93 of file ObjSelector.h.

template< size_t id > static std::string meow::ObjSelector< id >::name (ObjBase ∗ *obj*) [inline], [static]

typename

Definition at line 124 of file ObjSelector.h.

template< size_t id > static std::vector< std::string > meow::ObjSelector< id >::names ( ) [inline], [static]

name

Definition at line 138 of file ObjSelector.h.

template< size_t id > static ObjBase∗ meow::ObjSelector< id >::read (FILE ∗ *f*, bool *binary*) [inline], [static]

()

Definition at line 188 of file ObjSelector.h.

template< size_t id > static bool meow::ObjSelector< id >::write (FILE ∗ *f*, bool *binary*, ObjBase ∗ *obj*, unsigned int *fg*) [inline], [static]

()

Definition at line 171 of file ObjSelector.h.

6.34.4 Friends And Related Function Documentation

template< size_t id > friend struct Info [friend]

Definition at line 45 of file ObjSelector.h.

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

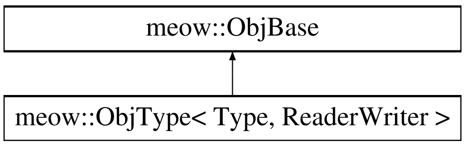
- meowpp/oo/ObjSelector.h

6.35 meow::ObjType< Type, ReaderWriter > Class Template Reference

`Type` , ObjBase

`#include "ObjTypes.h"`

Inheritance diagram for meow::ObjType< Type, ReaderWriter >:



Public Member Functions

- ObjType ()

  constructor
- ObyType (Type const &t)

  constructor,
- ObjType (ObjType const &a)

constructor, copy

- ∼ObjType ()
- ObjType & copyFrom (ObjType const &a)
- ObjType & referenceFrom (ObjType const &a)
- Type const & access () const
- Type & modify ()
- ObjType & operator= (ObjType const &a)
- Type const & operator() () const
- Type & operator() ()
- bool write (FILE ∗f, bool bin, unsigned int fg) const

  , implement `false`
- bool read (FILE ∗f, bool bin, unsigned int fg)

  , implement `false`
- ObjBase ∗ create () const

  new, implement `NULL`
- ObjBase ∗ copyFrom (ObjBase const ∗b)

  , operator=
- char const ∗ ctype () const

  C-style stringclasstype name
- std::string type () const

  std::stringclasstype name

Additional Inherited Members

6.35.1 Detailed Description

template<class Type, class ReaderWriter>class meow::ObjType< Type, ReaderWriter >

`Type` , ObjBase

Author

cathook

Definition at line 18 of file ObjTypes.h.

6.35.2 Constructor & Destructor Documentation

template<class Type , class ReaderWriter > meow::ObjType< Type, ReaderWriter >::ObjType ( ) `[inline]`

constructor

Definition at line 35 of file ObjTypes.h.

template<class Type , class ReaderWriter > meow::ObjType< Type, ReaderWriter >::ObjType (ObjType< Type, ReaderWriter > const & *a*) `[inline]`

constructor, copy

Definition at line 43 of file ObjTypes.h.

template<class Type , class ReaderWriter > meow::ObjType< Type, ReaderWriter >::∼ObjType ( ) `[inline]`

Definition at line 46 of file ObjTypes.h.

6.35.3 Member Function Documentation

template< class Type , class ReaderWriter > Type const& meow::ObjType< Type, ReaderWriter >::access ( ) const `[inline]`

Definition at line 59 of file ObjTypes.h.

template< class Type , class ReaderWriter > ObjType& meow::ObjType< Type, ReaderWriter >::copyFrom (ObjType< Type, ReaderWriter > const & *a*) `[inline]`

Definition at line 49 of file ObjTypes.h.

template< class Type , class ReaderWriter > ObjBase∗ meow::ObjType< Type, ReaderWriter >::copyFrom (ObjBase const ∗ *b*) `[inline]`, `[virtual]`

, operator=

Parameters

`in`	*b*	

Returns

`this`

Reimplemented from meow::ObjBase.
Definition at line 91 of file ObjTypes.h.

template< class Type , class ReaderWriter > ObjBase∗ meow::ObjType< Type, ReaderWriter >::create ( ) const `[inline]`, `[virtual]`

new, implement `NULL`
Reimplemented from meow::ObjBase.
Definition at line 87 of file ObjTypes.h.

template< class Type , class ReaderWriter > char const∗ meow::ObjType< Type, ReaderWriter >::ctype ( ) const `[inline]`, `[virtual]`

C-style stringclasstype name
Reimplemented from meow::ObjBase.
Definition at line 95 of file ObjTypes.h.

template< class Type , class ReaderWriter > Type& meow::ObjType< Type, ReaderWriter >::modify ( ) `[inline]`

Definition at line 63 of file ObjTypes.h.

template< class Type , class ReaderWriter > meow::ObjType< Type, ReaderWriter >::ObyType (Type const & *t*) `[inline]`

constructor,
Definition at line 39 of file ObjTypes.h.

template< class Type , class ReaderWriter > Type const& meow::ObjType< Type, ReaderWriter >::operator() ( ) const `[inline]`

Definition at line 71 of file ObjTypes.h.

template< class Type , class ReaderWriter > Type& meow::ObjType< Type, ReaderWriter >::operator() ( ) `[inline]`

Definition at line 75 of file ObjTypes.h.

template<class Type , class ReaderWriter > ObjType& meow::ObjType< Type, ReaderWriter >::operator= (ObjType< Type, ReaderWriter > const & *a*) `[inline]`

Definition at line 67 of file ObjTypes.h.

template<class Type , class ReaderWriter > bool meow::ObjType< Type, ReaderWriter >::read (FILE ∗ *f*, bool *bin*, unsigned int *fg*) `[inline]`, `[virtual]`

, implement `false`

Parameters

`in`	*f*	
`in`	*bin*	binary
`in`	*fg*	argument

Returns

Reimplemented from meow::ObjBase.

Definition at line 83 of file ObjTypes.h.

template<class Type , class ReaderWriter > ObjType& meow::ObjType< Type, ReaderWriter >::referenceFrom (ObjType< Type, ReaderWriter > const & *a*) `[inline]`

Definition at line 54 of file ObjTypes.h.

template<class Type , class ReaderWriter > std::string meow::ObjType< Type, ReaderWriter >::type ( ) const `[inline]`, `[virtual]`

std::stringclasstype name

Reimplemented from meow::ObjBase.

Definition at line 99 of file ObjTypes.h.

template<class Type , class ReaderWriter > bool meow::ObjType< Type, ReaderWriter >::write (FILE ∗ *f*, bool *bin*, unsigned int *fg*) const `[inline]`, `[virtual]`

, implement `false`

Parameters

`in`	*f*	
`in`	*bin*	binary
`in`	*fg*	argument

Returns

Reimplemented from meow::ObjBase.

Definition at line 79 of file ObjTypes.h.

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

- meowpp/oo/ObjTypes.h

6.36 meow::PairToPair< F1, F2, T1, T2 > Struct Template Reference

.from.first, .from.second, .to.first, .to.second

```
#include "utility.h"
```

Public Member Functions

- PairToPair ()
- PairToPair (PairToPair const &pp)
- PairToPair (F1 const &f1, F2 const &f2, T1 const &t1, T2 const &t2)
- bool operator== (PairToPair const &p) const

Public Attributes

- std::pair< F1, F2 > from
- std::pair< T1, T2 > to

6.36.1 Detailed Description

template<class F1, class F2 = F1, class T1 = F1, class T2 = T1>struct meow::PairToPair< F1, F2, T1, T2 >

.from.first, .from.second, .to.first, .to.second

Author

cathook

Definition at line 19 of file utility.h.

6.36.2 Constructor & Destructor Documentation

template<class F1 , class F2 = F1, class T1 = F1, class T2 = T1> meow::PairToPair< F1, F2, T1, T2 >::PairToPair ( ) `[inline]`

Definition at line 23 of file utility.h.

template<class F1 , class F2 = F1, class T1 = F1, class T2 = T1> meow::PairToPair< F1, F2, T1, T2 >::PairToPair (PairToPair< F1, F2, T1, T2 > const & *pp*) `[inline]`

Definition at line 25 of file utility.h.

template<class F1 , class F2 = F1, class T1 = F1, class T2 = T1> meow::PairToPair< F1, F2, T1, T2 >::PairToPair (F1 const & *f1,* F2 const & *f2,* T1 const & *t1,* T2 const & *t2*) `[inline]`

Definition at line 27 of file utility.h.

6.36.3 Member Function Documentation

template<class F1 , class F2 = F1, class T1 = F1, class T2 = T1> bool meow::PairToPair< F1, F2, T1, T2 >::operator== (PairToPair< F1, F2, T1, T2 > const & *p*) const `[inline]`

Definition at line 30 of file utility.h.

6.36.4 Member Data Documentation

template<class F1 , class F2 = F1, class T1 = F1, class T2 = T1> std::pair<F1, F2> meow::PairToPair< F1, F2, T1, T2 >::from

Definition at line 20 of file utility.h.

template<class F1 , class F2 = F1, class T1 = F1, class T2 = T1> std::pair<T1, T2> meow::PairToPair< F1, F2, T1, T2 >::to

Definition at line 21 of file utility.h.

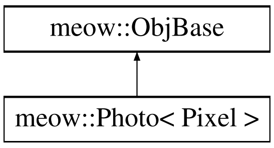
The documentation for this struct was generated from the following file:

- meowpp/utility.h

6.37 meow::Photo< Pixel > Class Template Reference

`#include "Photo.h"`

Inheritance diagram for meow::Photo< Pixel >:



Public Member Functions

- Photo ()
  constructor
- Photo (Photo const &b)
  constructor
- Photo (Bitmap< Pixel > const &bmp)
  constructor
- Photo (Bitmap< Pixel > const &bmp, double f)
  constructor
- Photo (Bitmap< Pixel > const &bmp, double f, Vector2D< double > const &c)
  constructor
- ∼Photo ()
  destructor
- Photo & copyFrom (Photo const &b)
- Photo & referneceFrom (Photo const &b)
- void reset (Bitmap< Pixel > const &bmp)
  bitmap, focal
- void reset (Bitmap< Pixel > const &bmp, double f)
  bitmap, focal
- void reset (Bitmap< Pixel > const &bmp, double f, Vector2D< double > const &c)
  bitmap, focal, center
- Bitmap< Pixel > const & bitmap () const
  `bitmap`
- Bitmap< Pixel > & bitmapGet ()
  `bitmap` (constant)
- Bitmap< Pixel > const & bitmap (Bitmap< Pixel > const &bmp)
  bitmap
- double focal () const
  focal length
- double focal (double f)
  focal length
- PhotoProjection< double > projection () const
  photo projection
- PhotoProjection< double > projection (PhotoProjection< double > const &p)
  photo projection
- Vector2D< double > const & center () const
- Vector2D< double > & centerGet ()

  (non-constant reference)
- Vector2D< double > const & center (Vector2D< double > const &c)

- size_t width () const

  bitmap
- size_t height () const

  bitmap
- Pixel pixel (size_t y, size_t x) const

  bitmappixel
- Pixel pixel (size_t y, size_t x, Pixel const &p)

  pixel
- bool inside (Vector2D< double > const &yx) const

- bool inside (Vector3D< double > const &p) const

- Pixel color (Vector2D< double > const &yx) const

- Pixel color (Vector3D< double > const &p) const

- Photo & operator= (Photo const &b)

  same as .copyFrom(b)
- bool write (FILE *f, bool bin, unsigned int fg) const

- bool read (FILE *f, bool bin, unsigned int fg)

- ObjBase * create () const

  new
- ObjBase * copyFrom (ObjBase const *b)

- char const * ctype () const

  classtype
- std::string type () const

  classtype

Additional Inherited Members

6.37.1 Detailed Description

template<class Pixel>class meow::Photo< Pixel >

`Bitmap focal`

Author

cat_leopard

Definition at line 31 of file Photo.h.

6.37.2 Constructor & Destructor Documentation

template<class Pixel> meow::Photo< Pixel >::Photo ( ) `[inline]`

constructor

focal 1

Definition at line 59 of file Photo.h.

template<class Pixel> meow::Photo< Pixel >::Photo (Photo< Pixel > const & *b*) `[inline]`

constructor

Parameters

`in`	*b*	

Definition at line 70 of file Photo.h.

template<class Pixel> meow::Photo< Pixel >::Photo (Bitmap< Pixel > const & *bmp*) `[inline]`

constructor

,
Parameters

`in`	*bmp*	

Definition at line 80 of file Photo.h.

template<class Pixel> meow::Photo< Pixel >::Photo (Bitmap< Pixel > const & *bmp*, double *f*) `[inline]`

constructor
Parameters

`in`	*bmp*	
`in`	*f*	

Definition at line 92 of file Photo.h.

template<class Pixel> meow::Photo< Pixel >::Photo (Bitmap< Pixel > const & *bmp*, double *f*, Vector2D< double > const & *c*) `[inline]`

constructor

,
Parameters

`in`	*bmp*	
`in`	*f*	
`in`	*c*	

Definition at line 105 of file Photo.h.

template<class Pixel> meow::Photo< Pixel >::~Photo ( ) `[inline]`

destructor
Definition at line 112 of file Photo.h.

6.37.3 Member Function Documentation

template<class Pixel> Bitmap<Pixel> const& meow::Photo< Pixel >::bitmap ( ) const `[inline]`

`bitmap`
Definition at line 178 of file Photo.h.

template<class Pixel> Bitmap<Pixel> const& meow::Photo< Pixel >::bitmap (Bitmap< Pixel > const & *bmp*) `[inline]`

bitmap
Parameters

`in`	*bmp*	bitmap

Returns

`bitmap`

Definition at line 195 of file Photo.h.

template<class Pixel> Bitmap<Pixel>& meow::Photo< Pixel >::bitmapGet ( ) [inline]

`bitmap` (constant)

Definition at line 185 of file Photo.h.

template<class Pixel> Vector2D<double> const& meow::Photo< Pixel >::center ( ) const [inline]

Returns

vector

Definition at line 240 of file Photo.h.

template<class Pixel> Vector2D<double> const& meow::Photo< Pixel >::center (Vector2D< double > const & *c*) [inline]

Parameters

in	*c*	

Returns

Definition at line 260 of file Photo.h.

template<class Pixel> Vector2D<double>& meow::Photo< Pixel >::centerGet ( ) [inline]

(non-constant reference)

Returns

vector

Definition at line 249 of file Photo.h.

template<class Pixel> Pixel meow::Photo< Pixel >::color (Vector2D< double > const & *yx*) const [inline]

vector,

Parameters

in	*yx*	(center)

Returns

pixel

Definition at line 329 of file Photo.h.

template<class Pixel> Pixel meow::Photo< Pixel >::color (Vector3D< double > const & *p*) const [inline]

Parameters

in	*p*	p

Returns

pixel

Definition at line 354 of file Photo.h.

template<class Pixel> Photo& meow::Photo< Pixel >::copyFrom (Photo< Pixel > const & *b*) [inline]

Parameters

`in`	*b*	

Definition at line 120 of file Photo.h.

template<class Pixel> ObjBase∗ meow::Photo< Pixel >::copyFrom (ObjBase const ∗ *b*) `[inline]`, `[virtual]`

`ObjBase const* Bitmap`. method`copyFrom`

Parameters

`in`	*b*	

Returns

this

Reimplemented from meow::ObjBase.
Definition at line 420 of file Photo.h.

template<class Pixel> ObjBase∗ meow::Photo< Pixel >::create ( ) const `[inline]`, `[virtual]`

new

Returns

newPhoto<Pixel>

Reimplemented from meow::ObjBase.
Definition at line 407 of file Photo.h.

template<class Pixel> char const∗ meow::Photo< Pixel >::ctype ( ) const `[inline]`, `[virtual]`

classtype

Returns

`char const* typename`

Reimplemented from meow::ObjBase.
Definition at line 428 of file Photo.h.

template<class Pixel> double meow::Photo< Pixel >::focal ( ) const `[inline]`

focal length
Definition at line 203 of file Photo.h.

template<class Pixel> double meow::Photo< Pixel >::focal (double *f*) `[inline]`

focal length

Parameters

`in`	*f*	focal length

Returns

`focal` length

Definition at line 213 of file Photo.h.

template<class Pixel> size_t meow::Photo< Pixel >::height ( ) const `[inline]`

bitmap
Definition at line 275 of file Photo.h.

template<class Pixel> bool meow::Photo< Pixel >::inside (Vector2D< double > const & *yx*) const `[inline]`

Parameters

in	*yx*	

Returns

`true/false`

Definition at line 301 of file Photo.h.

template<class Pixel> bool meow::Photo< Pixel >::inside (Vector3D< double > const & *p*) const `[inline]`

Parameters

in	*p*	

Returns

`true/false`

Definition at line 315 of file Photo.h.

template<class Pixel> Photo& meow::Photo< Pixel >::operator= (Photo< Pixel > const & *b*) `[inline]`

same as .copyFrom(b)

Definition at line 361 of file Photo.h.

template<class Pixel> Pixel meow::Photo< Pixel >::pixel (size_t *y*, size_t *x*) const `[inline]`

bitmappixel

Definition at line 282 of file Photo.h.

template<class Pixel> Pixel meow::Photo< Pixel >::pixel (size_t *y*, size_t *x*, Pixel const & *p*) `[inline]`

pixel

Definition at line 289 of file Photo.h.

template<class Pixel> PhotoProjection<double> meow::Photo< Pixel >::projection ( ) const `[inline]`

photo projection

Definition at line 221 of file Photo.h.

template<class Pixel> PhotoProjection<double> meow::Photo< Pixel >::projection (PhotoProjection< double > const & *p*) `[inline]`

photo projection

Definition at line 228 of file Photo.h.

template<class Pixel> bool meow::Photo< Pixel >::read (FILE ∗ *f*, bool *bin*, unsigned int *fg*) `[inline]`, `[virtual]`

Note

Reimplemented from meow::ObjBase.

Definition at line 388 of file Photo.h.

template<class Pixel> Photo& meow::Photo< Pixel >::referneceFrom (Photo< Pixel > const & *b*) `[inline]`

Parameters

in	*b*	

Definition at line 130 of file Photo.h.

template< class Pixel > void meow::Photo< Pixel >::reset (Bitmap< Pixel > const & *bmp*) [inline]

bitmap, focal
focal, centerbitmap

Parameters

in	*bmp*	bitmap

Definition at line 142 of file Photo.h.

template< class Pixel > void meow::Photo< Pixel >::reset (Bitmap< Pixel > const & *bmp*, double *f*) [inline]

bitmap, focal
centerbitmap

Parameters

in	*bmp*	bitmap
in	*f*	focal

Definition at line 156 of file Photo.h.

template< class Pixel > void meow::Photo< Pixel >::reset (Bitmap< Pixel > const & *bmp*, double *f*, Vector2D< double > const & *c*) [inline]

bitmap, focal, center

Parameters

in	*bmp*	bitmap
in	*f*	focal
in	*c*	

Definition at line 169 of file Photo.h.

template< class Pixel > std::string meow::Photo< Pixel >::type ( ) const [inline], [virtual]

classtype

Returns

`std::string` typename

Reimplemented from meow::ObjBase.
Definition at line 436 of file Photo.h.

template< class Pixel > size_t meow::Photo< Pixel >::width ( ) const [inline]

bitmap
Definition at line 268 of file Photo.h.

template< class Pixel > bool meow::Photo< Pixel >::write (FILE ∗ *f*, bool *bin*, unsigned int *fg*) const [inline], [virtual]

Note

Reimplemented from meow::ObjBase.
Definition at line 369 of file Photo.h.
The documentation for this class was generated from the following file:

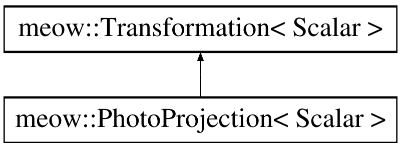
- meowpp/gra/Photo.h

6.38 meow::PhotoProjection< Scalar > Class Template Reference

A **photo projection** is a kind of transformation that project point/vector to a flat **photo**.

`#include "Transformations.h"`

Inheritance diagram for meow::PhotoProjection< Scalar >:


meow::Transformation< Scalar >
meow::PhotoProjection< Scalar >

Public Member Functions

- PhotoProjection (size_t dimension)
- PhotoProjection (size_t dimension, Scalar const &f)
- PhotoProjection (PhotoProjection const &p)
- PhotoProjection & copyFrom (PhotoProjection const &b)
- PhotoProjection & referenceFrom (PhotoProjection const &b)
- Scalar parameter (size_t i) const

  Same as `focal()`
- Scalar parameter (size_t i, Scalar const &s)

  Same as `focal(s)`
- Scalar focal () const

  Get the focal length.
- Scalar focal (Scalar const &f)

  Set the focal length.
- size_t dimension () const

  Get the dimension of this projection.
- Matrix< Scalar > transformate (Matrix< Scalar > const &x) const

  Project the input vector(s) onto the plane.
- Matrix< Scalar > jacobian (Matrix< Scalar > const &x) const

  Return the jacobian matrix (derivate by the input vector) of this projection.
- Matrix< Scalar > jacobian (Matrix< Scalar > const &x, size_t i) const

  Return the jacobian matrix (derivate by the focus length) of this projection.
- PhotoProjection & operator= (PhotoProjection const &b)

  Same as `copyFrom(b)`
- Matrix< Scalar > operator() (Matrix< Scalar > const &v) const

  Same as `transformate(v)`

Additional Inherited Members

6.38.1 Detailed Description

template<class Scalar>class meow::PhotoProjection< Scalar >

A **photo projection** is a kind of transformation that project point/vector to a flat **photo**.

Assume:

- The dimension of a photo projection is N
- The length of the input vector is L
- The focal length is f

Then transformation is like below:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ . \\ . \\ . \\ x_N \end{bmatrix} \overset{transformate}{\rightarrow} \begin{bmatrix} \frac{-x_1 \times f}{x_N} \\ \frac{-x_2 \times f}{x_N} \\ \frac{-x_3 \times f}{x_N} \\ . \\ . \\ . \\ -f \end{bmatrix}$$

i.e. projecte the vector onto the plane $x_N = -f$.

Author

cat_leopard

Definition at line 323 of file Transformations.h.

6.38.2 Constructor & Destructor Documentation

template<class Scalar> meow::PhotoProjection< Scalar >::PhotoProjection (size_t *dimension*) `[inline]`

Constructor, focal = 1

Definition at line 347 of file Transformations.h.

template<class Scalar> meow::PhotoProjection< Scalar >::PhotoProjection (size_t *dimension,* Scalar const & *f*) `[inline]`

Constructor

Definition at line 355 of file Transformations.h.

template<class Scalar> meow::PhotoProjection< Scalar >::PhotoProjection (PhotoProjection< Scalar > const & *p*) `[inline]`

Constructor, copy settings from another PhotoProjection.

Definition at line 363 of file Transformations.h.

6.38.3 Member Function Documentation

template<class Scalar> PhotoProjection& meow::PhotoProjection< Scalar >::copyFrom (PhotoProjection< Scalar > const & *b*) `[inline]`

Copy settings from another one

Parameters

`in`	*b*	another one

Returns

`*this`

Definition at line 372 of file Transformations.h.

template<class Scalar> size_t meow::PhotoProjection< Scalar >::dimension ( ) const `[inline]`

Get the dimension of this projection.

Definition at line 425 of file Transformations.h.

template<class Scalar> Scalar meow::PhotoProjection< Scalar >::focal ( ) const `[inline]`

Get the focal length.

Returns

Focal length

Definition at line 407 of file Transformations.h.

template<class Scalar> Scalar meow::PhotoProjection< Scalar >::focal (Scalar const & *f*) `[inline]`

Set the focal length.

Parameters

`in`	*f*	New focal length

Returns

New focal length

Definition at line 417 of file Transformations.h.

template<class Scalar> Matrix<Scalar> meow::PhotoProjection< Scalar >::jacobian (Matrix< Scalar > const & *x*) const `[inline], [virtual]`

Return the jacobian matrix (derivate by the input vector) of this projection.

This method only allow a vector-like matrix be input. Assume:

- The dimension of this projection is N
- The length of the input vector is $L = \sqrt{x_1^2 + x_2^2 + ... + x_N^2}$
- The focal length of this projection is f

Then the jacobian matrix is like below:

$$f \times \begin{bmatrix} \frac{-1}{x_N} & 0 & 0 & ... & \frac{1}{x_N^2} \\ 0 & \frac{-1}{x_N} & 0 & ... & \frac{1}{x_N^2} \\ 0 & 0 & \frac{-1}{x_N} & ... & \frac{1}{x_N^2} \\ . & . & . & & . \\ . & . & . & & . \\ . & . & . & & . \\ 0 & 0 & 0 & ... & 0 \end{bmatrix}$$

Parameters

`in`	*x*	The input matrix.

Returns

The output matrix.

Reimplemented from meow::Transformation< Scalar >.

Definition at line 485 of file Transformations.h.

template<class Scalar> Matrix<Scalar> meow::PhotoProjection< Scalar >::jacobian (Matrix< Scalar > const & *x*, size_t *i*) const `[inline], [virtual]`

Return the jacobian matrix (derivate by the focus length) of this projection.

This method only allow a vector-like matrix be input. Assume:

- The dimension of this projection is N

- The length of the input vector is $L = \sqrt{x_1^2 + x_2^2 + ... + x_N^2}$
- The focal length of this projection is f

Then the jacobian matrix is like below:

$$\begin{bmatrix} \frac{-x_1}{x_N} \\ \frac{-x_2}{x_N} \\ \frac{-x_3}{x_N} \\ . \\ . \\ . \\ -1 \end{bmatrix}$$

Parameters

in	*x*	The input matrix.
in	*i*	Useless parameter

Returns

The output matrix.

Reimplemented from meow::Transformation< Scalar >.
Definition at line 523 of file Transformations.h.

template<class Scalar> Matrix<Scalar> meow::PhotoProjection< Scalar >::operator() (Matrix< Scalar > const & *v*) const [inline]

Same as `transformate(v)`
Definition at line 541 of file Transformations.h.

template<class Scalar> PhotoProjection& meow::PhotoProjection< Scalar >::operator= (PhotoProjection< Scalar > const & *b*) [inline]

Same as `copyFrom(b)`
Definition at line 534 of file Transformations.h.

template<class Scalar> Scalar meow::PhotoProjection< Scalar >::parameter (size_t *i*) const [inline], [virtual]

Same as `focal()`
Implements meow::Transformation< Scalar >.
Definition at line 392 of file Transformations.h.

template<class Scalar> Scalar meow::PhotoProjection< Scalar >::parameter (size_t *i*, Scalar const & *s*) [inline], [virtual]

Same as `focal(s)`
Implements meow::Transformation< Scalar >.
Definition at line 399 of file Transformations.h.

template<class Scalar> PhotoProjection& meow::PhotoProjection< Scalar >::referenceFrom (PhotoProjection< Scalar > const & *b*) [inline]

Reference settings from another one

Parameters

`in`	*b*	another one

Returns

`*this`

Definition at line 383 of file Transformations.h.

template<class Scalar> Matrix<Scalar> meow::PhotoProjection< Scalar >::transformate (Matrix< Scalar > const & *x*) const `[inline], [virtual]`

Project the input vector(s) onto the plane.

The equation of the plane is $x_N = -f$, where the N is the dimension of this projection and f is the focal length.

If the number of columns of the input matrix is larger than 1, this method will think that you want to transform multiple vector once and the number of columns of the output matrix will be the same of the number of columns of the input one.

Parameters

`in`	*x*	The input matrix.

Returns

The output matrix.

Note

Take into account that too much safty checking will lead to inefficient, this method will not checking whether the dimension of the input vector/matrix is right. So be sure the data is valid before you call this method.

Implements meow::Transformation< Scalar >.

Definition at line 446 of file Transformations.h.

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

- meowpp/math/Transformations.h

6.39 meow::ReaderWriter_double Class Reference

`#include "ObjTypes.h"`

Static Public Member Functions

- static bool write (FILE *f, bool bin, unsigned int fg, double const &k)
- static bool read (FILE *f, bool bin, unsigned int fg, double *k)

6.39.1 Detailed Description

Definition at line 144 of file ObjTypes.h.

6.39.2 Member Function Documentation

static bool meow::ReaderWriter_double::read (FILE * *f*, bool *bin*, unsigned int *fg*, double * *k*) `[inline], [static]`

Definition at line 154 of file ObjTypes.h.

static bool meow::ReaderWriter_double::write (FILE ∗ *f*, bool *bin*, unsigned int *fg*, double const & *k*) [inline], [static]

Definition at line 146 of file ObjTypes.h.

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

- meowpp/oo/ObjTypes.h

6.40 meow::ReaderWriter_int Class Reference

```
#include "ObjTypes.h"
```

Static Public Member Functions

- static bool write (FILE ∗f, bool bin, unsigned int fg, int const &k)
- static bool read (FILE ∗f, bool bin, unsigned int fg, int ∗k)

6.40.1 Detailed Description

Definition at line 104 of file ObjTypes.h.

6.40.2 Member Function Documentation

static bool meow::ReaderWriter_int::read (FILE ∗ *f*, bool *bin*, unsigned int *fg*, int ∗ *k*) [inline], [static]

Definition at line 114 of file ObjTypes.h.

static bool meow::ReaderWriter_int::write (FILE ∗ *f*, bool *bin*, unsigned int *fg*, int const & *k*) [inline], [static]

Definition at line 106 of file ObjTypes.h.

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

- meowpp/oo/ObjTypes.h

6.41 meow::ReaderWriter_size_t Class Reference

```
#include "ObjTypes.h"
```

Static Public Member Functions

- static bool write (FILE ∗f, bool bin, unsigned int fg, size_t const &k)
- static bool read (FILE ∗f, bool bin, unsigned int fg, size_t ∗k)

6.41.1 Detailed Description

Definition at line 124 of file ObjTypes.h.

6.41.2 Member Function Documentation

static bool meow::ReaderWriter_size_t::read (FILE ∗ *f*, bool *bin*, unsigned int *fg*, size_t ∗ *k*) [inline], [static]

Definition at line 134 of file ObjTypes.h.

static bool meow::ReaderWriter_size_t::write (FILE ∗ *f*, bool *bin*, unsigned int *fg*, size_t const & *k*) `[inline]`, `[static]`

Definition at line 126 of file ObjTypes.h.

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

- meowpp/oo/ObjTypes.h

6.42 meow::ReaderWriter_string Class Reference

`#include "ObjTypes.h"`

Static Public Member Functions

- static bool write (FILE ∗f, bool bin, unsigned int fg, std::string const &k)
- static bool read (FILE ∗f, bool bin, unsigned int fg, std::string ∗k)

6.42.1 Detailed Description

Definition at line 164 of file ObjTypes.h.

6.42.2 Member Function Documentation

static bool meow::ReaderWriter_string::read (FILE ∗ *f*, bool *bin*, unsigned int *fg*, std::string ∗ *k*) `[inline]`, `[static]`

Definition at line 178 of file ObjTypes.h.

static bool meow::ReaderWriter_string::write (FILE ∗ *f*, bool *bin*, unsigned int *fg*, std::string const & *k*) `[inline]`, `[static]`

Definition at line 166 of file ObjTypes.h.

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

- meowpp/oo/ObjTypes.h

6.43 meow::RegisterInterface< T > Class Template Reference

`#include "Register_Implement.h"`

Public Member Functions

- virtual bool regImplement (ImplementInterface< T > ∗imp)
- virtual ImplementInterface< T > ∗ getImplement (T const &identify)
- virtual ∼RegisterInterface ()

Protected Member Functions

- RegisterInterface ()

6.43.1 Detailed Description

template<class T>class meow::RegisterInterface< T >

Definition at line 17 of file Register_Implement.h.

6.43.2 Constructor & Destructor Documentation

template<class T > meow::RegisterInterface< T >::RegisterInterface ( ) `[inline], [protected]`

Definition at line 5 of file Register_Implement.hpp.

template<class T > virtual meow::RegisterInterface< T >::~RegisterInterface ( ) `[inline], [virtual]`

Definition at line 25 of file Register_Implement.h.

6.43.3 Member Function Documentation

template<class T > ImplementInterface< T > ∗ meow::RegisterInterface< T >::getImplement (T const & *identify*) `[inline], [virtual]`

Definition at line 16 of file Register_Implement.hpp.

template<class T > bool meow::RegisterInterface< T >::regImplement (ImplementInterface< T > ∗ *imp*) `[inline], [virtual]`

Definition at line 7 of file Register_Implement.hpp.

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

- meowpp/oo/Register_Implement.h
- meowpp/oo/Register_Implement.hpp

6.44 meow::RGB< T > Class Template Reference

`#include "RGB.h"`

Public Member Functions

- virtual ~RGB ()
- virtual T rMax () const =0
- virtual T rMin () const =0
- virtual T gMax () const =0
- virtual T gMin () const =0
- virtual T bMax () const =0
- virtual T bMin () const =0
- T r () const
- T g () const
- T b () const
- T rgb (size_t i) const
- T bgr (size_t i) const
- T r (T const &val)
- T g (T const &val)
- T b (T const &val)
- T rgb (size_t i, T const &val)
- T bgr (size_t i, T const &val)

Protected Member Functions

- RGB ()
- RGB (T const &r, T const &g, T const &b)
- RGB (T const ∗rgb)

Protected Attributes

- T rgb_ [3]

6.44.1 Detailed Description

template<class T>class meow::RGB< T >

Definition at line 5 of file RGB.h.

6.44.2 Constructor & Destructor Documentation

template<class T > meow::RGB< T >::RGB ( ) `[inline], [protected]`

Definition at line 5 of file RGB.hpp.

template<class T> meow::RGB< T >::RGB (T const & *r*, T const & *g*, T const & *b*) `[inline], [protected]`

Definition at line 6 of file RGB.hpp.

template<class T> meow::RGB< T >::RGB (T const * *rgb*) `[inline], [protected]`

Definition at line 9 of file RGB.hpp.

template<class T> virtual meow::RGB< T >::~RGB ( ) `[inline], [virtual]`

Definition at line 12 of file RGB.h.

6.44.3 Member Function Documentation

template<class T > T meow::RGB< T >::b ( ) const `[inline]`

Definition at line 16 of file RGB.hpp.

template<class T> T meow::RGB< T >::b (T const & *val*) `[inline]`

Definition at line 24 of file RGB.hpp.

template<class T > T meow::RGB< T >::bgr (size_t *i*) const `[inline]`

Definition at line 20 of file RGB.hpp.

template<class T> T meow::RGB< T >::bgr (size_t *i*, T const & *val*) `[inline]`

Definition at line 29 of file RGB.hpp.

template<class T> virtual T meow::RGB< T >::bMax ( ) const `[pure virtual]`

Implemented in meow::RGBi, and meow::RGBf.

template<class T> virtual T meow::RGB< T >::bMin ( ) const `[pure virtual]`

Implemented in meow::RGBi, and meow::RGBf.

template<class T > T meow::RGB< T >::g ( ) const `[inline]`

Definition at line 15 of file RGB.hpp.

template<class T> T meow::RGB< T >::g (T const & *val*) `[inline]`

Definition at line 23 of file RGB.hpp.

template<class T> virtual T meow::RGB< T >::gMax ( ) const `[pure virtual]`

Implemented in meow::RGBi, and meow::RGBf.

template<class T> virtual T meow::RGB< T >::gMin ( ) const `[pure virtual]`

Implemented in meow::RGBi, and meow::RGBf.

template<class T > T meow::RGB< T >::r ( ) const `[inline]`

Definition at line 14 of file RGB.hpp.

template<class T> T meow::RGB< T >::r (T const & *val*) `[inline]`

Definition at line 22 of file RGB.hpp.

template<class T > T meow::RGB< T >::rgb (size_t *i*) const `[inline]`

Definition at line 17 of file RGB.hpp.

template<class T> T meow::RGB< T >::rgb (size_t *i*, T const & *val*) `[inline]`

Definition at line 25 of file RGB.hpp.

template<class T> virtual T meow::RGB< T >::rMax ( ) const `[pure virtual]`

Implemented in meow::RGBi, and meow::RGBf.

template<class T> virtual T meow::RGB< T >::rMin ( ) const `[pure virtual]`

Implemented in meow::RGBi, and meow::RGBf.

6.44.4 Member Data Documentation

template<class T> T meow::RGB< T >::rgb_[3] `[protected]`

Definition at line 7 of file RGB.h.

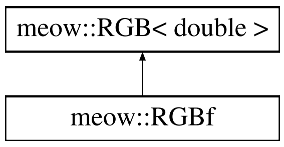
The documentation for this class was generated from the following files:

- meowpp/colors/RGB.h
- meowpp/colors/RGB.hpp

6.45 meow::RGBf Class Reference

`#include "RGB.h"`

Inheritance diagram for meow::RGBf:



Public Member Functions

- RGBf ()
- RGBf (double const &r, double const &g, double const &b)
- RGBf (double const ∗rgb)
- ∼RGBf ()
- double rMin () const
- double rMax () const
- double gMin () const
- double gMax () const
- double bMin () const
- double bMax () const

Additional Inherited Members

6.45.1 Detailed Description

Definition at line 34 of file RGB.h.

6.45.2 Constructor & Destructor Documentation

meow::RGBf::RGBf ( ) `[inline]`

Definition at line 35 of file RGB.hpp.

meow::RGBf::RGBf (double const & *r*, double const & *g*, double const & *b*) `[inline]`

Definition at line 37 of file RGB.hpp.

meow::RGBf::RGBf (double const ∗ *rgb*) `[inline]`

Definition at line 38 of file RGB.hpp.

meow::RGBf::∼RGBf ( ) `[inline]`

Definition at line 36 of file RGB.hpp.

6.45.3 Member Function Documentation

double meow::RGBf::bMax ( ) const `[inline]`, `[virtual]`

Implements meow::RGB< double >.
Definition at line 44 of file RGB.hpp.

double meow::RGBf::bMin ( ) const `[inline]`, `[virtual]`

Implements meow::RGB< double >.
Definition at line 43 of file RGB.hpp.

double meow::RGBf::gMax ( ) const `[inline]`, `[virtual]`

Implements meow::RGB< double >.
Definition at line 42 of file RGB.hpp.

double meow::RGBf::gMin ( ) const `[inline]`, `[virtual]`

Implements meow::RGB< double >.
Definition at line 41 of file RGB.hpp.

double meow::RGBf::rMax ( ) const [inline], [virtual]

Implements meow::RGB< double >.

Definition at line 40 of file RGB.hpp.

double meow::RGBf::rMin ( ) const [inline], [virtual]

Implements meow::RGB< double >.

Definition at line 39 of file RGB.hpp.

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

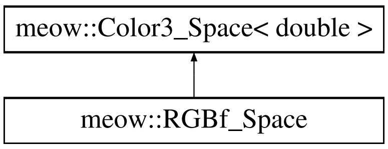
- meowpp/colors/RGB.h
- meowpp/colors/RGB.hpp

6.46 meow::RGBf_Space Class Reference

Red, **Green**, **Blue**

`#include "RGB_Space.h"`

Inheritance diagram for meow::RGBf_Space:



Public Member Functions

- RGBf_Space ()
- RGBf_Space (double c)
- RGBf_Space (Vector3D< double > const &v)
- RGBf_Space (RGBf_Space const &b)
- ∼RGBf_Space ()
- double const & rgbMin (size_t i) const
- double const & rMin () const
- double const & gMin () const
- double const & bMin () const
- double const & rgbMax (size_t i) const
- double const & rMax () const
- double const & gMax () const
- double const & bMax () const
- double const & rgb (size_t i) const
- double const & r () const
- double const & g () const
- double const & b () const
- double const & rgb (size_t i, double c)
- double const & r (double c)
- double const & g (double c)
- double const & b (double c)
- double & rgbGet (size_t i)
- double & rGet ()
- double & gGet ()
- double & bGet ()
- RGBf_Space & operator= (RGBf_Space const &b)
- RGBf_Space operator+ (RGBf_Space const &b) const
- RGBf_Space operator- (RGBf_Space const &b) const

- RGBf_Space operator∗ (double const &c) const
- RGBf_Space operator/ (double const &c) const
- double operator∗ (RGBf_Space const &b) const

Additional Inherited Members

6.46.1 Detailed Description

Red, **Green**, **Blue**
0.0∼1.0

Author

cat_leopard

Definition at line 86 of file RGB_Space.h.

6.46.2 Constructor & Destructor Documentation

meow::RGBf_Space::RGBf_Space ( ) [inline]

Definition at line 88 of file RGB_Space.h.

meow::RGBf_Space::RGBf_Space (double *c*) [inline]

Definition at line 92 of file RGB_Space.h.

meow::RGBf_Space::RGBf_Space (Vector3D< double > const & *v*) [inline]

Definition at line 96 of file RGB_Space.h.

meow::RGBf_Space::RGBf_Space (RGBf_Space const & *b*) [inline]

Definition at line 101 of file RGB_Space.h.

meow::RGBf_Space::∼RGBf_Space ( ) [inline]

Definition at line 103 of file RGB_Space.h.

6.46.3 Member Function Documentation

double const& meow::RGBf_Space::b ( ) const [inline]

Definition at line 116 of file RGB_Space.h.

double const& meow::RGBf_Space::b (double *c*) [inline]

Definition at line 120 of file RGB_Space.h.

double& meow::RGBf_Space::bGet ( ) [inline]

Definition at line 124 of file RGB_Space.h.

double const& meow::RGBf_Space::bMax ( ) const [inline]

Definition at line 112 of file RGB_Space.h.

double const& meow::RGBf_Space::bMin ( ) const [inline]

Definition at line 108 of file RGB_Space.h.

double const& meow::RGBf_Space::g ( ) const [inline]

Definition at line 115 of file RGB_Space.h.

double const& meow::RGBf_Space::g (double *c*) [inline]

Definition at line 119 of file RGB_Space.h.

double& meow::RGBf_Space::gGet ( ) [inline]

Definition at line 123 of file RGB_Space.h.

double const& meow::RGBf_Space::gMax ( ) const [inline]

Definition at line 111 of file RGB_Space.h.

double const& meow::RGBf_Space::gMin ( ) const [inline]

Definition at line 107 of file RGB_Space.h.

RGBf_Space meow::RGBf_Space::operator∗ (double const & *c*) const [inline]

Definition at line 135 of file RGB_Space.h.

double meow::RGBf_Space::operator∗ (RGBf_Space const & *b*) const [inline]

Definition at line 141 of file RGB_Space.h.

RGBf_Space meow::RGBf_Space::operator+ (RGBf_Space const & *b*) const [inline]

Definition at line 129 of file RGB_Space.h.

RGBf_Space meow::RGBf_Space::operator- (RGBf_Space const & *b*) const [inline]

Definition at line 132 of file RGB_Space.h.

RGBf_Space meow::RGBf_Space::operator/ (double const & *c*) const [inline]

Definition at line 138 of file RGB_Space.h.

RGBf_Space& meow::RGBf_Space::operator= (RGBf_Space const & *b*) [inline]

Definition at line 125 of file RGB_Space.h.

double const& meow::RGBf_Space::r ( ) const [inline]

Definition at line 114 of file RGB_Space.h.

double const& meow::RGBf_Space::r (double *c*) [inline]

Definition at line 118 of file RGB_Space.h.

double const& meow::RGBf_Space::rgb (size_t *i*) const [inline]

Definition at line 113 of file RGB_Space.h.

double const& meow::RGBf_Space::rgb (size_t *i*, double *c*) [inline]

Definition at line 117 of file RGB_Space.h.

double& meow::RGBf_Space::rgbGet (size_t *i*) `[inline]`

Definition at line 121 of file RGB_Space.h.

double const& meow::RGBf_Space::rgbMax (size_t *i*) const `[inline]`

Definition at line 109 of file RGB_Space.h.

double const& meow::RGBf_Space::rgbMin (size_t *i*) const `[inline]`

Definition at line 105 of file RGB_Space.h.

double& meow::RGBf_Space::rGet ( ) `[inline]`

Definition at line 122 of file RGB_Space.h.

double const& meow::RGBf_Space::rMax ( ) const `[inline]`

Definition at line 110 of file RGB_Space.h.

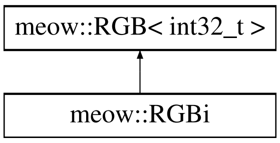
double const& meow::RGBf_Space::rMin ( ) const `[inline]`

Definition at line 106 of file RGB_Space.h.
The documentation for this class was generated from the following file:

- meowpp/colors/RGB_Space.h

6.47 meow::RGBi Class Reference

`#include "RGB.h"`
Inheritance diagram for meow::RGBi:



Public Member Functions

- RGBi ()
- RGBi (int32_t const &r, int32_t const &g, int32_t const &b)
- RGBi (int32_t const ∗rgb)
- ∼RGBi ()
- int32_t rMin () const
- int32_t rMax () const
- int32_t gMin () const
- int32_t gMax () const
- int32_t bMin () const
- int32_t bMax () const

Additional Inherited Members

6.47.1 Detailed Description

Definition at line 48 of file RGB.h.

6.47.2 Constructor & Destructor Documentation

meow::RGBi::RGBi ( ) `[inline]`

Definition at line 49 of file RGB.hpp.

meow::RGBi::RGBi (int32_t const & *r*, int32_t const & *g*, int32_t const & *b*) `[inline]`

Definition at line 51 of file RGB.hpp.

meow::RGBi::RGBi (int32_t const ∗ *rgb*) `[inline]`

Definition at line 52 of file RGB.hpp.

meow::RGBi::∼RGBi ( ) `[inline]`

Definition at line 50 of file RGB.hpp.

6.47.3 Member Function Documentation

int32_t meow::RGBi::bMax ( ) const `[inline]`, `[virtual]`

Implements meow::RGB< int32_t >.

Definition at line 58 of file RGB.hpp.

int32_t meow::RGBi::bMin ( ) const `[inline]`, `[virtual]`

Implements meow::RGB< int32_t >.

Definition at line 57 of file RGB.hpp.

int32_t meow::RGBi::gMax ( ) const `[inline]`, `[virtual]`

Implements meow::RGB< int32_t >.

Definition at line 56 of file RGB.hpp.

int32_t meow::RGBi::gMin ( ) const `[inline]`, `[virtual]`

Implements meow::RGB< int32_t >.

Definition at line 55 of file RGB.hpp.

int32_t meow::RGBi::rMax ( ) const `[inline]`, `[virtual]`

Implements meow::RGB< int32_t >.

Definition at line 54 of file RGB.hpp.

int32_t meow::RGBi::rMin ( ) const `[inline]`, `[virtual]`

Implements meow::RGB< int32_t >.

Definition at line 53 of file RGB.hpp.

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

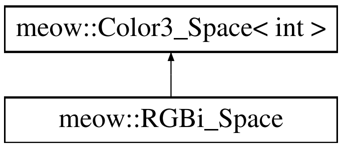
- meowpp/colors/RGB.h
- meowpp/colors/RGB.hpp

6.48 meow::RGBi_Space Class Reference

Red, Green, Blue

`#include "RGB_Space.h"`

Inheritance diagram for meow::RGBi_Space:



Public Member Functions

- RGBi_Space ()
- RGBi_Space (int c)
- RGBi_Space (Vector3D< int > const &v)
- RGBi_Space (RGBi_Space const &b)
- ∼RGBi_Space ()
- int const & rgbMin (size_t i) const
- int const & rMin () const
- int const & gMin () const
- int const & bMin () const
- int const & rgbMax (size_t i) const
- int const & rMax () const
- int const & gMax () const
- int const & bMax () const
- int const & rgb (size_t i) const
- int const & r () const
- int const & g () const
- int const & b () const
- int const & rgb (size_t i, int c)
- int const & r (int c)
- int const & g (int c)
- int const & b (int c)
- int & rgbGet (size_t i)
- int & rGet ()
- int & gGet ()
- int & bGet ()
- RGBi_Space & operator= (RGBi_Space const &b)
- RGBi_Space operator+ (RGBi_Space const &b) const
- RGBi_Space operator- (RGBi_Space const &b) const
- RGBi_Space operator∗ (int c) const
- RGBi_Space operator/ (int c) const
- int operator∗ (RGBi_Space const &b) const

Additional Inherited Members

6.48.1 Detailed Description

Red, Green, Blue

0∼255

Author

cat_leopard

Definition at line 19 of file RGB_Space.h.

6.48.2 Constructor & Destructor Documentation

meow::RGBi_Space::RGBi_Space ( ) `[inline]`

Definition at line 21 of file RGB_Space.h.

meow::RGBi_Space::RGBi_Space (int *c*) `[inline]`

Definition at line 25 of file RGB_Space.h.

meow::RGBi_Space::RGBi_Space (Vector3D< int > const & *v*) `[inline]`

Definition at line 29 of file RGB_Space.h.

meow::RGBi_Space::RGBi_Space (RGBi_Space const & *b*) `[inline]`

Definition at line 34 of file RGB_Space.h.

meow::RGBi_Space::∼RGBi_Space ( ) `[inline]`

Definition at line 36 of file RGB_Space.h.

6.48.3 Member Function Documentation

int const& meow::RGBi_Space::b ( ) const `[inline]`

Definition at line 49 of file RGB_Space.h.

int const& meow::RGBi_Space::b (int *c*) `[inline]`

Definition at line 53 of file RGB_Space.h.

int& meow::RGBi_Space::bGet ( ) `[inline]`

Definition at line 57 of file RGB_Space.h.

int const& meow::RGBi_Space::bMax ( ) const `[inline]`

Definition at line 45 of file RGB_Space.h.

int const& meow::RGBi_Space::bMin ( ) const `[inline]`

Definition at line 41 of file RGB_Space.h.

int const& meow::RGBi_Space::g ( ) const `[inline]`

Definition at line 48 of file RGB_Space.h.

int const& meow::RGBi_Space::g (int *c*) `[inline]`

Definition at line 52 of file RGB_Space.h.

int& meow::RGBi_Space::gGet ( ) `[inline]`

Definition at line 56 of file RGB_Space.h.

int const& meow::RGBi_Space::gMax ( ) const `[inline]`

Definition at line 44 of file RGB_Space.h.

int const& meow::RGBi_Space::gMin ( ) const `[inline]`

Definition at line 40 of file RGB_Space.h.

RGBi_Space meow::RGBi_Space::operator* (int *c*) const `[inline]`

Definition at line 68 of file RGB_Space.h.

int meow::RGBi_Space::operator* (RGBi_Space const & *b*) const `[inline]`

Definition at line 74 of file RGB_Space.h.

RGBi_Space meow::RGBi_Space::operator+ (RGBi_Space const & *b*) const `[inline]`

Definition at line 62 of file RGB_Space.h.

RGBi_Space meow::RGBi_Space::operator- (RGBi_Space const & *b*) const `[inline]`

Definition at line 65 of file RGB_Space.h.

RGBi_Space meow::RGBi_Space::operator/ (int *c*) const `[inline]`

Definition at line 71 of file RGB_Space.h.

RGBi_Space& meow::RGBi_Space::operator= (RGBi_Space const & *b*) `[inline]`

Definition at line 58 of file RGB_Space.h.

int const& meow::RGBi_Space::r ( ) const `[inline]`

Definition at line 47 of file RGB_Space.h.

int const& meow::RGBi_Space::r (int *c*) `[inline]`

Definition at line 51 of file RGB_Space.h.

int const& meow::RGBi_Space::rgb (size_t *i*) const `[inline]`

Definition at line 46 of file RGB_Space.h.

int const& meow::RGBi_Space::rgb (size_t *i*, int *c*) `[inline]`

Definition at line 50 of file RGB_Space.h.

int& meow::RGBi_Space::rgbGet (size_t *i*) `[inline]`

Definition at line 54 of file RGB_Space.h.

int const& meow::RGBi_Space::rgbMax (size_t *i*) const `[inline]`

Definition at line 42 of file RGB_Space.h.

int const& meow::RGBi_Space::rgbMin (size_t *i*) const `[inline]`

Definition at line 38 of file RGB_Space.h.

int& meow::RGBi_Space::rGet ( ) `[inline]`

Definition at line 55 of file RGB_Space.h.

int const& meow::RGBi_Space::rMax ( ) const [inline]

Definition at line 43 of file RGB_Space.h.

int const& meow::RGBi_Space::rMin ( ) const [inline]

Definition at line 39 of file RGB_Space.h.

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

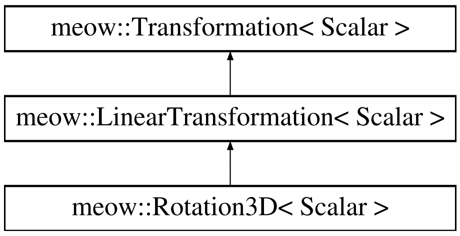
- meowpp/colors/RGB_Space.h

6.49 meow::Rotation3D< Scalar > Class Template Reference

Rotation a point/vector alone an axis with given angle in 3D world.

`#include "LinearTransformations.h"`

Inheritance diagram for meow::Rotation3D< Scalar >:


meow::Transformation< Scalar >
meow::LinearTransformation< Scalar >
meow::Rotation3D< Scalar >

Public Member Functions

- Rotation3D ()
- Rotation3D (Rotation3D const &b)
- ∼Rotation3D ()
- Rotation3D & copyFrom (Rotation3D const &b)

  Copy data.
- Rotation3D & referenceFrom (Rotation3D const &b)

  Reference data.
- Scalar parameter (size_t i) const

  same as `theta(i)`
- Scalar parameter (size_t i, Scalar const &s)

  same as `theta(i, s)`
- Scalar const & theta (size_t i) const

  Get the `i` -th theta.
- Scalar const & theta (size_t i, Scalar const &s)

  Set the `i` -th theta.
- void axisAngle (Vector< Scalar > const &axis, Scalar const &angle)

  Setting.
- Rotation3D & add (Rotation3D const &r)

  Concat another rotation transformation.
- Matrix< Scalar > transformate (Matrix< Scalar > const &x) const

  Do the transformate.
- Matrix< Scalar > jacobian (Matrix< Scalar > const &x) const

  Return the jacobian matrix (derivate by the input vector) of this transformate.
- Matrix< Scalar > jacobian (Matrix< Scalar > const &x, size_t i) const

  Return the jacobian matrix of this transformate.
- Matrix< Scalar > transformateInv (Matrix< Scalar > const &x) const

Do the inverse transformate.

- Matrix< Scalar > jacobianInv (Matrix< Scalar > const &x) const

  Return the jacobian matrix of the inverse form of this transformate.
- Matrix< Scalar > jacobianInv (Matrix< Scalar > const &x, size_t i) const

  Return the jacobian matrix of the inverse form of this transformate.
- Matrix< Scalar > matrixInv () const

  Return the inverse matrix.
- Rotation3D & operator= (Rotation3D const &b)

  same as `copyFrom(b)`

Additional Inherited Members

6.49.1 Detailed Description

template<class Scalar>class meow::Rotation3D< Scalar >

Rotation a point/vector alone an axis with given angle in 3D world.

Author

cat_leopard

Definition at line 20 of file LinearTransformations.h.

6.49.2 Constructor & Destructor Documentation

template<class Scalar> meow::Rotation3D< Scalar >::Rotation3D ( ) `[inline]`

Constructor with no rotation
Definition at line 69 of file LinearTransformations.h.

template<class Scalar> meow::Rotation3D< Scalar >::Rotation3D (Rotation3D< Scalar > const & *b*) `[inline]`

Constructor and copy data
Definition at line 75 of file LinearTransformations.h.

template<class Scalar> meow::Rotation3D< Scalar >::~Rotation3D ( ) `[inline]`

Destructor
Definition at line 82 of file LinearTransformations.h.

6.49.3 Member Function Documentation

template<class Scalar> Rotation3D& meow::Rotation3D< Scalar >::add (Rotation3D< Scalar > const & *r*) `[inline]`

Concat another rotation transformation.
Parameters

in	*r*	another rotation transformation

Definition at line 171 of file LinearTransformations.h.

template<class Scalar> void meow::Rotation3D< Scalar >::axisAngle (Vector< Scalar > const & *axis*, Scalar const & *angle*) `[inline]`

Setting.

Parameters

`in`	*axis*	axis
`in`	*angle*	angle

Definition at line 160 of file LinearTransformations.h.

template<class Scalar> Rotation3D& meow::Rotation3D< Scalar >::copyFrom (Rotation3D< Scalar > const & *b*) `[inline]`

Copy data.

Parameters

`in`	*b*	another Rotation3D class.

Returns

`*this`

Definition at line 91 of file LinearTransformations.h.

template<class Scalar> Matrix<Scalar> meow::Rotation3D< Scalar >::jacobian (Matrix< Scalar > const & *x*) const `[inline]`, `[virtual]`

Return the jacobian matrix (derivate by the input vector) of this transformate.

The matrix we return is:

$$\begin{bmatrix} 2(n_x^2-1)\sin^2\phi+1 & 2n_xn_y\sin^2\phi-2n_z\cos\phi\sin\phi & 2n_xn_z\sin^2\phi+2n_y\cos\phi\sin\phi \\ 2n_yn_x\sin^2\phi+2n_z\cos\phi\sin\phi & 2(n_y^2-1)\sin^2\phi+1 & 2n_yn_z\sin^2\phi-2n_x\cos\phi\sin\phi \\ 2n_zn_x\sin^2\phi-2n_y\cos\phi\sin\phi & 2n_zn_y\sin^2\phi+2n_x\cos\phi\sin\phi & 2(n_z^2-1)\sin^2\phi+1 \end{bmatrix}$$

Where the definition of $\vec{n}$ and ϕ is the same as the definition in the description of the method **transformate()** .

Parameters

`in`	*x*	the input vector (in this case it is a useless parameter)

Returns

a matrix

Reimplemented from meow::Transformation< Scalar >.

Definition at line 243 of file LinearTransformations.h.

template<class Scalar> Matrix<Scalar> meow::Rotation3D< Scalar >::jacobian (Matrix< Scalar > const & *x*, size_t *i*) const `[inline]`, `[virtual]`

Return the jacobian matrix of this transformate.

Here we need to discussion in three case:

- $i = 0$, derivate by the x axis of the vector theta

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 2(n_x^2-1)\sin^2\phi+1 & 2n_xn_y\sin^2\phi-2n_z\cos\phi\sin\phi & 2n_xn_z\sin^2\phi+2n_y\cos\phi\sin\phi \\ 2n_yn_x\sin^2\phi+2n_z\cos\phi\sin\phi & 2(n_y^2-1)\sin^2\phi+1 & 2n_yn_z\sin^2\phi-2n_x\cos\phi\sin\phi \\ 2n_zn_x\sin^2\phi-2n_y\cos\phi\sin\phi & 2n_zn_y\sin^2\phi+2n_x\cos\phi\sin\phi & 2(n_z^2-1)\sin^2\phi+1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

- $i = 1$, derivate by the y axis of the vector theta

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 2(n_x^2-1)\sin^2\phi+1 & 2n_xn_y\sin^2\phi-2n_z\cos\phi\sin\phi & 2n_xn_z\sin^2\phi+2n_y\cos\phi\sin\phi \\ 2n_yn_x\sin^2\phi+2n_z\cos\phi\sin\phi & 2(n_y^2-1)\sin^2\phi+1 & 2n_yn_z\sin^2\phi-2n_x\cos\phi\sin\phi \\ 2n_zn_x\sin^2\phi-2n_y\cos\phi\sin\phi & 2n_zn_y\sin^2\phi+2n_x\cos\phi\sin\phi & 2(n_z^2-1)\sin^2\phi+1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

- $i = 2$, derivate by the z axis of the vector theta

$$\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 2(n_x^2-1)\sin^2\phi+1 & 2n_xn_y\sin^2\phi-2n_z\cos\phi\sin\phi & 2n_xn_z\sin^2\phi+2n_y\cos\phi\sin\phi \\ 2n_yn_x\sin^2\phi+2n_z\cos\phi\sin\phi & 2(n_y^2-1)\sin^2\phi+1 & 2n_yn_z\sin^2\phi-2n_x\cos\phi\sin\phi \\ 2n_zn_x\sin^2\phi-2n_y\cos\phi\sin\phi & 2n_zn_y\sin^2\phi+2n_x\cos\phi\sin\phi & 2(n_z^2-1)\sin^2\phi+1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Where (x,y,z) is the input vector, $\vec{n},\phi$ is the same one in the description of **transformate()**.

Parameters

in	*x*	the input vector
in	*i*	the index of the parameters(theta) to dervite

Returns

a matrix

Reimplemented from meow::Transformation< Scalar >.
Definition at line 320 of file LinearTransformations.h.

template<class Scalar> Matrix<Scalar> meow::Rotation3D< Scalar >::jacobianInv (Matrix< Scalar > const & *x*) const [inline], [virtual]

Return the jacobian matrix of the inverse form of this transformate.
Parameters

in	*x*	the input vector

Returns

a matrix

Reimplemented from meow::Transformation< Scalar >.
Definition at line 354 of file LinearTransformations.h.

template<class Scalar> Matrix<Scalar> meow::Rotation3D< Scalar >::jacobianInv (Matrix< Scalar > const & *x*, size_t *i*) const [inline], [virtual]

Return the jacobian matrix of the inverse form of this transformate.
Parameters

in	*x*	the input vector
in	*i*	the index of the parameters(theta) to dervite

Returns

a matrix

Reimplemented from meow::Transformation< Scalar >.
Definition at line 365 of file LinearTransformations.h.

template<class Scalar> Matrix<Scalar> meow::Rotation3D< Scalar >::matrixInv ( ) const [inline], [virtual]

Return the inverse matrix.
In this case, the inverse matrix is equal to the transpose of the matrix

Returns

a matrix

Reimplemented from meow::LinearTransformation< Scalar >.
Definition at line 391 of file LinearTransformations.h.

template<class Scalar> Rotation3D& meow::Rotation3D< Scalar >::operator= (Rotation3D< Scalar > const & *b*) [inline]

same as `copyFrom(b)`
Definition at line 397 of file LinearTransformations.h.

template< class Scalar > Scalar meow::Rotation3D< Scalar >::parameter (size_t *i*) const [inline], [virtual]

same as `theta(i)`

Implements meow::Transformation< Scalar >.

Definition at line 112 of file LinearTransformations.h.

template< class Scalar > Scalar meow::Rotation3D< Scalar >::parameter (size_t *i*, Scalar const & *s*) [inline], [virtual]

same as `theta(i, s)`

Implements meow::Transformation< Scalar >.

Definition at line 119 of file LinearTransformations.h.

template< class Scalar > Rotation3D& meow::Rotation3D< Scalar >::referenceFrom (Rotation3D< Scalar > const & *b*) [inline]

Reference data.

Parameters

`in`	*b*	another Rotation3D class.

Returns

`*this`

Definition at line 103 of file LinearTransformations.h.

template< class Scalar > Scalar const& meow::Rotation3D< Scalar >::theta (size_t *i*) const [inline]

Get the `i` -th theta.

`i` can only be 1, 2 or 3

Parameters

`in`	*i*	index

Returns

`i` -th theta

Definition at line 131 of file LinearTransformations.h.

template< class Scalar > Scalar const& meow::Rotation3D< Scalar >::theta (size_t *i*, Scalar const & *s*) [inline]

Set the `i` -th theta.

`i` can only be 1, 2 or 3

Parameters

`in`	*i*	index
`in`	*s*	new theta value

Returns

`i` -th theta

Definition at line 144 of file LinearTransformations.h.

template<class Scalar> Matrix<Scalar> meow::Rotation3D< Scalar >::transformate (Matrix< Scalar > const & *x*) const [inline], [virtual]

Do the transformate.

Assume:

- The input vector is (x, y, z)
- The output vector is (x', y', z')
- The parameters theta is $\vec{\theta} = (\theta_x, \theta_y, \theta_z)$

Then we have:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 2(n_x^2-1)\sin^2\phi+1 & 2n_xn_y\sin^2\phi-2n_z\cos\phi\sin\phi & 2n_xn_z\sin^2\phi+2n_y\cos\phi\sin\phi \\ 2n_yn_x\sin^2\phi+2n_z\cos\phi\sin\phi & 2(n_y^2-1)\sin^2\phi+1 & 2n_yn_z\sin^2\phi-2n_x\cos\phi\sin\phi \\ 2n_zn_x\sin^2\phi-2n_y\cos\phi\sin\phi & 2n_zn_y\sin^2\phi+2n_x\cos\phi\sin\phi & 2(n_z^2-1)\sin^2\phi+1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Where:

- ϕ is the helf of length of $\vec{\theta}$, which means $\phi = \frac{|\vec{\theta}|}{2} = \frac{1}{2}\sqrt{\theta_x^2+\theta_y^2+\theta_z^2}$
- $\vec{n}$ is the normalized form of $\vec{\theta}$, which means $\vec{n} = (n_x, n_y, n_z) = \vec{\theta}/2\phi$

Parameters

in	*x*	the input vector

Returns

the output matrix

Implements meow::Transformation< Scalar >.

Definition at line 213 of file LinearTransformations.h.

template<class Scalar> Matrix<Scalar> meow::Rotation3D< Scalar >::transformateInv (Matrix< Scalar > const & *x*) const [inline], [virtual]

Do the inverse transformate.

Parameters

in	*x*	the input vector

Returns

the output vector

Reimplemented from meow::Transformation< Scalar >.

Definition at line 344 of file LinearTransformations.h.

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

- meowpp/math/LinearTransformations.h

6.50 meow::SceneInfo< Pixel > Struct Template Reference

```
#include "BundleAdjustment.h"
```

Public Member Functions

- SceneInfo ()
- SceneInfo (Eye< Pixel > ∗e, unsigned long f)
- SceneInfo (SceneInfo const &si)
- ∼SceneInfo ()

Public Attributes

- Eye< Pixel > * eye
- unsigned long flag

6.50.1 Detailed Description

template<class Pixel>struct meow::SceneInfo< Pixel >

Definition at line 17 of file BundleAdjustment.h.

6.50.2 Constructor & Destructor Documentation

template<class Pixel > meow::SceneInfo< Pixel >::SceneInfo ( ) `[inline]`

Definition at line 21 of file BundleAdjustment.h.

template<class Pixel > meow::SceneInfo< Pixel >::SceneInfo (Eye< Pixel > * *e*, unsigned long *f*) `[inline]`

Definition at line 24 of file BundleAdjustment.h.

template<class Pixel > meow::SceneInfo< Pixel >::SceneInfo (SceneInfo< Pixel > const & *si*) `[inline]`

Definition at line 27 of file BundleAdjustment.h.

template<class Pixel > meow::SceneInfo< Pixel >::~SceneInfo ( ) `[inline]`

Definition at line 30 of file BundleAdjustment.h.

6.50.3 Member Data Documentation

template<class Pixel > Eye<Pixel>* meow::SceneInfo< Pixel >::eye

Definition at line 18 of file BundleAdjustment.h.

template<class Pixel > unsigned long meow::SceneInfo< Pixel >::flag

Definition at line 19 of file BundleAdjustment.h.
The documentation for this struct was generated from the following file:

- meowpp/gra/BundleAdjustment.h

6.51 meow::SegmentTree< Value > Class Template Reference

`#include "SegmentTree.h"`

Public Member Functions

- SegmentTree ()

  constructor
- SegmentTree (size_t size)

  constructor, with `size` gived
- SegmentTree (SegmentTree const &tree2)

  constructor,
- SegmentTree copyFrom (SegmentTree const &b)

- size_t size () const

size

- void reset (size_t size)

  0∼size-1

- Value query (ssize_t first, ssize_t last) const

  [first,last] ()

- void override (ssize_t first, ssize_t last, Value const &value)

  [first,last] value

- void offset (ssize_t first, ssize_t last, Value const &delta)

  [first,last] delta

- SegmentTree & operator= (SegmentTree const &b)

  same as copyFrom(b)

6.51.1 Detailed Description

template<class Value>class meow::SegmentTree< Value >

, user,

Template Class Operators Request

const?	Typename	Operator	Parameters	Return Type	Description
const	Vector	operator[]	(size_t n)	Scalar	n
const	Vector	operator<	(Vector v)	bool	
const	Scalar	operator∗	(Scalar s)	Scalar	
const	Scalar	operator+	(Scalar s)	Scalar	
const	Scalar	operator-	(Scalar s)	Scalar	
const	Scalar	operator<	(Scalar s)	bool	
const	Value	operator+	(Value v)	Value	()
const	Value	operator∗	(size_t n)	Value	Value,

n | |const |Value |operator{b}|(Value v) |Value | |

- , [a, b] ,
  - operator+ ''
  - operator∗ '∗this'
  - operator| 'std::min(∗this, v)'
- , [a, b] ,
  - operator+ ''
  - operator∗ '(∗this) ∗ n'
  - operator| ''

Author

cat_leopard

Definition at line 45 of file SegmentTree.h.

6.51.2 Constructor & Destructor Documentation

template<class Value > meow::SegmentTree< Value >::SegmentTree ( ) [inline]

constructor

Definition at line 121 of file SegmentTree.h.

template<class Value > meow::SegmentTree< Value >::SegmentTree (size_t *size*) `[inline]`

constructor, with `size` gived

Definition at line 126 of file SegmentTree.h.

template<class Value > meow::SegmentTree< Value >::SegmentTree (SegmentTree< Value > const & *tree2*) `[inline]`

constructor,

Definition at line 131 of file SegmentTree.h.

6.51.3 Member Function Documentation

template<class Value > SegmentTree meow::SegmentTree< Value >::copyFrom (SegmentTree< Value > const & *b*) `[inline]`

Definition at line 138 of file SegmentTree.h.

template<class Value > void meow::SegmentTree< Value >::offset (ssize_t *first,* ssize_t *last,* Value const & *delta*) `[inline]`

[first,last] `delta`

Definition at line 181 of file SegmentTree.h.

template<class Value > SegmentTree& meow::SegmentTree< Value >::operator= (SegmentTree< Value > const & *b*) `[inline]`

same as copyFrom(b)

Definition at line 187 of file SegmentTree.h.

template<class Value > void meow::SegmentTree< Value >::override (ssize_t *first,* ssize_t *last,* Value const & *value*) `[inline]`

[first,last] `value`

Definition at line 173 of file SegmentTree.h.

template<class Value > Value meow::SegmentTree< Value >::query (ssize_t *first,* ssize_t *last*) const `[inline]`

[first,last] ()

Definition at line 165 of file SegmentTree.h.

template<class Value > void meow::SegmentTree< Value >::reset (size_t *size*) `[inline]`

`0~size-1`

Definition at line 154 of file SegmentTree.h.

template<class Value > size_t meow::SegmentTree< Value >::size ( ) const `[inline]`

size

Definition at line 147 of file SegmentTree.h.

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

- meowpp/dsa/SegmentTree.h

6.52 meow::Self< Data > Class Template Reference

A little class use for packing the data part of another class. With this technique, it can achieve Copy-On-Write(COR) mechanism at background and have a reference mechanism which much more flexible then the one C++ has.

`#include "Self.h"`

Public Types

- enum DuplicateType { COPY_FROM, REFERENCE_FROM }

  Kind of ways of duplicating.

Public Member Functions

- Self ()

  constructor with a real entity
- Self (Data const &d)

  connstructor with a real entity with it using its copy constructor
- Self (Self const &b, DuplicateType d)

  constructor with given another Self
- Self (Self const &b)

  Disallow copy constructor.
- ∼Self ()

  destructor
- Data const ∗ operator-> () const

  Return the constant pointer to the data.
- Data ∗ operator-> ()

  Return the non-constant pointer to the data (COR's clone might occure here.
- Self & operator() () const

  Return the non-constant reference of `*this`.
- Self const & copyFrom (Self const &s)

  Copy the gived `Self` to myself.
- Self const & referenceFrom (Self const &s)

  Reference myself from given `Self` object.
- Self const & duplicateFrom (Self const &s, DuplicateType t)

  call `copyFrom()` or `referenceFrom()` depend on your instruction
- bool same (Self const &s) const

  Compare tht if the gived `Self` object is reference from the same object of me.
- bool equal (Self const &s) const

  Compare that the data are the same.
- bool referenceLess (Self const &s) const

  Order compare by reference pointer.
- void operator= (Self const &a)

  Disallow default `'operator='`.

6.52.1 Detailed Description

template<class Data>class meow::Self< Data >

A little class use for packing the data part of another class. With this technique, it can achieve Copy-On-Write(COR) mechanism at background and have a reference mechanism which much more flexible then the one C++ has.

Sample code:

```
class A {
private:
  struct Myself {
    int data;

    Myself() {                              // Necessary
      data = 0;
    }

    Myself(Myself const& b): data(b.data) {  // Necessary, copy constructor
    }

    ~Myself() {
    }

    bool operator==(Myself const& b) const { // Optional (this method will
                                             // be called only if you use
                                             // Self::equal() method)
      return (data == b.data);
    }
  };

  Self<Myself> const self;           // Here we use 'constant' data type in
                                     // order to have a coutious coding style
                                     // and allow the COR mechanism to clone
                                     // data only when we really want to
                                     // modify them.
public:
  A(): self() { }                             // Default constructor

  A(A const& a): self(a.self, COPY_FROM) { } // Copy constructor. You must
                                              // tell me which way of
                                              // duplicating should I use.
                                              // It strongly recommended you
                                              // use COYP_FROM for keeping the
                                              // C++'s original behavior.
  ~A() { }

  int getMemember(int wh) const {
    return self->data;    // Use 'operator->()' to get the pointer of the data
                          // The pointer is constant or not will depend on
                          // whether the left side variable of '->' is
                          // constant or not.
                          // If we just want to read the data, use
                          // 'self' instead of 'self()'
  }
  void setMemeber(int k) {
    self()->data = k;     // As a result of 'self()' returning a non-constant
                          // reference of itself, here we get the permission
                          // for modiying data.
                          // So now we can observe that if you type
                          // 'Self<Myself> self' instead of the one above,
                          // 'self' and 'self()' will become the same one and
                          // both of them allow you using '->' for getting
                          // writing permission. At the same time, the COR
                          // machanism will become useless because everytime
                          // you want to access the date, Self will copy the
                          // data to prevent you to modify it no matter that
                          // you might just want to read it.
  }

  A referenceFrom(A const& a) {
    self.referenceFrom(a.self);
  }

  A copyFrom(A const& a) {
    self.copyFrom(a.self);
  }

  A& operator=(A const& b) { // If you really like to use operator=, it
                             // strongly recommended you use 'copyFrom()' for
                             // keeping C++'s original behavior.
    copyFrom(b);
  }
};
```

Note that 'referenceFrom()' will cause the two object become the same one, Which means that if you do something like '`a.referenceFrom(b);a.copyFrom(c);`', the result is that the value of `a`,b,c will all the same one.

Author

cathook

Warning

This class disabled the method `operator=` and copy constructor in order to prevent upexplicit default behavior, so if you want to have one of them (or both), you must implement yourself

Definition at line 104 of file Self.h.

6.52.2 Member Enumeration Documentation

template<class Data> enum meow::Self::DuplicateType

Kind of ways of duplicating.

Enumerator

COPY_FROM Normal copy operation.

REFERENCE_FROM By reference, much like pointer's copy operation.

Definition at line 109 of file Self.h.

6.52.3 Constructor & Destructor Documentation

template<class Data> meow::Self< Data >::Self ( ) `[inline]`

constructor with a real entity

Definition at line 173 of file Self.h.

template<class Data> meow::Self< Data >::Self (Data const & *d*) `[inline]`

connstructor with a real entity with it using its copy constructor

Parameters

`in`	*d*	Inital data

Definition at line 181 of file Self.h.

template<class Data> meow::Self< Data >::Self (Self< Data > const & *b*, DuplicateType *d*) `[inline]`

constructor with given another Self

Parameters

`in`	*b*	Another Self object.
`in`	*d*	To indicate type of way of duplicating

Definition at line 190 of file Self.h.

template<class Data> meow::Self< Data >::Self (Self< Data > const & *b*)

Disallow copy constructor.

template<class Data> meow::Self< Data >::~Self ( ) `[inline]`

destructor

Definition at line 206 of file Self.h.

6.52.4 Member Function Documentation

template<class Data> Self const& meow::Self< Data >::copyFrom (Self< Data > const & *s*) `[inline]`

Copy the gived `Self` to myself.

Parameters

`in`	*s*	gived `Self`

Returns

$*$this

Definition at line 233 of file Self.h.

template<class Data> Self const& meow::Self< Data >::duplicateFrom (Self< Data > const & *s*, DuplicateType *t*) `[inline]`

call `copyFrom()` or `referenceFrom()` depend on your instruction

Parameters

in	*s*	gived `Self` object
in	*t*	instruction

Returns

∗this

Definition at line 262 of file Self.h.

template< class Data > bool meow::Self< Data >::equal (Self< Data > const & *s*) const [inline]

Compare that the data are the same.

Parameters

in	*s*	another `Self` object

Returns

`true` if the data are same.

Note

This will need the method 'Data::equal()'

Definition at line 289 of file Self.h.

template< class Data > Self& meow::Self< Data >::operator() ( ) const [inline]

Return the non-constant reference of `*this`.

Definition at line 223 of file Self.h.

template< class Data > Data const∗ meow::Self< Data >::operator-> ( ) const [inline]

Return the constant pointer to the data.

Definition at line 211 of file Self.h.

template< class Data > Data∗ meow::Self< Data >::operator-> ( ) [inline]

Return the non-constant pointer to the data (COR's clone might occure here.

Definition at line 218 of file Self.h.

template< class Data > void meow::Self< Data >::operator= (Self< Data > const & *a*)

Disallow default `'operator='`.

template< class Data > Self const& meow::Self< Data >::referenceFrom (Self< Data > const & *s*) [inline]

Reference myself from given `Self` object.

Parameters

in	*s*	given `Self`

Returns

∗this

Definition at line 246 of file Self.h.

template< class Data > bool meow::Self< Data >::referenceLess (Self< Data > const & *s*) const [inline]

Order compare by reference pointer.

Parameters

in	*s*	another `Self` object

Definition at line 299 of file Self.h.

template<class Data> bool meow::Self< Data >::same (Self< Data > const & *s*) const [inline]

Compare tht if the gived `Self` object is reference from the same object of me.

Parameters

in	*s*	gived `Self` object

Returns

`true` if we are referenced to the same object.

Definition at line 277 of file Self.h.

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

- meowpp/Self.h

6.53 meow::SplayTree< Key, Value > Class Template Reference

, Key->Value . `std::map` , `split` , `merge` , `keyOffset`

`#include "SplayTree.h"`

Classes

- class Element

  `stl iterator ,Element`

Public Member Functions

- SplayTree ()

  constructor
- SplayTree (SplayTree const &tree2)

  constructor,
- ∼SplayTree ()

  destructor
- SplayTree & copyFrom (SplayTree const &tree2)

- void moveTo (SplayTree ∗tree2)

  `tree2` ,
- Element lowerBound (Key const &key) const

  () Element $k <=$ Key, .
- Element upperBound (Key const &key) const

  () Element $k <$ Key, .
- Element rLowerBound (Key const &key) const

  () Element $k >=$ Key, .
- Element rUpperBound (Key const &key) const

  () Element $k >$ Key, .
- Element find (Key const &key) const

  Key= k Elemenet . `this->end()`
- Element order (size_t order) const

  ElementsKey, `ord` Element (0).
- Element first () const

KeyElement, SplayTree, `this->end()`

- Element last () const

  KeyElement, SplayTree, `this->end()`

- Element end () const

  NULLElement,

- size_t size () const

- bool empty () const

- void clear ()

- bool insert (Key const &key, Value const &value)

  (Key —> `Value`*)*

- bool erase (Key const &key)

- void keyOffset (Key const &delta)

  ElementKey `delta`

- void splitOut (Key const &upper_bound, SplayTree ∗right)

  `tree2` *, Key >* `upper_bound` *Element*

- bool mergeAfter (SplayTree ∗tree2)

- bool merge (SplayTree ∗tree2)

- Value & operator[] (Key const &key)

  `stl::map::operator[]`

- SplayTree & operator= (SplayTree const &tree2)

  same as `copyFrom(tree2)`

6.53.1 Detailed Description

template<class Key, class Value>class meow::SplayTree< Key, Value >

, Key->Value . `std::map` , `split` , `merge` , `keyOffset`

Template Class Operators Request

const?	Typename	Operator	Parameters	Return Type	Description
const	Key	operator+	(Key k)	Key	
const	Key	operator<	(Key k)	bool	
	Key	operator=	(Key k)	Key	copy oper
	Key	Key	(int n)		,n 0
	Value	Value	( )		

Note

: -SplayTree `A` `B`, : - `B.moveTo(&A)` `B` , `A` - `A.merge(&B)` `A.mergeAfter(&B)` merge, `B`

Author

cat_leopard

Definition at line 37 of file SplayTree.h.

6.53.2 Constructor & Destructor Documentation

template< class Key , class Value > meow::SplayTree< Key, Value >::SplayTree ( ) [inline]

constructor

Definition at line 253 of file SplayTree.h.

template< class Key , class Value > meow::SplayTree< Key, Value >::SplayTree (SplayTree< Key, Value > const & *tree2*) [inline]

constructor,

Definition at line 257 of file SplayTree.h.

template< class Key , class Value > meow::SplayTree< Key, Value >::~SplayTree ( ) [inline]

destructor

Definition at line 262 of file SplayTree.h.

6.53.3 Member Function Documentation

template< class Key , class Value > void meow::SplayTree< Key, Value >::clear ( ) [inline]

Definition at line 400 of file SplayTree.h.

template< class Key , class Value > SplayTree& meow::SplayTree< Key, Value >::copyFrom (SplayTree< Key, Value > const & *tree2*) [inline]

Definition at line 269 of file SplayTree.h.

template< class Key , class Value > bool meow::SplayTree< Key, Value >::empty ( ) const [inline]

Definition at line 393 of file SplayTree.h.

template< class Key , class Value > Element meow::SplayTree< Key, Value >::end ( ) const [inline]

NULLElement,

`find` ,`order` ,`first` ,`last` Element

Definition at line 379 of file SplayTree.h.

template< class Key , class Value > bool meow::SplayTree< Key, Value >::erase (Key const & *key*) [inline]

ElementKey `key`, , `true`, `false`

Definition at line 435 of file SplayTree.h.

template< class Key , class Value > Element meow::SplayTree< Key, Value >::find (Key const & *key*) const [inline]

Key= `k` Elemenet . `this->end()`

Definition at line 339 of file SplayTree.h.

template< class Key , class Value > Element meow::SplayTree< Key, Value >::first ( ) const [inline]

KeyElement, SplayTree, `this->end()`

Definition at line 361 of file SplayTree.h.

template<class Key , class Value > bool meow::SplayTree< Key, Value >::insert (Key const & *key*, Value const & *value*) `[inline]`

(Key —> `Value`)

ElementKey `key, false ,` (Key -> Value) = (`key` -> `value`)Element, `true`

Definition at line 411 of file SplayTree.h.

template<class Key , class Value > void meow::SplayTree< Key, Value >::keyOffset (Key const & *delta*) `[inline]`

ElementKey `delta`

Definition at line 468 of file SplayTree.h.

template<class Key , class Value > Element meow::SplayTree< Key, Value >::last ( ) const `[inline]`

KeyElement, SplayTree, `this->end()`

Definition at line 369 of file SplayTree.h.

template<class Key , class Value > Element meow::SplayTree< Key, Value >::lowerBound (Key const & *key*) const `[inline]`

() Element `k` <= Key, .

`this->end()`

Definition at line 289 of file SplayTree.h.

template<class Key , class Value > bool meow::SplayTree< Key, Value >::merge (SplayTree< Key, Value > * *tree2*) `[inline]`

Key `tree2` Key, , `tree2`' Element , `tree2` , `false`

Definition at line 511 of file SplayTree.h.

template<class Key , class Value > bool meow::SplayTree< Key, Value >::mergeAfter (SplayTree< Key, Value > * *tree2*) `[inline]`

Key `tree2` Key, `tree2`' Element , `tree2` , `false`

Definition at line 494 of file SplayTree.h.

template<class Key , class Value > void meow::SplayTree< Key, Value >::moveTo (SplayTree< Key, Value > * *tree2*) `[inline]`

`tree2` ,

Definition at line 278 of file SplayTree.h.

template<class Key , class Value > SplayTree& meow::SplayTree< Key, Value >::operator= (SplayTree< Key, Value > const & *tree2*) `[inline]`

same as `copyFrom(tree2)`

Definition at line 538 of file SplayTree.h.

template<class Key , class Value > Value& meow::SplayTree< Key, Value >::operator[] (Key const & *key*) `[inline]`

`stl::map::operator`[]

ElementKey `key`, ValueReference `insert(key,Value())` Reference

Definition at line 532 of file SplayTree.h.

template<class Key , class Value > Element meow::SplayTree< Key, Value >::order (size_t *order*) const `[inline]`

ElementsKey, `ord` Element (0).

`ord>N-1, this->last()`

Definition at line 352 of file SplayTree.h.

template<class Key , class Value > Element meow::SplayTree< Key, Value >::rLowerBound (Key const & *key*) const `[inline]`

() Element `k` >= Key, .

`this->end()`

Definition at line 315 of file SplayTree.h.

template<class Key , class Value > Element meow::SplayTree< Key, Value >::rUpperBound (Key const & *key*) const `[inline]`

() Element `k` > Key, .

`this->end()`

Definition at line 328 of file SplayTree.h.

template<class Key , class Value > size_t meow::SplayTree< Key, Value >::size ( ) const `[inline]`

Definition at line 386 of file SplayTree.h.

template<class Key , class Value > void meow::SplayTree< Key, Value >::splitOut (Key const & *upper_bound,* SplayTree< Key, Value > * *right*) `[inline]`

`tree2` , Key > `upper_bound` Element

Definition at line 477 of file SplayTree.h.

template<class Key , class Value > Element meow::SplayTree< Key, Value >::upperBound (Key const & *key*) const `[inline]`

() Element `k` < Key, .

`this->end()`

Definition at line 302 of file SplayTree.h.

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

- meowpp/dsa/SplayTree.h

6.54 meow::SplayTree_Range< Key, Value > Class Template Reference

SplayTree, , (operator `SegmentTree`)

`#include "SplayTree.h"`

Classes

- class Element

  `stl iterator ,Element`

Public Member Functions

- SplayTree_Range ()

  constructor
- SplayTree_Range (SplayTree_Range const &tree2)

  constructor,
- ~SplayTree_Range ()

destructor

- SplayTree_Range & copyFrom (SplayTree_Range const &tree2)
- void moveTo (SplayTree_Range ∗tree2)

  `tree2` ,
- Element lowerBound (Key const &key) const

  () Element $k <=$ *Key, .*
- Element upperBound (Key const &key) const

  () Element $k <$ *Key, .*
- Element rLowerBound (Key const &key) const

  () Element $k >=$ *Key, .*
- Element rUpperBound (Key const &key) const

  () Element $k >$ *Key, .*
- Element find (Key const &key) const

  Key= k *Elemenet .* `this->end()`
- Element order (size_t order) const

  ElementsKey, `ord` *Element (0).*
- Element first () const

  KeyElement, SplayTree, `this->end()`
- Element last () const

  KeyElement, SplayTree, `this->end()`
- Element end () const

  NULLElement,
- size_t size () const
- bool empty () const
- Value query () const
- Value query (Key const &first, Key const &last) const
- void clear ()
- bool insert (Key const &key, Value const &value)

  (Key —> `Value`*)*
- bool erase (Key const &key)
- void keyOffset (Key const &delta)

  ElementKey `delta`
- void valueOffset (Value const &delta)

  ElementValue `delta`
- void valueOverride (Value const &value)

  ElementValue`value`
- void splitOut (Key const &upper_bound, SplayTree_Range ∗right)

  `tree2` *, Key >* `upper_bound` *Element*
- bool mergeAfter (SplayTree_Range ∗tree2)
- bool merge (SplayTree_Range ∗tree2)
- Value & operator[] (Key const &key)

  `stl::map::operator[]`
- SplayTree_Range & operator= (SplayTree_Range const &tree2)

  same as `copyFrom(tree2)`

6.54.1 Detailed Description

template<class Key, class Value>class meow::SplayTree_Range< Key, Value >

SplayTree, , (operator `SegmentTree`)

Template Class Operators Request

const?	Typename	Operator	Parameters	Return Type	Description
const	Key	operator+	(Key `k`)	Key	
const	Key	operator<	(Key `k`)	bool	
	Key	operator=	(Key `k`)	Key	copy oper
	Key	Key	(int `n`)		,`n` 0
	Value	Value	( )		

Note

: -SplayTree `A` `B`, : - `B.moveTo(&A)` `B` , `A` - `A.merge(&B)` `A.mergeAfter(&B)` merge, `B`

Author

cat_leopard

Definition at line 569 of file SplayTree.h.

6.54.2 Constructor & Destructor Documentation

template<class Key , class Value > meow::SplayTree_Range< Key, Value >::SplayTree_Range ( ) `[inline]`

constructor

Definition at line 812 of file SplayTree.h.

template<class Key , class Value > meow::SplayTree_Range< Key, Value >::SplayTree_Range (SplayTree_Range< Key, Value > const & *tree2*) `[inline]`

constructor,

Definition at line 816 of file SplayTree.h.

template<class Key , class Value > meow::SplayTree_Range< Key, Value >::~SplayTree_Range ( ) `[inline]`

destructor

Definition at line 821 of file SplayTree.h.

6.54.3 Member Function Documentation

template<class Key , class Value > void meow::SplayTree_Range< Key, Value >::clear ( ) `[inline]`

Definition at line 988 of file SplayTree.h.

template<class Key , class Value > SplayTree_Range& meow::SplayTree_Range< Key, Value >::copyFrom (SplayTree_Range< Key, Value > const & *tree2*) `[inline]`

Definition at line 828 of file SplayTree.h.

template<class Key , class Value > bool meow::SplayTree_Range< Key, Value >::empty ( ) const `[inline]`

Definition at line 952 of file SplayTree.h.

template<class Key , class Value > Element meow::SplayTree_Range< Key, Value >::end ( ) const [inline]

NULLElement,
`find` ,`order` ,`first` ,`last` Element
Definition at line 938 of file SplayTree.h.

template<class Key , class Value > bool meow::SplayTree_Range< Key, Value >::erase (Key const & *key*) [inline]

ElementKey `key`, , `true`, `false`
Definition at line 1023 of file SplayTree.h.

template<class Key , class Value > Element meow::SplayTree_Range< Key, Value >::find (Key const & *key*) const [inline]

Key= `k` Elemenet . `this->end()`
Definition at line 898 of file SplayTree.h.

template<class Key , class Value > Element meow::SplayTree_Range< Key, Value >::first ( ) const [inline]

KeyElement, SplayTree, `this->end()`
Definition at line 920 of file SplayTree.h.

template<class Key , class Value > bool meow::SplayTree_Range< Key, Value >::insert (Key const & *key*, Value const & *value*) [inline]

(Key —> `Value`)
ElementKey `key`, `false` , (Key -> Value) = (`key` -> `value`)Element, `true`
Definition at line 999 of file SplayTree.h.

template<class Key , class Value > void meow::SplayTree_Range< Key, Value >::keyOffset (Key const & *delta*) [inline]

ElementKey `delta`
Definition at line 1056 of file SplayTree.h.

template<class Key , class Value > Element meow::SplayTree_Range< Key, Value >::last ( ) const [inline]

KeyElement, SplayTree, `this->end()`
Definition at line 928 of file SplayTree.h.

template<class Key , class Value > Element meow::SplayTree_Range< Key, Value >::lowerBound (Key const & *key*) const [inline]

() Element `k` <= Key, .
`this->end()`
Definition at line 848 of file SplayTree.h.

template<class Key , class Value > bool meow::SplayTree_Range< Key, Value >::merge (SplayTree_Range< Key, Value > * *tree2*) [inline]

Key `tree2` Key, , `tree2` ' Element , `tree2` , `false`
Definition at line 1117 of file SplayTree.h.

template<class Key , class Value > bool meow::SplayTree_Range< Key, Value >::mergeAfter (SplayTree_Range< Key, Value > ∗ *tree2*) `[inline]`

Key `tree2` Key, `tree2`' Element , `tree2` , `false`

Definition at line 1100 of file SplayTree.h.

template<class Key , class Value > void meow::SplayTree_Range< Key, Value >::moveTo (SplayTree_Range< Key, Value > ∗ *tree2*) `[inline]`

`tree2` ,

Definition at line 837 of file SplayTree.h.

template<class Key , class Value > SplayTree_Range& meow::SplayTree_Range< Key, Value >::operator= (SplayTree_Range< Key, Value > const & *tree2*) `[inline]`

same as `copyFrom(tree2)`

Definition at line 1144 of file SplayTree.h.

template<class Key , class Value > Value& meow::SplayTree_Range< Key, Value >::operator[] (Key const & *key*) `[inline]`

`stl::map::operator[]`

ElementKey `key`, ValueReference `insert(key,Value())` Reference

Definition at line 1138 of file SplayTree.h.

template<class Key , class Value > Element meow::SplayTree_Range< Key, Value >::order (size_t *order*) const `[inline]`

ElementsKey, `ord` Element (0).

`ord>N-1`, `this->last()`

Definition at line 911 of file SplayTree.h.

template<class Key , class Value > Value meow::SplayTree_Range< Key, Value >::query ( ) const `[inline]`

range

Definition at line 961 of file SplayTree.h.

template<class Key , class Value > Value meow::SplayTree_Range< Key, Value >::query (Key const & *first*, Key const & *last*) const `[inline]`

range

Definition at line 971 of file SplayTree.h.

template<class Key , class Value > Element meow::SplayTree_Range< Key, Value >::rLowerBound (Key const & *key*) const `[inline]`

() Element `k` >= Key, .

`this->end()`

Definition at line 874 of file SplayTree.h.

template<class Key , class Value > Element meow::SplayTree_Range< Key, Value >::rUpperBound (Key const & *key*) const `[inline]`

() Element `k` > Key, .

`this->end()`

Definition at line 887 of file SplayTree.h.

template< class Key , class Value > size_t meow::SplayTree_Range< Key, Value >::size ( ) const `[inline]`

Definition at line 945 of file SplayTree.h.

template< class Key , class Value > void meow::SplayTree_Range< Key, Value >::splitOut (Key const & *upper_bound*, SplayTree_Range< Key, Value > ∗ *right*) `[inline]`

`tree2` , Key > `upper_bound` Element
Definition at line 1083 of file SplayTree.h.

template< class Key , class Value > Element meow::SplayTree_Range< Key, Value >::upperBound (Key const & *key*) const `[inline]`

() Element `k` < Key, .
`this->end()`
Definition at line 861 of file SplayTree.h.

template< class Key , class Value > void meow::SplayTree_Range< Key, Value >::valueOffset (Value const & *delta*) `[inline]`

ElementValue `delta`
Definition at line 1065 of file SplayTree.h.

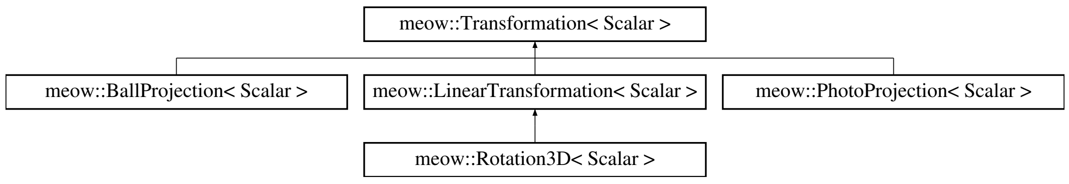
template< class Key , class Value > void meow::SplayTree_Range< Key, Value >::valueOverride (Value const & *value*) `[inline]`

ElementValue`value`
Definition at line 1074 of file SplayTree.h.
The documentation for this class was generated from the following file:

- meowpp/dsa/SplayTree.h

6.55 meow::Transformation< Scalar > Class Template Reference

A base class for implementing kinds of transformations.
`#include "Transformation.h"`
Inheritance diagram for meow::Transformation< Scalar >:


meow::Transformation< Scalar >
meow::BallProjection< Scalar >
meow::LinearTransformation< Scalar >
meow::PhotoProjection< Scalar >
meow::Rotation3D< Scalar >

Public Member Functions

- virtual ∼Transformation ()
- size_t inputRows () const

  Return the number of rows of the input matrix.
- size_t inputCols () const

  Return the number of columns of the input matrix.
- size_t outputRows () const

  Return the number of rows of the output matrix.
- size_t outputCols () const

  Return the number of columns of the output matrix.

- size_t parameterSize () const

  Return the number of parameters.
- virtual Scalar parameter (size_t i) const =0

  Get the i -th parameter.
- virtual Scalar parameter (size_t i, Scalar const &s)=0

  Setup the i -th parameter.
- virtual Matrix< Scalar > transformate (Matrix< Scalar > const &x) const =0

  Do transformate.
- virtual Matrix< Scalar > jacobian (Matrix< Scalar > const &x) const

  Calculate the jacobian matrix (derivate by the input matrix) of the transformation.
- virtual Matrix< Scalar > jacobian (Matrix< Scalar > const &x, size_t i) const

  Calculate the jacobian matrix (derivate by the i -th parameter) of the transformation.
- virtual bool inversable () const

  Return whether this transformation is inversable or not.
- virtual Matrix< Scalar > transformateInv (Matrix< Scalar > const &x) const

  Do the inverse transformation.
- virtual Matrix< Scalar > jacobianInv (Matrix< Scalar > const &x) const

  Return the jacobian matrix of the inverse transformation.
- virtual Matrix< Scalar > jacobianInv (Matrix< Scalar > const &x, size_t i) const

  Return the jacobian matrix of the inverse transformation.

Protected Member Functions

- Transformation (size_t inputRows, size_t inputCols, size_t outputRows, size_t outputCols, size_t psize)
- Transformation (Transformation const &b)
- Transformation & copyFrom (Transformation const &b)

  Copy from the specified one.
- Transformation & referenceFrom (Transformation const &b)

  reference from the specified one

6.55.1 Detailed Description

template<class Scalar>class meow::Transformation< Scalar >

A base class for implementing kinds of transformations.

We define that the input and output form of our transformations all be **matrix** . Some advance methods such as calculating jacobian matrix will require that the input form must be a vector.

Author

cat_leopard

Definition at line 21 of file Transformation.h.

6.55.2 Constructor & Destructor Documentation

template<class Scalar> meow::Transformation< Scalar >::Transformation (size_t *inputRows,* size_t *inputCols,* size_t *outputRows,* size_t *outputCols,* size_t *psize*) `[inline], [protected]`

Construct and setup

Parameters

in	*inputRows*	number of rows of the input matrix.
in	*inputCols*	number of columns of the input matrix.
in	*outputRows*	number of rows of the output matrix.
in	*outputCols*	number of columns of the output matrix.
in	*psize*	number of parameters

Definition at line 55 of file Transformation.h.

template<class Scalar> meow::Transformation< Scalar >::Transformation (Transformation< Scalar > const & *b*) `[inline]`, `[protected]`

Construct and copy setings from another transformation class.

Parameters

in	*b*	Specify where to copy the informations.

Definition at line 65 of file Transformation.h.

template<class Scalar> virtual meow::Transformation< Scalar >::~Transformation ( ) `[inline]`, `[virtual]`

Destructor

Definition at line 94 of file Transformation.h.

6.55.3 Member Function Documentation

template<class Scalar> Transformation& meow::Transformation< Scalar >::copyFrom (Transformation< Scalar > const & *b*) `[inline]`, `[protected]`

Copy from the specified one.

Parameters

in	*b*	The specified one

Returns

`*this`

Definition at line 75 of file Transformation.h.

template<class Scalar> size_t meow::Transformation< Scalar >::inputCols ( ) const `[inline]`

Return the number of columns of the input matrix.

Returns

Number of columns.

Definition at line 111 of file Transformation.h.

template<class Scalar> size_t meow::Transformation< Scalar >::inputRows ( ) const `[inline]`

Return the number of rows of the input matrix.

Returns

Number of rows.

Definition at line 102 of file Transformation.h.

template<class Scalar> virtual bool meow::Transformation< Scalar >::inversable ( ) const `[inline], [virtual]`

Return whether this transformation is inversable or not.

Returns

`false`

Definition at line 201 of file Transformation.h.

template<class Scalar> virtual Matrix<Scalar> meow::Transformation< Scalar >::jacobian (Matrix< Scalar > const & *x*) const `[inline], [virtual]`

Calculate the jacobian matrix (derivate by the input matrix) of the transformation.

Consider the case of a non-differentiable transformation might be implemented, we return an empty matrix now instead of making it be a pure virtual method.

Parameters

`in`	*x*	The input matrix.

Returns

An empty matrix.

Reimplemented in meow::PhotoProjection< Scalar >, meow::PhotoProjection< double >, meow::Rotation3D< Scalar >, meow::Rotation3D< double >, meow::BallProjection< Scalar >, and meow::BallProjection< double >.

Definition at line 177 of file Transformation.h.

template<class Scalar> virtual Matrix<Scalar> meow::Transformation< Scalar >::jacobian (Matrix< Scalar > const & *x*, size_t *i*) const `[inline], [virtual]`

Calculate the jacobian matrix (derivate by the *i* -th parameter) of the transformation.

Consider the case of a non-differentiable transformation might be implemented, we return an empty matrix now instead of making it be a pure virtual method.

Parameters

`in`	*x*	The input matrix.
`in`	*i*	The index of the specified parameter.

Returns

An empty matrix.

Reimplemented in meow::PhotoProjection< Scalar >, meow::PhotoProjection< double >, meow::Rotation3D< Scalar >, meow::Rotation3D< double >, meow::BallProjection< Scalar >, and meow::BallProjection< double >.

Definition at line 192 of file Transformation.h.

template<class Scalar> virtual Matrix<Scalar> meow::Transformation< Scalar >::jacobianInv (Matrix< Scalar > const & *x*) const `[inline], [virtual]`

Return the jacobian matrix of the inverse transformation.

Parameters

`in`	*x*	The input matirx

Returns

An empty matrix

Reimplemented in meow::Rotation3D< Scalar >, and meow::Rotation3D< double >.

Definition at line 219 of file Transformation.h.

template<class Scalar> virtual Matrix<Scalar> meow::Transformation< Scalar >::jacobianInv (Matrix< Scalar > const & *x*, size_t *i*) const `[inline]`, `[virtual]`

Return the jacobian matrix of the inverse transformation.

Parameters

in	*x*	The input matirx
in	*i*	The index of the specified parameter.

Returns

An empty matrix

Reimplemented in meow::Rotation3D< Scalar >, and meow::Rotation3D< double >.
Definition at line 230 of file Transformation.h.

template<class Scalar> size_t meow::Transformation< Scalar >::outputCols ( ) const [inline]

Return the number of columns of the output matrix.

Returns

Number of columns.

Definition at line 129 of file Transformation.h.

template<class Scalar> size_t meow::Transformation< Scalar >::outputRows ( ) const [inline]

Return the number of rows of the output matrix.

Returns

Number of rows.

Definition at line 120 of file Transformation.h.

template<class Scalar> virtual Scalar meow::Transformation< Scalar >::parameter (size_t *i*) const [pure virtual]

Get the *i* -th parameter.

Parameters

in	*i*	The index of the specified parameter.

Note

It's a pure virtual method.

Implemented in meow::PhotoProjection< Scalar >, meow::PhotoProjection< double >, meow::BallProjection< Scalar >, meow::BallProjection< double >, meow::Rotation3D< Scalar >, and meow::Rotation3D< double >.

template<class Scalar> virtual Scalar meow::Transformation< Scalar >::parameter (size_t *i*, Scalar const & *s*) [pure virtual]

Setup the *i* -th parameter.

Parameters

in	*i*	The index of the specified parameter.
in	*s*	The new value to the specified parameter.

Note

It's a pure virtual method.

Implemented in meow::PhotoProjection< Scalar >, meow::PhotoProjection< double >, meow::BallProjection< Scalar >, meow::BallProjection< double >, meow::Rotation3D< Scalar >, and meow::Rotation3D< double >.

template<class Scalar> size_t meow::Transformation< Scalar >::parameterSize ( ) const [inline]

Return the number of parameters.

Returns

Number of parameters.

Definition at line 138 of file Transformation.h.

template<class Scalar> Transformation& meow::Transformation< Scalar >::referenceFrom (Transformation< Scalar > const & *b*) [inline], [protected]

reference from the specified one

Parameters

in	*b*	The specified one

Returns

`*this`

Definition at line 86 of file Transformation.h.

template<class Scalar> virtual Matrix<Scalar> meow::Transformation< Scalar >::transformate (Matrix< Scalar > const & *x*) const [pure virtual]

Do transformate.

Parameters

in	*x*	The input matrix.

Note

It's a pure virtual method.

Implemented in meow::PhotoProjection< Scalar >, meow::PhotoProjection< double >, meow::Rotation3D< Scalar >, meow::Rotation3D< double >, meow::BallProjection< Scalar >, and meow::BallProjection< double >.

template<class Scalar> virtual Matrix<Scalar> meow::Transformation< Scalar >::transformateInv (Matrix< Scalar > const & *x*) const [inline], [virtual]

Do the inverse transformation.

Parameters

in	*x*	The input matirx

Returns

An empty matrix

Reimplemented in meow::Rotation3D< Scalar >, and meow::Rotation3D< double >.
Definition at line 209 of file Transformation.h.
The documentation for this class was generated from the following file:

- meowpp/math/Transformation.h

6.56 meow::Usage Class Reference

, usage document, argc, argv

`#include "Usage.h"`

Public Member Functions

- Usage ()

  constructor
- Usage (String const &name)

  constructor
- Usage (Usage const &usage)

  constructor
- bool import (Usage const &usage)

  usage
- bool update (Usage const &usage)

  usage
- bool optionAdd (String opt, String const &des)

- bool optionAdd (String opt, String const &des, String const &val_type, String const &val_default, bool must)

- bool optionValueAcceptAdd (String opt, String const &val, String const &des)

  -(opt)
- bool hasOptionSetup (String opt) const

- size_t optionValuesSize (String opt) const

  -*(opt)*
- String optionValue (String opt, size_t index) const

  -*(opt)* ***index***
- size_t procArgsSize () const

  process arguments
- String procArg (size_t index) const

  iprocess argument
- Strings const & procArgs () const

  process arguments array
- void usageBeginAdd (String const &des)

  usage document
- void usageEndAdd (String const &des)

  usage document
- String usage () const

  usage string
- bool arguments (int argc, char **argv, String *errmsg)

  argc, argv,

6.56.1 Detailed Description

, usage document, argc, argv

Usage argc, argvusage documentclass.
argc, argv,

- **-c** *c* ,

- **-c** *value* *value* , , valuevalue

- *value* , **process arguments**

Author

cathook

Definition at line 26 of file Usage.h.

6.56.2 Constructor & Destructor Documentation

meow::Usage::Usage ( ) `[inline]`

constructor

<name>

Definition at line 184 of file Usage.h.

meow::Usage::Usage (String const & *name*) `[inline]`

constructor

"<name>" **name**

Definition at line 192 of file Usage.h.

meow::Usage::Usage (Usage const & *usage*) `[inline]`

constructor

usage

Definition at line 202 of file Usage.h.

6.56.3 Member Function Documentation

bool meow::Usage::arguments (int *argc,* char ** *argv,* String * *errmsg*) `[inline]`

argc, argv,

Parameters

`in`	*argc,argv*	
`out`	*errmsg*	(NULL pointer,)

Returns

`true/false` ()

Definition at line 414 of file Usage.h.

bool meow::Usage::hasOptionSetup (String *opt*) const `[inline]`

Parameters

`in`	*opt*	

Returns

`true/false`

Definition at line 304 of file Usage.h.

bool meow::Usage::import (Usage const & *usage*) `[inline]`

usage

Parameters

`in`	*usage*	usage

Returns

`true/false`

Definition at line 216 of file Usage.h.

bool meow::Usage::optionAdd (String *opt,* String const & *des*) `[inline]`

Parameters

`in`	*opt*	
`in`	*des*	description,

Returns

`true/false`

Definition at line 258 of file Usage.h.

bool meow::Usage::optionAdd (String *opt*, String const & *des*, String const & *val_type*, String const & *val_default*, bool *must*) `[inline]`

Parameters

`in`	*opt*	
`in`	*des*	description,
`in`	*val_type*	, USAGE
`in`	*val_default*	,
`in`	*must*	

Returns

`true/false`

Definition at line 274 of file Usage.h.

String meow::Usage::optionValue (String *opt*, size_t *index*) const `[inline]`

-(opt) **index**

Parameters

`in`	*opt*	
`in`	*index*	

Returns

-(opt) **index**

Definition at line 328 of file Usage.h.

bool meow::Usage::optionValueAcceptAdd (String *opt*, String const & *val*, String const & *des*) `[inline]`

-(opt)

Parameters

`in`	*opt*	
`in`	*val*	
`in`	*des*	

Returns

`true/false`

Definition at line 292 of file Usage.h.

size_t meow::Usage::optionValuesSize (String *opt*) const `[inline]`

-(opt)

Parameters

`in`	*opt*	

Returns

Definition at line 315 of file Usage.h.

String meow::Usage::procArg (size_t *index*) const [inline]

iprocess argument

Parameters

`in`	*index*	

Returns

index **process argument**

Definition at line 349 of file Usage.h.

Strings const& meow::Usage::procArgs ( ) const [inline]

process arguments array

Returns

`std::vector`, **Process arguments**

Definition at line 361 of file Usage.h.

size_t meow::Usage::procArgsSize ( ) const [inline]

process arguments

Returns

process arguments

Definition at line 339 of file Usage.h.

bool meow::Usage::update (Usage const & *usage*) [inline]

usage

Parameters

`in`	*usage*	usage

Returns

`true/false`

Definition at line 239 of file Usage.h.

String meow::Usage::usage ( ) const [inline]

usage string

Returns

usage string

Definition at line 388 of file Usage.h.

void meow::Usage::usageBeginAdd (String const & *des*) [inline]

usage document

Parameters

in	*des*	usage document

Definition at line 370 of file Usage.h.

void meow::Usage::usageEndAdd (String const & *des*) `[inline]`

usage document

Parameters

in	*des*	usage document

Definition at line 379 of file Usage.h.

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

- meowpp/Usage.h
- meowpp/Usage.hpp

6.57 meow::Vector< Scalar > Class Template Reference

vector

```
#include "Vector.h"
```

Public Types

- typedef Matrix< Scalar >::EntryRefK ScalarRefK
- typedef Matrix< Scalar >::EntryRef ScalarRef

Public Member Functions

- Vector ()

  constructor
- Vector (Vector const &v)

  constructor
- Vector (Matrix< Scalar > const &m)

  constructor
- Vector (std::vector< Scalar > const &v)

  constructor
- Vector (size_t d, Scalar const &e)

  constructor
- ∼Vector ()

  destructor
- Vector & copyFrom (Vector const &v)

  copy from ...
- Vector & referenceFrom (Vector const &v)

  reference from ...
- Matrix< Scalar > const & matrix () const

  Return a dimension x 1 matrix form of it.
- size_t dimension () const

  return dimension
- size_t dimension (size_t d, Scalar const &s)

  resize the dimension
- bool valid () const

  Return whether `dimension>0` is true or not.
- Scalar entry (size_t i) const

return i -th entry

- Scalar entry (size_t i, Scalar const &s)

  change i -th entry
- ScalarRef entryGet (size_t i)

  return i -th entry with non-constant type
- void entries (size_t i, size_t j, Scalar const &s)

  change i -th to j -th entries
- Vector subVector (size_t i, size_t j)

  subvector form i-th to j-th
- Vector positive () const

  return +(∗this)
- Vector negative () const

  return -(∗this)
- Vector add (Vector const &v) const

  return (∗this)+v
- Vector sub (Vector const &v) const

  return (∗this)-v
- Vector mul (Scalar const &s) const

  return (∗this)∗s , where s is a scalar
- Vector div (Scalar const &s) const

  return (∗this)/s , where s is a scalar
- Scalar dot (Vector const &v) const

  dot
- Scalar length () const

  sqrt of length2
- Scalar length2 () const

  same as (∗this).dot(∗this)
- Vector normalize () const

  return a normalize form of itself
- Vector & normalized ()

  Let itself be normalize form.
- Vector & operator= (Vector const &v)

  same as copyFrom
- Scalar operator() (size_t i) const

  same as entry(i)
- Vector operator+ () const

  same as positive()
- Vector operator- () const

  same as negative()
- Vector operator+ (Vector const &v) const

  same as add(v)
- Vector operator- (Vector const &v) const

  same as sub(v)
- Scalar operator∗ (Vector const &v) const

  same as dot(v)
- Vector operator∗ (Scalar const &s) const

  same as mul(s)
- Vector operator/ (Scalar const &s) const

  same as div(s)

6.57.1 Detailed Description

template<class Scalar>class meow::Vector< Scalar >

vector

Author

cat_leopard

Definition at line 19 of file Vector.h.

6.57.2 Member Typedef Documentation

template<class Scalar> typedef Matrix<Scalar>::EntryRef meow::Vector< Scalar >::ScalarRef

Definition at line 22 of file Vector.h.

template<class Scalar> typedef Matrix<Scalar>::EntryRefK meow::Vector< Scalar >::ScalarRefK

Definition at line 21 of file Vector.h.

6.57.3 Constructor & Destructor Documentation

template<class Scalar> meow::Vector< Scalar >::Vector ( ) `[inline]`

constructor

With **dimension=0**, which means **invalid**.

Definition at line 31 of file Vector.h.

template<class Scalar> meow::Vector< Scalar >::Vector (Vector< Scalar > const & *v*) `[inline]`

constructor

Copy from another vector

Parameters

`in`	*v*	another vector

Definition at line 41 of file Vector.h.

template<class Scalar> meow::Vector< Scalar >::Vector (Matrix< Scalar > const & *m*) `[inline]`

constructor

From matrix's first column

Parameters

`in`	*m*	matrix

Definition at line 51 of file Vector.h.

template<class Scalar> meow::Vector< Scalar >::Vector (std::vector< Scalar > const & *v*) `[inline]`

constructor

Copy from another std::vector

Parameters

`in`	*v*	vector

Definition at line 61 of file Vector.h.

template<class Scalar> meow::Vector< Scalar >::Vector (size_t *d*, Scalar const & *e*) `[inline]`

constructor

setup dimension and inital value

Parameters

`in`	*d*	dimension
`in`	*e*	inital value

Definition at line 75 of file Vector.h.

template<class Scalar> meow::Vector< Scalar >::~Vector ( ) `[inline]`

destructor

Definition at line 79 of file Vector.h.

6.57.4 Member Function Documentation

template<class Scalar> Vector meow::Vector< Scalar >::add (Vector< Scalar > const & *v*) const `[inline]`

return (∗this)+v

Definition at line 174 of file Vector.h.

template<class Scalar> Vector& meow::Vector< Scalar >::copyFrom (Vector< Scalar > const & *v*) `[inline]`

copy from ...

Definition at line 83 of file Vector.h.

template<class Scalar> size_t meow::Vector< Scalar >::dimension ( ) const `[inline]`

return dimension

Definition at line 100 of file Vector.h.

template<class Scalar> size_t meow::Vector< Scalar >::dimension (size_t *d*, Scalar const & *s*) `[inline]`

resize the dimension

Parameters

`in`	*d*	new dimension
`in`	*s*	inital entry

Returns

new dimension

Definition at line 111 of file Vector.h.

template<class Scalar> Vector meow::Vector< Scalar >::div (Scalar const & *s*) const `[inline]`

return (∗this)/s , where s is a scalar

Definition at line 189 of file Vector.h.

template<class Scalar> Scalar meow::Vector< Scalar >::dot (Vector< Scalar > const & *v*) const `[inline]`

dot

Definition at line 194 of file Vector.h.

template<class Scalar> void meow::Vector< Scalar >::entries (size_t *i*, size_t *j*, Scalar const & *s*) `[inline]`

change *i* -th to *j* -th entries

Parameters

in	*i*	i-th
in	*j*	j-th
in	*s*	new value

Definition at line 152 of file Vector.h.

template<class Scalar> Scalar meow::Vector< Scalar >::entry (size_t *i*) const [inline]

return *i* -th entry

Definition at line 125 of file Vector.h.

template<class Scalar> Scalar meow::Vector< Scalar >::entry (size_t *i*, Scalar const & *s*) [inline]

change *i* -th entry

Parameters

in	*i*	i-th
in	*s*	new value

Definition at line 135 of file Vector.h.

template<class Scalar> ScalarRef meow::Vector< Scalar >::entryGet (size_t *i*) [inline]

return *i* -th entry with non-constant type

Definition at line 141 of file Vector.h.

template<class Scalar> Scalar meow::Vector< Scalar >::length ( ) const [inline]

sqrt of *length2*

Definition at line 199 of file Vector.h.

template<class Scalar> Scalar meow::Vector< Scalar >::length2 ( ) const [inline]

same as (∗this).dot(∗this)

Definition at line 204 of file Vector.h.

template<class Scalar> Matrix<Scalar> const& meow::Vector< Scalar >::matrix ( ) const [inline]

Return a *dimension* x 1 matrix form of it.

Definition at line 95 of file Vector.h.

template<class Scalar> Vector meow::Vector< Scalar >::mul (Scalar const & *s*) const [inline]

return (∗this)∗s , where s is a scalar

Definition at line 184 of file Vector.h.

template<class Scalar> Vector meow::Vector< Scalar >::negative ( ) const [inline]

return -(∗this)

Definition at line 169 of file Vector.h.

template<class Scalar> Vector meow::Vector< Scalar >::normalize ( ) const [inline]

return a normalize form of itself

Definition at line 209 of file Vector.h.

template<class Scalar> Vector& meow::Vector< Scalar >::normalized ( ) [inline]

Let itself be normalize form.

Definition at line 214 of file Vector.h.

template<class Scalar> Scalar meow::Vector< Scalar >::operator() (size_t *i*) const `[inline]`

same as entry(i)

Definition at line 225 of file Vector.h.

template<class Scalar> Scalar meow::Vector< Scalar >::operator∗ (Vector< Scalar > const & *v*) const `[inline]`

same as dot(v)

Definition at line 250 of file Vector.h.

template<class Scalar> Vector meow::Vector< Scalar >::operator∗ (Scalar const & *s*) const `[inline]`

same as mul(s)

Definition at line 255 of file Vector.h.

template<class Scalar> Vector meow::Vector< Scalar >::operator+ ( ) const `[inline]`

same as positive()

Definition at line 230 of file Vector.h.

template<class Scalar> Vector meow::Vector< Scalar >::operator+ (Vector< Scalar > const & *v*) const `[inline]`

same as add(v)

Definition at line 240 of file Vector.h.

template<class Scalar> Vector meow::Vector< Scalar >::operator- ( ) const `[inline]`

same as negative()

Definition at line 235 of file Vector.h.

template<class Scalar> Vector meow::Vector< Scalar >::operator- (Vector< Scalar > const & *v*) const `[inline]`

same as sub(v)

Definition at line 245 of file Vector.h.

template<class Scalar> Vector meow::Vector< Scalar >::operator/ (Scalar const & *s*) const `[inline]`

same as div(s)

Definition at line 260 of file Vector.h.

template<class Scalar> Vector& meow::Vector< Scalar >::operator= (Vector< Scalar > const & *v*) `[inline]`

same as copyFrom

Definition at line 220 of file Vector.h.

template<class Scalar> Vector meow::Vector< Scalar >::positive ( ) const `[inline]`

return +(∗this)

Definition at line 164 of file Vector.h.

template<class Scalar> Vector& meow::Vector< Scalar >::referenceFrom (Vector< Scalar > const & *v*) `[inline]`

reference from ...

Definition at line 89 of file Vector.h.

template<class Scalar> Vector meow::Vector< Scalar >::sub (Vector< Scalar > const & *v*) const `[inline]`

return (∗this)-v

Definition at line 179 of file Vector.h.

template<class Scalar> Vector meow::Vector< Scalar >::subVector (size_t *i*, size_t *j*) `[inline]`

subvector form i-th to j-th

Definition at line 159 of file Vector.h.

template<class Scalar> bool meow::Vector< Scalar >::valid ( ) const `[inline]`

Return whether `dimension>0` is true or not.

Returns

`true/false`

Definition at line 120 of file Vector.h.

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

- meowpp/math/Vector.h

6.58 meow::Vector2D< Scalar > Class Template Reference

2D's vector

```
#include "Vectors.h"
```

Public Member Functions

- Vector2D ()

    consturctor (0, 0)
- Vector2D (Vector2D const &v)

    consturctor (from another Vector2D)
- Vector2D (Scalar const &s)

    constructor (s, s)
- Vector2D (Scalar const &sx, Scalar const &sy)

    constructor (sx, sy)
- Vector2D (Vector< Scalar > const &v)

    constructor (from another Vector)
- Vector2D (Vector< Scalar > const &v, size_t i)

    constructor (from another Vector, i-th)
- ∼Vector2D ()

    destructor
- Vector2D & copyFrom (Vector2D const &v)

    copy
- Scalar const & x () const

    access x
- Scalar & xGet ()

    access x with non constant reference
- Scalar & yGet ()

    access y with non constant reference
- Scalar const & y () const

    access y
- Scalar const & x (Scalar const &s)

modify x

- Scalar const & y (Scalar const &s)

  modify y
- Vector2D & xy (Scalar const &sx, Scalar const &sy)

  modify x and y
- Vector2D positive () const

  return +(∗this)
- Vector2D negative () const

  return -(∗this)
- Vector2D right () const

  return count-clockwise rotate 90 degree of itself
- Vector2D add (Vector2D const &v) const

  return (∗this)+v
- Vector2D & added (Vector2D const &v)

  Let itself add v.
- Vector2D sub (Vector2D const &v) const

  return (∗this)-v
- Vector2D & subed (Vector2D const &v)

  Let itself substract v.
- Vector2D mul (Scalar const &s) const

  return (∗this)∗s , where s is a scalar
- Vector2D & muled (Scalar const &s)

  Let itself mulitple s.
- Vector2D div (Scalar const &s) const

  return (∗this)/s , where s is a scalar
- Vector2D & dived (Scalar const &s)

  Let itself divide s.
- Scalar mul (Vector2D const &v) const

  same as dot(v)
- Scalar dot (Vector2D const &v) const

  dot
- Scalar cross (Vector2D const &v) const

  cross
- Scalar length () const

  sqrt of length2
- Scalar length2 () const

  same as dot(∗this)
- Vector2D normalize () const

  return normalize form of itself
- Vector2D & normalized ()

  normalize itself
- Vector2D rotate (Scalar const &theta) const

  return rotate theta degree of itself
- Vector2D & rotated (Scalar const &theta)

  Let itself rotate theta degree.
- Vector2D reflect (Vector2D const &v) const

  return reflect from given vector v
- Vector2D & reflected (Vector2D const &v)

  reflect itself given vector v
- Matrix< Scalar > matrix () const

  return a 2x1 matrix form of itself

- Matrix< Scalar > matrix (Scalar const &homo) const

    return a 3x1 matrix form of itself
- Scalar const & operator() (size_t n) const
- Vector2D & operator() (Scalar const &sx, Scalar const &sy)
- Vector2D operator+ () const
- Vector2D operator- () const
- Vector2D operator∼ () const
- Vector2D operator+ (Vector2D const &v) const
- Vector2D operator- (Vector2D const &v) const
- Vector2D operator∗ (Scalar const &s) const
- Vector2D operator/ (Scalar const &s) const
- Scalar operator∗ (Vector2D const &v) const
- Vector2D & operator= (Vector2D const &v)
- Vector2D & operator+= (Vector2D const &v)
- Vector2D & operator-= (Vector2D const &v)
- Vector2D & operator∗= (Scalar const &s)
- Vector2D & operator/= (Scalar const &s)

6.58.1 Detailed Description

template<class Scalar>class meow::Vector2D< Scalar >

2D's vector

Author

cat_leopard

Definition at line 18 of file Vectors.h.

6.58.2 Constructor & Destructor Documentation

template<class Scalar> meow::Vector2D< Scalar >::Vector2D ( ) `[inline]`

consturctor (0, 0)
Definition at line 23 of file Vectors.h.

template<class Scalar> meow::Vector2D< Scalar >::Vector2D (Vector2D< Scalar > const & *v*) `[inline]`

consturctor (from another Vector2D)
Definition at line 27 of file Vectors.h.

template<class Scalar> meow::Vector2D< Scalar >::Vector2D (Scalar const & *s*) `[inline]`

constructor (s, s)
Definition at line 31 of file Vectors.h.

template<class Scalar> meow::Vector2D< Scalar >::Vector2D (Scalar const & *sx*, Scalar const & *sy*) `[inline]`

constructor (sx, sy)
Definition at line 35 of file Vectors.h.

template<class Scalar> meow::Vector2D< Scalar >::Vector2D (Vector< Scalar > const & *v*) `[inline]`

constructor (from another Vector)
Definition at line 39 of file Vectors.h.

template<class Scalar> meow::Vector2D< Scalar >::Vector2D (Vector< Scalar > const & *v*, size_t *i*) `[inline]`

constructor (from another Vector, i-th)

Definition at line 43 of file Vectors.h.

template<class Scalar> meow::Vector2D< Scalar >::~Vector2D ( ) `[inline]`

destructor

Definition at line 47 of file Vectors.h.

6.58.3 Member Function Documentation

template<class Scalar> Vector2D meow::Vector2D< Scalar >::add (Vector2D< Scalar > const & *v*) const `[inline]`

return (∗this)+v

Definition at line 110 of file Vectors.h.

template<class Scalar> Vector2D& meow::Vector2D< Scalar >::added (Vector2D< Scalar > const & *v*) `[inline]`

Let itself add v.

Definition at line 115 of file Vectors.h.

template<class Scalar> Vector2D& meow::Vector2D< Scalar >::copyFrom (Vector2D< Scalar > const & *v*) `[inline]`

copy

Definition at line 51 of file Vectors.h.

template<class Scalar> Scalar meow::Vector2D< Scalar >::cross (Vector2D< Scalar > const & *v*) const `[inline]`

cross

Definition at line 160 of file Vectors.h.

template<class Scalar> Vector2D meow::Vector2D< Scalar >::div (Scalar const & *s*) const `[inline]`

return (∗this)/s , where s is a scalar

Definition at line 140 of file Vectors.h.

template<class Scalar> Vector2D& meow::Vector2D< Scalar >::dived (Scalar const & *s*) `[inline]`

Let itself divide s.

Definition at line 145 of file Vectors.h.

template<class Scalar> Scalar meow::Vector2D< Scalar >::dot (Vector2D< Scalar > const & *v*) const `[inline]`

dot

Definition at line 155 of file Vectors.h.

template<class Scalar> Scalar meow::Vector2D< Scalar >::length ( ) const `[inline]`

sqrt of length2

Definition at line 165 of file Vectors.h.

template<class Scalar> Scalar meow::Vector2D< Scalar >::length2 ( ) const `[inline]`

same as *dot(∗this)*

Definition at line 170 of file Vectors.h.

template<class Scalar> Matrix<Scalar> meow::Vector2D< Scalar >::matrix ( ) const `[inline]`

return a 2x1 matrix form of itself

Definition at line 208 of file Vectors.h.

template<class Scalar> Matrix<Scalar> meow::Vector2D< Scalar >::matrix (Scalar const & *homo*) const `[inline]`

return a 3x1 matrix form of itself

Definition at line 216 of file Vectors.h.

template<class Scalar> Vector2D meow::Vector2D< Scalar >::mul (Scalar const & *s*) const `[inline]`

return (∗this)∗s , where s is a scalar

Definition at line 130 of file Vectors.h.

template<class Scalar> Scalar meow::Vector2D< Scalar >::mul (Vector2D< Scalar > const & *v*) const `[inline]`

same as dot(v)

Definition at line 150 of file Vectors.h.

template<class Scalar> Vector2D& meow::Vector2D< Scalar >::muled (Scalar const & *s*) `[inline]`

Let itself mulitple s.

Definition at line 135 of file Vectors.h.

template<class Scalar> Vector2D meow::Vector2D< Scalar >::negative ( ) const `[inline]`

return -(∗this)

Definition at line 100 of file Vectors.h.

template<class Scalar> Vector2D meow::Vector2D< Scalar >::normalize ( ) const `[inline]`

return normalize form of itself

Definition at line 175 of file Vectors.h.

template<class Scalar> Vector2D& meow::Vector2D< Scalar >::normalized ( ) `[inline]`

normalize itself

Definition at line 180 of file Vectors.h.

template<class Scalar> Scalar const& meow::Vector2D< Scalar >::operator() (size_t *n*) const `[inline]`

Definition at line 224 of file Vectors.h.

template<class Scalar> Vector2D& meow::Vector2D< Scalar >::operator() (Scalar const & *sx*, Scalar const & *sy*) `[inline]`

Definition at line 228 of file Vectors.h.

template<class Scalar> Vector2D meow::Vector2D< Scalar >::operator∗ (Scalar const & *s*) const `[inline]`

Definition at line 238 of file Vectors.h.

template<class Scalar> Scalar meow::Vector2D< Scalar >::operator∗ (Vector2D< Scalar > const & *v*) const `[inline]`

Definition at line 240 of file Vectors.h.

template<class Scalar> Vector2D& meow::Vector2D< Scalar >::operator∗= (Scalar const & *s*) `[inline]`

Definition at line 245 of file Vectors.h.

template<class Scalar> Vector2D meow::Vector2D< Scalar >::operator+ ( ) const `[inline]`

Definition at line 232 of file Vectors.h.

template<class Scalar> Vector2D meow::Vector2D< Scalar >::operator+ (Vector2D< Scalar > const & *v*) const `[inline]`

Definition at line 236 of file Vectors.h.

template<class Scalar> Vector2D& meow::Vector2D< Scalar >::operator+= (Vector2D< Scalar > const & *v*) `[inline]`

Definition at line 243 of file Vectors.h.

template<class Scalar> Vector2D meow::Vector2D< Scalar >::operator- ( ) const `[inline]`

Definition at line 233 of file Vectors.h.

template<class Scalar> Vector2D meow::Vector2D< Scalar >::operator- (Vector2D< Scalar > const & *v*) const `[inline]`

Definition at line 237 of file Vectors.h.

template<class Scalar> Vector2D& meow::Vector2D< Scalar >::operator-= (Vector2D< Scalar > const & *v*) `[inline]`

Definition at line 244 of file Vectors.h.

template<class Scalar> Vector2D meow::Vector2D< Scalar >::operator/ (Scalar const & *s*) const `[inline]`

Definition at line 239 of file Vectors.h.

template<class Scalar> Vector2D& meow::Vector2D< Scalar >::operator/= (Scalar const & *s*) `[inline]`

Definition at line 246 of file Vectors.h.

template<class Scalar> Vector2D& meow::Vector2D< Scalar >::operator= (Vector2D< Scalar > const & *v*) `[inline]`

Definition at line 242 of file Vectors.h.

template<class Scalar> Vector2D meow::Vector2D< Scalar >::operator∼ ( ) const `[inline]`

Definition at line 234 of file Vectors.h.

template<class Scalar> Vector2D meow::Vector2D< Scalar >::positive ( ) const `[inline]`

return +(∗this)

Definition at line 95 of file Vectors.h.

template<class Scalar> Vector2D meow::Vector2D< Scalar >::reflect (Vector2D< Scalar > const & *v*) const `[inline]`

return reflect from given vector *v*

Definition at line 198 of file Vectors.h.

template<class Scalar> Vector2D& meow::Vector2D< Scalar >::reflected (Vector2D< Scalar > const & *v*) `[inline]`

reflect itself given vector *v*

Definition at line 203 of file Vectors.h.

template<class Scalar> Vector2D meow::Vector2D< Scalar >::right ( ) const `[inline]`

return *count-clockwise rotate 90 degree* of itself

Definition at line 105 of file Vectors.h.

template<class Scalar> Vector2D meow::Vector2D< Scalar >::rotate (Scalar const & *theta*) const `[inline]`

return rotate *theta* degree of itself

Definition at line 185 of file Vectors.h.

template<class Scalar> Vector2D& meow::Vector2D< Scalar >::rotated (Scalar const & *theta*) `[inline]`

Let itself rotate *theta* degree.

Definition at line 193 of file Vectors.h.

template<class Scalar> Vector2D meow::Vector2D< Scalar >::sub (Vector2D< Scalar > const & *v*) const `[inline]`

return (∗this)-v

Definition at line 120 of file Vectors.h.

template<class Scalar> Vector2D& meow::Vector2D< Scalar >::subed (Vector2D< Scalar > const & *v*) `[inline]`

Let itself substract v.

Definition at line 125 of file Vectors.h.

template<class Scalar> Scalar const& meow::Vector2D< Scalar >::x ( ) const `[inline]`

access x

Definition at line 56 of file Vectors.h.

template<class Scalar> Scalar const& meow::Vector2D< Scalar >::x (Scalar const & *s*) `[inline]`

modify x

Definition at line 76 of file Vectors.h.

template<class Scalar> Scalar& meow::Vector2D< Scalar >::xGet ( ) `[inline]`

access x with non constant reference

Definition at line 61 of file Vectors.h.

template<class Scalar> Vector2D& meow::Vector2D< Scalar >::xy (Scalar const & *sx*, Scalar const & *sy*) `[inline]`

modify x and y

Definition at line 88 of file Vectors.h.

template<class Scalar> Scalar const& meow::Vector2D< Scalar >::y ( ) const `[inline]`

access y

Definition at line 71 of file Vectors.h.

template<class Scalar> Scalar const& meow::Vector2D< Scalar >::y (Scalar const & *s*) `[inline]`

modify y

Definition at line 82 of file Vectors.h.

template<class Scalar> Scalar& meow::Vector2D< Scalar >::yGet ( ) `[inline]`

access y with non constant reference

Definition at line 66 of file Vectors.h.

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

- meowpp/geo/Vectors.h

6.59 meow::Vector3D< Scalar > Class Template Reference

3D's vector

`#include "Vectors.h"`

Public Member Functions

- Vector3D ()

  consturctor (0, 0)
- Vector3D (Vector3D const &v)

  consturctor (from another Vector3D)
- Vector3D (Scalar const &s)

  constructor (s, s)
- Vector3D (Scalar const &sx, Scalar const &sy, Scalar const &sz)

  constructor (sx, sy)
- Vector3D (Vector< Scalar > const &v)

  constructor (from another Vector)
- Vector3D (Vector< Scalar > const &v, size_t i)

  constructor (from another Vector, i-th)
- ∼Vector3D ()

  destructor
- Vector3D & copyFrom (Vector3D const &v)

  copy
- Scalar const & x () const

  access x
- Scalar const & y () const

  access y
- Scalar const & z () const

  access z
- Scalar & xGet ()

  access x with non constant reference

- Scalar & yGet ()

  access y with non constant reference
- Scalar & zGet ()

  access z with non constant reference
- Scalar const & x (Scalar const &s)

  modify x
- Scalar const & y (Scalar const &s)

  modify y
- Scalar const & z (Scalar const &s)

  modify z
- Vector3D & xyz (Scalar const &sx, Scalar const &sy, Scalar const &sz)

  modify x and y
- Vector3D positive () const

  return +(∗this)
- Vector3D negative () const

  return -(∗this)
- Vector3D add (Vector3D const &v) const

  return (∗this)+v
- Vector3D & added (Vector3D const &v)

  Let itself add v.
- Vector3D sub (Vector3D const &v) const

  return (∗this)-v
- Vector3D & subed (Vector3D const &v)

  Let itself substract v.
- Vector3D mul (Scalar const &s) const

  return (∗this)∗s , where s is a scalar
- Vector3D & muled (Scalar const &s)

  Let itself mulitple s.
- Vector3D div (Scalar const &s) const

  return (∗this)/s , where s is a scalar
- Vector3D & dived (Scalar const &s)

  Let itself divide s.
- Scalar mul (Vector3D const &v) const

  same as dot(v)
- Scalar dot (Vector3D const &v) const

  dot
- Vector3D cross (Vector3D const &v) const

  cross
- Vector3D & crossed (Vector3D const &v)

  crossed
- Scalar length () const

  sqrt of length2
- Scalar length2 () const

  same as dot(∗this)
- Vector3D normalize () const

  return normalize form of itself
- Vector3D & normalized ()

  normalize itself
- Vector3D rotate (Vector3D const &axis, double theta) const

  return rotate theta degree by axis of itself
- Vector3D & rotated (Vector3D const &axis, double theta)

Let itself rotate theta degree.
- Vector3D reflect (Vector3D const &v) const

  return reflect from given vector v
- Vector3D & reflected (Vector3D const &v)

  reflect itself given vector v
- Matrix< Scalar > matrix () const

  return a 3x1 matrix form of itself
- Matrix< Scalar > matrix (Scalar const &homo) const

  return a 3x1 matrix form of itself
- Scalar const & operator() (size_t n) const
- Vector3D & operator() (Scalar const &sx, Scalar const &sy, Scalar const &sz)
- Vector3D operator+ () const
- Vector3D operator- () const
- Vector3D operator+ (Vector3D const &v) const
- Vector3D operator- (Vector3D const &v) const
- Vector3D operator∗ (Scalar const &s) const
- Vector3D operator/ (Scalar const &s) const
- Scalar operator∗ (Vector3D const &v) const
- Vector3D & operator= (Vector3D const &v)
- Vector3D & operator+= (Vector3D const &v)
- Vector3D & operator-= (Vector3D const &v)
- Vector3D & operator∗= (Scalar const &s)
- Vector3D & operator/= (Scalar const &s)

6.59.1 Detailed Description

template<class Scalar>class meow::Vector3D< Scalar >

3D's vector

Author

cat_leopard

Definition at line 255 of file Vectors.h.

6.59.2 Constructor & Destructor Documentation

template<class Scalar> meow::Vector3D< Scalar >::Vector3D ( ) `[inline]`

consturctor (0, 0)
Definition at line 260 of file Vectors.h.

template<class Scalar> meow::Vector3D< Scalar >::Vector3D (Vector3D< Scalar > const & *v*) `[inline]`

consturctor (from another Vector3D)
Definition at line 264 of file Vectors.h.

template<class Scalar> meow::Vector3D< Scalar >::Vector3D (Scalar const & *s*) `[inline]`

constructor (s, s)
Definition at line 268 of file Vectors.h.

template<class Scalar> meow::Vector3D< Scalar >::Vector3D (Scalar const & *sx,* Scalar const & *sy,* Scalar const & *sz*) `[inline]`

constructor (sx, sy)
Definition at line 272 of file Vectors.h.

template<class Scalar> meow::Vector3D< Scalar >::Vector3D (Vector< Scalar > const & *v*) `[inline]`

constructor (from another Vector)

Definition at line 278 of file Vectors.h.

template<class Scalar> meow::Vector3D< Scalar >::Vector3D (Vector< Scalar > const & *v*, size_t *i*) `[inline]`

constructor (from another Vector, i-th)

Definition at line 282 of file Vectors.h.

template<class Scalar> meow::Vector3D< Scalar >::∼Vector3D ( ) `[inline]`

destructor

Definition at line 286 of file Vectors.h.

6.59.3 Member Function Documentation

template<class Scalar> Vector3D meow::Vector3D< Scalar >::add (Vector3D< Scalar > const & *v*) const `[inline]`

return (∗this)+v

Definition at line 361 of file Vectors.h.

template<class Scalar> Vector3D& meow::Vector3D< Scalar >::added (Vector3D< Scalar > const & *v*) `[inline]`

Let itself add v.

Definition at line 366 of file Vectors.h.

template<class Scalar> Vector3D& meow::Vector3D< Scalar >::copyFrom (Vector3D< Scalar > const & *v*) `[inline]`

copy

Definition at line 290 of file Vectors.h.

template<class Scalar> Vector3D meow::Vector3D< Scalar >::cross (Vector3D< Scalar > const & *v*) const `[inline]`

cross

Definition at line 411 of file Vectors.h.

template<class Scalar> Vector3D& meow::Vector3D< Scalar >::crossed (Vector3D< Scalar > const & *v*) `[inline]`

crossed

Definition at line 418 of file Vectors.h.

template<class Scalar> Vector3D meow::Vector3D< Scalar >::div (Scalar const & *s*) const `[inline]`

return (∗this)/s , where s is a scalar

Definition at line 391 of file Vectors.h.

template<class Scalar> Vector3D& meow::Vector3D< Scalar >::dived (Scalar const & *s*) `[inline]`

Let itself divide s.

Definition at line 396 of file Vectors.h.

template<class Scalar> Scalar meow::Vector3D< Scalar >::dot (Vector3D< Scalar > const & *v*) const `[inline]`

dot

Definition at line 406 of file Vectors.h.

template<class Scalar> Scalar meow::Vector3D< Scalar >::length ( ) const `[inline]`

sqrt of length2

Definition at line 423 of file Vectors.h.

template<class Scalar> Scalar meow::Vector3D< Scalar >::length2 ( ) const `[inline]`

same as *dot(∗this)*

Definition at line 428 of file Vectors.h.

template<class Scalar> Matrix<Scalar> meow::Vector3D< Scalar >::matrix ( ) const `[inline]`

return a 3x1 matrix form of itself

Definition at line 466 of file Vectors.h.

template<class Scalar> Matrix<Scalar> meow::Vector3D< Scalar >::matrix (Scalar const & *homo*) const `[inline]`

return a 3x1 matrix form of itself

Definition at line 475 of file Vectors.h.

template<class Scalar> Vector3D meow::Vector3D< Scalar >::mul (Scalar const & *s*) const `[inline]`

return (∗this)∗s , where s is a scalar

Definition at line 381 of file Vectors.h.

template<class Scalar> Scalar meow::Vector3D< Scalar >::mul (Vector3D< Scalar > const & *v*) const `[inline]`

same as dot(v)

Definition at line 401 of file Vectors.h.

template<class Scalar> Vector3D& meow::Vector3D< Scalar >::muled (Scalar const & *s*) `[inline]`

Let itself mulitple s.

Definition at line 386 of file Vectors.h.

template<class Scalar> Vector3D meow::Vector3D< Scalar >::negative ( ) const `[inline]`

return -(∗this)

Definition at line 356 of file Vectors.h.

template<class Scalar> Vector3D meow::Vector3D< Scalar >::normalize ( ) const `[inline]`

return normalize form of itself

Definition at line 433 of file Vectors.h.

template<class Scalar> Vector3D& meow::Vector3D< Scalar >::normalized ( ) `[inline]`

normalize itself

Definition at line 438 of file Vectors.h.

template<class Scalar> Scalar const& meow::Vector3D< Scalar >::operator() (size_t *n*) const `[inline]`

Definition at line 484 of file Vectors.h.

template<class Scalar> Vector3D& meow::Vector3D< Scalar >::operator() (Scalar const & *sx*, Scalar const & *sy*, Scalar const & *sz*) `[inline]`

Definition at line 488 of file Vectors.h.

template<class Scalar> Vector3D meow::Vector3D< Scalar >::operator* (Scalar const & *s*) const `[inline]`

Definition at line 497 of file Vectors.h.

template<class Scalar> Scalar meow::Vector3D< Scalar >::operator* (Vector3D< Scalar > const & *v*) const `[inline]`

Definition at line 499 of file Vectors.h.

template<class Scalar> Vector3D& meow::Vector3D< Scalar >::operator*= (Scalar const & *s*) `[inline]`

Definition at line 504 of file Vectors.h.

template<class Scalar> Vector3D meow::Vector3D< Scalar >::operator+ ( ) const `[inline]`

Definition at line 492 of file Vectors.h.

template<class Scalar> Vector3D meow::Vector3D< Scalar >::operator+ (Vector3D< Scalar > const & *v*) const `[inline]`

Definition at line 495 of file Vectors.h.

template<class Scalar> Vector3D& meow::Vector3D< Scalar >::operator+= (Vector3D< Scalar > const & *v*) `[inline]`

Definition at line 502 of file Vectors.h.

template<class Scalar> Vector3D meow::Vector3D< Scalar >::operator- ( ) const `[inline]`

Definition at line 493 of file Vectors.h.

template<class Scalar> Vector3D meow::Vector3D< Scalar >::operator- (Vector3D< Scalar > const & *v*) const `[inline]`

Definition at line 496 of file Vectors.h.

template<class Scalar> Vector3D& meow::Vector3D< Scalar >::operator-= (Vector3D< Scalar > const & *v*) `[inline]`

Definition at line 503 of file Vectors.h.

template<class Scalar> Vector3D meow::Vector3D< Scalar >::operator/ (Scalar const & *s*) const `[inline]`

Definition at line 498 of file Vectors.h.

template<class Scalar> Vector3D& meow::Vector3D< Scalar >::operator/= (Scalar const & *s*) `[inline]`

Definition at line 505 of file Vectors.h.

template<class Scalar> Vector3D& meow::Vector3D< Scalar >::operator= (Vector3D< Scalar > const & *v*) `[inline]`

Definition at line 501 of file Vectors.h.

template<class Scalar> Vector3D meow::Vector3D< Scalar >::positive ( ) const `[inline]`

return +(∗this)
Definition at line 351 of file Vectors.h.

template<class Scalar> Vector3D meow::Vector3D< Scalar >::reflect (Vector3D< Scalar > const & *v*) const `[inline]`

return reflect from given vector *v*
Definition at line 456 of file Vectors.h.

template<class Scalar> Vector3D& meow::Vector3D< Scalar >::reflected (Vector3D< Scalar > const & *v*) `[inline]`

reflect itself given vector *v*
Definition at line 461 of file Vectors.h.

template<class Scalar> Vector3D meow::Vector3D< Scalar >::rotate (Vector3D< Scalar > const & *axis,* double *theta*) const `[inline]`

return rotate *theta* degree by *axis* of itself
Definition at line 443 of file Vectors.h.

template<class Scalar> Vector3D& meow::Vector3D< Scalar >::rotated (Vector3D< Scalar > const & *axis,* double *theta*) `[inline]`

Let itself rotate *theta* degree.
Definition at line 451 of file Vectors.h.

template<class Scalar> Vector3D meow::Vector3D< Scalar >::sub (Vector3D< Scalar > const & *v*) const `[inline]`

return (∗this)-v
Definition at line 371 of file Vectors.h.

template<class Scalar> Vector3D& meow::Vector3D< Scalar >::subed (Vector3D< Scalar > const & *v*) `[inline]`

Let itself substract v.
Definition at line 376 of file Vectors.h.

template<class Scalar> Scalar const& meow::Vector3D< Scalar >::x ( ) const `[inline]`

access x
Definition at line 295 of file Vectors.h.

template<class Scalar> Scalar const& meow::Vector3D< Scalar >::x (Scalar const & *s*) `[inline]`

modify x
Definition at line 325 of file Vectors.h.

template< class Scalar> Scalar& meow::Vector3D< Scalar >::xGet ( ) `[inline]`

access x with non constant reference

Definition at line 310 of file Vectors.h.

template< class Scalar> Vector3D& meow::Vector3D< Scalar >::xyz (Scalar const & *sx*, Scalar const & *sy*, Scalar const & *sz*) `[inline]`

modify x and y

Definition at line 343 of file Vectors.h.

template< class Scalar> Scalar const& meow::Vector3D< Scalar >::y ( ) const `[inline]`

access y

Definition at line 300 of file Vectors.h.

template< class Scalar> Scalar const& meow::Vector3D< Scalar >::y (Scalar const & *s*) `[inline]`

modify y

Definition at line 331 of file Vectors.h.

template< class Scalar> Scalar& meow::Vector3D< Scalar >::yGet ( ) `[inline]`

access y with non constant reference

Definition at line 315 of file Vectors.h.

template< class Scalar> Scalar const& meow::Vector3D< Scalar >::z ( ) const `[inline]`

access z

Definition at line 305 of file Vectors.h.

template< class Scalar> Scalar const& meow::Vector3D< Scalar >::z (Scalar const & *s*) `[inline]`

modify z

Definition at line 337 of file Vectors.h.

template< class Scalar> Scalar& meow::Vector3D< Scalar >::zGet ( ) `[inline]`

access z with non constant reference

Definition at line 320 of file Vectors.h.

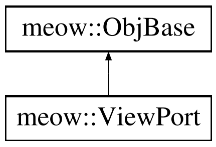
The documentation for this class was generated from the following file:

- meowpp/geo/Vectors.h

6.60 meow::ViewPort Class Reference

`#include "ViewPort.h"`

Inheritance diagram for meow::ViewPort:



Additional Inherited Members

6.60.1 Detailed Description

Definition at line 11 of file ViewPort.h.

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

- meowpp/gra/ViewPort.h

6.61 meow::VP_Tree< Vector, Scalar > Class Template Reference

KD_Tree

`#include "VP_Tree.h"`

Public Types

- typedef std::vector< Vector > Vectors

Public Member Functions

- VP_Tree ()

  constructor, with dimension = 1
- VP_Tree (VP_Tree const &tree2)

  constructor,
- VP_Tree (size_t dimension)

  constructor, dimension
- ∼VP_Tree ()

  destructor
- VP_Tree & copyFrom (VP_Tree const &tree2)
- void insert (Vector const &vector)

  Vectorset
- bool erase (Vector const &vector)

  Vectorset
- void build ()

  insert/erase `rebuild()`
- void forceBuild ()
- Vectors query (Vector const &vector, size_t nearestNumber, bool compareWholeVector) const
- void clear ()
- size_t reset (size_t dimension)
- VP_Tree & operator= (VP_Tree const &tree2)

  same as `copyFrom(tree2)`

6.61.1 Detailed Description

template<class Vector, class Scalar>class meow::VP_Tree< Vector, Scalar >

KD_Tree

`VP_Tree` **NK** , set **i**∗ . `KD_Tree` , , `VP_Tree` , , ..., , random
:

- `http://stevehanov.ca/blog/index.php?id=130`
- `http://pnylab.com/pny/papers/vptree/vptree`

Template Class Operators Request

const?	Typename	Operator	Parameters	Return Type	Description
const	Vector	operator[]	(size_t n)	Scalar	n
const	Vector	operator=	(Vector v)	Vector&	copy operator
const	Vector	operator<	(Vector v)	bool	
const	Scalar	'Scalar'	(int n)	Scalar	,

n=0or4 | |const | Scalar|operator* |(Scalar s) | Scalar | | |const | Scalar|operator+ |(Scalar s) | Scalar | | |const | Scalar|operator- |(Scalar s) | Scalar | | |const | Scalar|operator- |( ) | Scalar | | |const | Scalar|operator< |(Scalar s) | bool | |

Note

: -, , KD_Tree, VP_Tree **random** , O(logN) , KD_Tree , VP_Tree -TODO insert(), erase()

Definition at line 51 of file VP_Tree.h.

6.61.2 Member Typedef Documentation

template<class Vector, class Scalar> typedef std::vector<Vector> meow::VP_Tree< Vector, Scalar >::Vectors

Definition at line 53 of file VP_Tree.h.

6.61.3 Constructor & Destructor Documentation

template<class Vector, class Scalar> meow::VP_Tree< Vector, Scalar >::VP_Tree ( ) [inline]

constructor, with dimension = 1
Definition at line 212 of file VP_Tree.h.

template<class Vector, class Scalar> meow::VP_Tree< Vector, Scalar >::VP_Tree (VP_Tree< Vector, Scalar > const & *tree2*) [inline]

constructor,
Definition at line 217 of file VP_Tree.h.

template<class Vector, class Scalar> meow::VP_Tree< Vector, Scalar >::VP_Tree (size_t *dimension*) [inline]

constructor, dimension
Definition at line 225 of file VP_Tree.h.

template<class Vector, class Scalar> meow::VP_Tree< Vector, Scalar >::~VP_Tree ( ) [inline]

destructor
Definition at line 234 of file VP_Tree.h.

6.61.4 Member Function Documentation

template<class Vector, class Scalar> void meow::VP_Tree< Vector, Scalar >::build ( ) [inline]

insert/erase rebuild()
Definition at line 275 of file VP_Tree.h.

template<class Vector, class Scalar> void meow::VP_Tree< Vector, Scalar >::clear ( ) [inline]

Definition at line 313 of file VP_Tree.h.

template<class Vector, class Scalar> VP_Tree& meow::VP_Tree< Vector, Scalar >::copyFrom (VP_Tree< Vector, Scalar > const & *tree2*) [inline]

Definition at line 241 of file VP_Tree.h.

template<class Vector, class Scalar> bool meow::VP_Tree< Vector, Scalar >::erase (Vector const & *vector*) `[inline]`

Vectorset

Definition at line 260 of file VP_Tree.h.

template<class Vector, class Scalar> void meow::VP_Tree< Vector, Scalar >::forceBuild ( ) `[inline]`

Definition at line 284 of file VP_Tree.h.

template<class Vector, class Scalar> void meow::VP_Tree< Vector, Scalar >::insert (Vector const & *vector*) `[inline]`

Vectorset

Definition at line 252 of file VP_Tree.h.

template<class Vector, class Scalar> VP_Tree& meow::VP_Tree< Vector, Scalar >::operator= (VP_Tree< Vector, Scalar > const & *tree2*) `[inline]`

same as `copyFrom(tree2)`

Definition at line 330 of file VP_Tree.h.

template<class Vector, class Scalar> Vectors meow::VP_Tree< Vector, Scalar >::query (Vector const & *vector,* size_t *nearestNumber,* bool *compareWholeVector*) const `[inline]`

set `i` , . `v1`,v2 , `cmp` `true` , `v1<v2` . .

Definition at line 296 of file VP_Tree.h.

template<class Vector, class Scalar> size_t meow::VP_Tree< Vector, Scalar >::reset (size_t *dimension*) `[inline]`

Definition at line 323 of file VP_Tree.h.

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

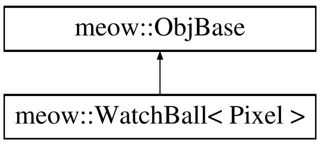
- meowpp/dsa/VP_Tree.h

6.62 meow::WatchBall< Pixel > Class Template Reference

camera, offset, rotation

`#include "WatchBall.h"`

Inheritance diagram for meow::WatchBall< Pixel >:



Public Types

- typedef std::vector< Camera < Pixel > > Cameras

Public Member Functions

- WatchBall ()

    constructor
- WatchBall (WatchBall const &b)

    copy constructor
- ∼WatchBall ()

    destructor
- WatchBall & copyFrom (WatchBall const &b)

    copy data
- WatchBall & referenceFrom (WatchBall const &b)

    reference
- size_t cameraSize () const

    camera
- Cameras const & cameras () const

    cameras
- Cameras & camerasGet ()

    cameras (non-constant)
- Cameras const & cameras (Cameras const &c)

    camera
- Camera< Pixel > const & camera (size_t i) const

    icamera
- Camera< Pixel > & camera (size_t i)

    icamera (non-constant reference)
- Camera< Pixel > const & camera (size_t i, Camera< Pixel > const &c)

    icamera
- Vector3D< double > const & offset () const

    offset
- Vector3D< double > & offset ()

    offset (non-constant reference)
- Vector3D< double > const & offset (Vector3D< double > const &ofs)

    offset
- Pixel color (Vector3D< double > p) const

    color
- Bitmap< Pixel > expand (double radius) const

- WatchBall & operator= (WatchBall const &b)

    same as `copyFrom(b)`
- bool write (FILE ∗f, bool bin, unsigned int fg) const

- bool read (FILE ∗f, bool bin, unsigned int fg)

- ObjBase ∗ create () const

    new
- ObjBase ∗ copyFrom (ObjBase const ∗b)

- char const ∗ ctype () const

    classtype
- std::string type () const

    classtype

Additional Inherited Members

6.62.1 Detailed Description

template<class Pixel>class meow::WatchBall< Pixel >

camera, offset, rotation

Author

cat_leopard

Definition at line 22 of file WatchBall.h.

6.62.2 Member Typedef Documentation

template<class Pixel > typedef std::vector<Camera<Pixel> > meow::WatchBall< Pixel >::Cameras

Definition at line 24 of file WatchBall.h.

6.62.3 Constructor & Destructor Documentation

template<class Pixel > meow::WatchBall< Pixel >::WatchBall ( ) `[inline]`

constructor
Definition at line 46 of file WatchBall.h.

template<class Pixel > meow::WatchBall< Pixel >::WatchBall (WatchBall< Pixel > const & *b*) `[inline]`

copy constructor
Definition at line 52 of file WatchBall.h.

template<class Pixel > meow::WatchBall< Pixel >::~WatchBall ( ) `[inline]`

destructor
Definition at line 58 of file WatchBall.h.

6.62.4 Member Function Documentation

template<class Pixel > Camera<Pixel> const& meow::WatchBall< Pixel >::camera (size_t *i*) const `[inline]`

icamera
Definition at line 109 of file WatchBall.h.

template<class Pixel > Camera<Pixel>& meow::WatchBall< Pixel >::camera (size_t *i*) `[inline]`

icamera (non-constant reference)
Definition at line 116 of file WatchBall.h.

template<class Pixel > Camera<Pixel> const& meow::WatchBall< Pixel >::camera (size_t *i*, Camera< Pixel > const & *c*) `[inline]`

icamera
Definition at line 123 of file WatchBall.h.

template<class Pixel > Cameras const& meow::WatchBall< Pixel >::cameras ( ) const `[inline]`

cameras
Definition at line 87 of file WatchBall.h.

template< class Pixel > Cameras const& meow::WatchBall< Pixel >::cameras (Cameras const & *c*) [inline]

camera

Definition at line 101 of file WatchBall.h.

template< class Pixel > Cameras& meow::WatchBall< Pixel >::camerasGet ( ) [inline]

cameras (non-constant)

Definition at line 94 of file WatchBall.h.

template< class Pixel > size_t meow::WatchBall< Pixel >::cameraSize ( ) const [inline]

camera

Definition at line 80 of file WatchBall.h.

template< class Pixel > Pixel meow::WatchBall< Pixel >::color (Vector3D< double > *p*) const [inline]

color

Definition at line 153 of file WatchBall.h.

template< class Pixel > WatchBall& meow::WatchBall< Pixel >::copyFrom (WatchBall< Pixel > const & *b*) [inline]

copy data

Definition at line 64 of file WatchBall.h.

template< class Pixel > ObjBase* meow::WatchBall< Pixel >::copyFrom (ObjBase const * *b*) [inline], [virtual]

`ObjBase const*` method`copyFrom`

Parameters

in	*b*	

Returns

this

Reimplemented from meow::ObjBase.

Definition at line 230 of file WatchBall.h.

template< class Pixel > ObjBase* meow::WatchBall< Pixel >::create ( ) const [inline], [virtual]

new

Returns

newpointer

Reimplemented from meow::ObjBase.

Definition at line 217 of file WatchBall.h.

template< class Pixel > char const* meow::WatchBall< Pixel >::ctype ( ) const [inline], [virtual]

classtype

Returns

`char const* typename`

Reimplemented from meow::ObjBase.

Definition at line 238 of file WatchBall.h.

template<class Pixel > Bitmap<Pixel> meow::WatchBall< Pixel >::expand (double *radius*) const `[inline]`

Parameters

`in`	*radius*	

Definition at line 171 of file WatchBall.h.

template<class Pixel > Vector3D<double> const& meow::WatchBall< Pixel >::offset ( ) const `[inline]`

offset

Definition at line 131 of file WatchBall.h.

template<class Pixel > Vector3D<double>& meow::WatchBall< Pixel >::offset ( ) `[inline]`

offset (non-constant reference)

Definition at line 138 of file WatchBall.h.

template<class Pixel > Vector3D<double> const& meow::WatchBall< Pixel >::offset (Vector3D< double > const & *ofs*) `[inline]`

offset

Definition at line 145 of file WatchBall.h.

template<class Pixel > WatchBall& meow::WatchBall< Pixel >::operator= (WatchBall< Pixel > const & *b*) `[inline]`

same as `copyFrom(b)`

Definition at line 193 of file WatchBall.h.

template<class Pixel > bool meow::WatchBall< Pixel >::read (FILE ∗ *f*, bool *bin*, unsigned int *fg*) `[inline]`, `[virtual]`

Note

Reimplemented from meow::ObjBase.

Definition at line 209 of file WatchBall.h.

template<class Pixel > WatchBall& meow::WatchBall< Pixel >::referenceFrom (WatchBall< Pixel > const & *b*) `[inline]`

reference

Definition at line 72 of file WatchBall.h.

template<class Pixel > std::string meow::WatchBall< Pixel >::type ( ) const `[inline]`, `[virtual]`

classtype

Returns

`std::string` typename

Reimplemented from meow::ObjBase.

Definition at line 247 of file WatchBall.h.

template<class Pixel > bool meow::WatchBall< Pixel >::write (FILE ∗ *f*, bool *bin*, unsigned int *fg*) const `[inline]`, `[virtual]`

Note

Reimplemented from meow::ObjBase.
Definition at line 201 of file WatchBall.h.
The documentation for this class was generated from the following file:

- meowpp/gra/WatchBall.h

6.63 meow::YUV< T > Class Template Reference

`#include "YUV.h"`

Public Member Functions

- virtual ∼YUV ()
- virtual T yMax () const =0
- virtual T yMin () const =0
- virtual T uMax () const =0
- virtual T uMin () const =0
- virtual T vMax () const =0
- virtual T vMin () const =0
- T y () const
- T u () const
- T v () const
- T yuv (size_t i) const
- T vuy (size_t i) const
- T y (T const &val)
- T u (T const &val)
- T v (T const &val)
- T yuv (size_t i, T const &val)
- T vuy (size_t i, T const &val)

Protected Member Functions

- YUV ()
- YUV (T const &y, T const &u, T const &v)
- YUV (T const ∗yuv)

Protected Attributes

- T yuv_ [3]

6.63.1 Detailed Description

template<class T>class meow::YUV< T >

Definition at line 7 of file YUV.h.

6.63.2 Constructor & Destructor Documentation

template<class T > meow::YUV< T >::YUV ( ) `[inline], [protected]`

Definition at line 6 of file YUV.hpp.

template<class T> meow::YUV< T >::YUV (T const & *y*, T const & *u*, T const & *v*) `[inline]`, `[protected]`

Definition at line 7 of file YUV.hpp.

template<class T> meow::YUV< T >::YUV (T const ∗ *yuv*) `[inline]`, `[protected]`

Definition at line 10 of file YUV.hpp.

template<class T> virtual meow::YUV< T >::∼YUV ( ) `[inline]`, `[virtual]`

Definition at line 14 of file YUV.h.

6.63.3 Member Function Documentation

template<class T > T meow::YUV< T >::u ( ) const `[inline]`

Definition at line 17 of file YUV.hpp.

template<class T> T meow::YUV< T >::u (T const & *val*) `[inline]`

Definition at line 25 of file YUV.hpp.

template<class T> virtual T meow::YUV< T >::uMax ( ) const `[pure virtual]`

Implemented in meow::YUVf.

template<class T> virtual T meow::YUV< T >::uMin ( ) const `[pure virtual]`

Implemented in meow::YUVf.

template<class T > T meow::YUV< T >::v ( ) const `[inline]`

Definition at line 18 of file YUV.hpp.

template<class T> T meow::YUV< T >::v (T const & *val*) `[inline]`

Definition at line 26 of file YUV.hpp.

template<class T> virtual T meow::YUV< T >::vMax ( ) const `[pure virtual]`

Implemented in meow::YUVf.

template<class T> virtual T meow::YUV< T >::vMin ( ) const `[pure virtual]`

Implemented in meow::YUVf.

template<class T > T meow::YUV< T >::vuy (size_t *i*) const `[inline]`

Definition at line 22 of file YUV.hpp.

template<class T> T meow::YUV< T >::vuy (size_t *i*, T const & *val*) `[inline]`

Definition at line 31 of file YUV.hpp.

template<class T > T meow::YUV< T >::y ( ) const `[inline]`

Definition at line 16 of file YUV.hpp.

template<class T> T meow::YUV< T >::y (T const & *val*) `[inline]`

Definition at line 24 of file YUV.hpp.

template<class T> virtual T meow::YUV< T >::yMax ( ) const `[pure virtual]`

Implemented in meow::YUVf.

template<class T> virtual T meow::YUV< T >::yMin ( ) const `[pure virtual]`

Implemented in meow::YUVf.

template<class T > T meow::YUV< T >::yuv (size_t *i*) const `[inline]`

Definition at line 19 of file YUV.hpp.

template<class T> T meow::YUV< T >::yuv (size_t *i*, T const & *val*) `[inline]`

Definition at line 27 of file YUV.hpp.

6.63.4 Member Data Documentation

template<class T> T meow::YUV< T >::yuv_[3] `[protected]`

Definition at line 9 of file YUV.h.

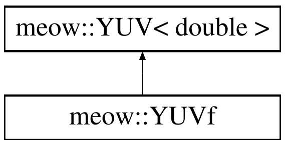
The documentation for this class was generated from the following files:

- meowpp/colors/YUV.h
- meowpp/colors/YUV.hpp

6.64 meow::YUVf Class Reference

`#include "YUV.h"`

Inheritance diagram for meow::YUVf:



Public Member Functions

- YUVf ()
- ∼YUVf ()
- YUVf (double const &y, double const &u, double const &v)
- YUVf (double const ∗yuv)
- double yMin () const
- double yMax () const
- double uMin () const
- double uMax () const
- double vMin () const
- double vMax () const

Additional Inherited Members

6.64.1 Detailed Description

Definition at line 36 of file YUV.h.

6.64.2 Constructor & Destructor Documentation

meow::YUVf::YUVf ( ) `[inline]`

Definition at line 35 of file YUV.hpp.

meow::YUVf::∼YUVf ( ) `[inline]`

Definition at line 36 of file YUV.hpp.

meow::YUVf::YUVf (double const & *y*, double const & *u*, double const & *v*) `[inline]`

Definition at line 37 of file YUV.hpp.

meow::YUVf::YUVf (double const ∗ *yuv*) `[inline]`

Definition at line 38 of file YUV.hpp.

6.64.3 Member Function Documentation

double meow::YUVf::uMax ( ) const `[inline]`, `[virtual]`

Implements meow::YUV< double >.

Definition at line 42 of file YUV.hpp.

double meow::YUVf::uMin ( ) const `[inline]`, `[virtual]`

Implements meow::YUV< double >.

Definition at line 41 of file YUV.hpp.

double meow::YUVf::vMax ( ) const `[inline]`, `[virtual]`

Implements meow::YUV< double >.

Definition at line 44 of file YUV.hpp.

double meow::YUVf::vMin ( ) const `[inline]`, `[virtual]`

Implements meow::YUV< double >.

Definition at line 43 of file YUV.hpp.

double meow::YUVf::yMax ( ) const `[inline]`, `[virtual]`

Implements meow::YUV< double >.

Definition at line 40 of file YUV.hpp.

double meow::YUVf::yMin ( ) const `[inline]`, `[virtual]`

Implements meow::YUV< double >.

Definition at line 39 of file YUV.hpp.

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

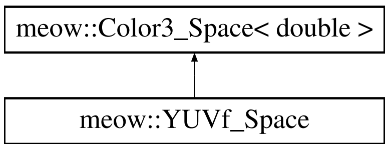
- meowpp/colors/YUV.h
- meowpp/colors/YUV.hpp

6.65 meow::YUVf_Space Class Reference

Y(), **U**(), **V**()

`#include "YUV_Space.h"`

Inheritance diagram for meow::YUVf_Space:



Public Member Functions

- YUVf_Space ()
- YUVf_Space (double c)
- YUVf_Space (Vector3D$<$ double $>$ const &v)
- YUVf_Space (YUV_Space const &b)
- ∼YUVf_Space ()
- double const & yuvMin (size_t i) const
- double const & yMin () const
- double const & uMin () const
- double const & vMin () const
- double const & yuvMax (size_t i) const
- double const & yMax () const
- double const & uMax () const
- double const & vMax () const
- double const & yuv (size_t i) const
- double const & y () const
- double const & u () const
- double const & v () const
- double const & yuv (size_t i, double c)
- double const & y (double c)
- double const & u (double c)
- double const & v (double c)
- double & yuvGet (size_t i)
- double & yGet ()
- double & uGet ()
- double & vGet ()
- YUVf_Space & operator= (YUVf_Space const &b)
- YUVf_Space operator+ (YUVf_Space const &b) const
- YUVf_Space operator- (YUVf_Space const &b) const
- YUVf_Space operator∗ (double const &c) const
- YUVf_Space operator/ (double const &c) const
- double operator∗ (YUVf_Space const &b) const

Additional Inherited Members

6.65.1 Detailed Description

Y(), **U**(), **V**()

0.0∼1.0

Author

cat_leopard

Definition at line 21 of file YUV_Space.h.

6.65.2 Constructor & Destructor Documentation

meow::YUVf_Space::YUVf_Space ( ) `[inline]`

Definition at line 23 of file YUV_Space.h.

meow::YUVf_Space::YUVf_Space (double *c*) `[inline]`

Definition at line 27 of file YUV_Space.h.

meow::YUVf_Space::YUVf_Space (Vector3D< double > const & *v*) `[inline]`

Definition at line 31 of file YUV_Space.h.

meow::YUVf_Space::YUVf_Space (YUV_Space const & *b*) `[inline]`

Definition at line 36 of file YUV_Space.h.

meow::YUVf_Space::∼YUVf_Space ( ) `[inline]`

Definition at line 38 of file YUV_Space.h.

6.65.3 Member Function Documentation

YUVf_Space meow::YUVf_Space::operator∗ (double const & *c*) const `[inline]`

Definition at line 70 of file YUV_Space.h.

double meow::YUVf_Space::operator∗ (YUVf_Space const & *b*) const `[inline]`

Definition at line 76 of file YUV_Space.h.

YUVf_Space meow::YUVf_Space::operator+ (YUVf_Space const & *b*) const `[inline]`

Definition at line 64 of file YUV_Space.h.

YUVf_Space meow::YUVf_Space::operator- (YUVf_Space const & *b*) const `[inline]`

Definition at line 67 of file YUV_Space.h.

YUVf_Space meow::YUVf_Space::operator/ (double const & *c*) const `[inline]`

Definition at line 73 of file YUV_Space.h.

YUVf_Space& meow::YUVf_Space::operator= (YUVf_Space const & *b*) `[inline]`

Definition at line 60 of file YUV_Space.h.

double const& meow::YUVf_Space::u ( ) const `[inline]`

Definition at line 50 of file YUV_Space.h.

double const& meow::YUVf_Space::u (double *c*) `[inline]`

Definition at line 54 of file YUV_Space.h.

double& meow::YUVf_Space::uGet ( ) `[inline]`

Definition at line 58 of file YUV_Space.h.

double const& meow::YUVf_Space::uMax ( ) const [inline]

Definition at line 46 of file YUV_Space.h.

double const& meow::YUVf_Space::uMin ( ) const [inline]

Definition at line 42 of file YUV_Space.h.

double const& meow::YUVf_Space::v ( ) const [inline]

Definition at line 51 of file YUV_Space.h.

double const& meow::YUVf_Space::v (double *c*) [inline]

Definition at line 55 of file YUV_Space.h.

double& meow::YUVf_Space::vGet ( ) [inline]

Definition at line 59 of file YUV_Space.h.

double const& meow::YUVf_Space::vMax ( ) const [inline]

Definition at line 47 of file YUV_Space.h.

double const& meow::YUVf_Space::vMin ( ) const [inline]

Definition at line 43 of file YUV_Space.h.

double const& meow::YUVf_Space::y ( ) const [inline]

Definition at line 49 of file YUV_Space.h.

double const& meow::YUVf_Space::y (double *c*) [inline]

Definition at line 53 of file YUV_Space.h.

double& meow::YUVf_Space::yGet ( ) [inline]

Definition at line 57 of file YUV_Space.h.

double const& meow::YUVf_Space::yMax ( ) const [inline]

Definition at line 45 of file YUV_Space.h.

double const& meow::YUVf_Space::yMin ( ) const [inline]

Definition at line 41 of file YUV_Space.h.

double const& meow::YUVf_Space::yuv (size_t *i*) const [inline]

Definition at line 48 of file YUV_Space.h.

double const& meow::YUVf_Space::yuv (size_t *i*, double *c*) [inline]

Definition at line 52 of file YUV_Space.h.

double& meow::YUVf_Space::yuvGet (size_t *i*) [inline]

Definition at line 56 of file YUV_Space.h.

double const& meow::YUVf_Space::yuvMax (size_t *i*) const `[inline]`

Definition at line 44 of file YUV_Space.h.

double const& meow::YUVf_Space::yuvMin (size_t *i*) const `[inline]`

Definition at line 40 of file YUV_Space.h.

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

- meowpp/colors/YUV_Space.h

Chapter 7

File Documentation

7.1 meowpp/colors/Color3_Space.h File Reference

```
#include "../geo/Vectors.h"
#include "../math/Matrix.h"
#include "../math/utility.h"
#include <cstdlib>
```

Classes

- class meow::Color3_Space< T >

  channel

Namespaces

- meow

7.2 meowpp/colors/HSL.h File Reference

```
#include "RGB.h"
#include "YUV.h"
#include "HSL.hpp"
```

Classes

- class meow::HSL< T >
- class meow::HSLf

Namespaces

- meow

Functions

- template<class RGB_T , class HSL_T >
  void meow::RGB_to_HSL (RGB< RGB_T > const &rgb, HSL< HSL_T > *hsl)
- template<class HSL_T , class RGB_T >
  void meow::HSL_to_RGB (HSL< HSL_T > const &hsl, RGB< RGB_T > *rgb)
- template<class YUV_T , class HSL_T >
  void meow::YUV_to_HSL (YUV< YUV_T > const &yuv, HSL< HSL_T > *hsl)

- template<class HSL_T , class YUV_T >
  void meow::HSL_to_YUV (HSL< HSL_T > const &hsl, YUV< YUV_T > *yuv)

7.3 meowpp/colors/HSL.hpp File Reference

```
#include "HSL.h"
#include "RGB.h"
#include "YUV.h"
#include "../utility.h"
```

Namespaces

- meow

Functions

- template<class RGB_T , class HSL_T >
  void meow::RGB_to_HSL (RGB< RGB_T > const &rgb, HSL< HSL_T > *hsl)
- template<class HSL_T , class RGB_T >
  void meow::HSL_to_RGB (HSL< HSL_T > const &hsl, RGB< RGB_T > *rgb)
- template<class YUV_T , class HSL_T >
  void meow::YUV_to_HSL (YUV< YUV_T > const &yuv, HSL< HSL_T > *hsl)
- template<class HSL_T , class YUV_T >
  void meow::HSL_to_YUV (HSL< HSL_T > const &hsl, YUV< YUV_T > *yuv)

7.4 meowpp/colors/HSL_Space.h File Reference

```
#include "Color3_Space.h"
#include "RGB_Space.h"
#include "YUV_Space.h"
#include "../geo/Vectors.h"
#include "../math/utility.h"
#include <cstdlib>
```

Classes

- class meow::HSLf_Space

  ***Y**(), **U**(), **V**()*

Namespaces

- meow

Functions

- void meow::colorTransformate (RGBf_Space const &rgb, HSLf_Space *hsl)

  `RGBf_Space` to `HSLf_Space`
- void meow::colorTransformate (YUVf_Space const &yuv, HSLf_Space *hsl)

  `YUVf_Space` to `HSLf_Space`
- void meow::colorTransformate (HSLf_Space const &hsl, RGBf_Space *rgb)

  `HSLf_Space` to `RGBf_Space`
- void meow::colorTransformate (HSLf_Space const &hsl, YUVf_Space *yuv)

  `HSLf_Space` to `YUVf_Space`

- void meow::colorTransformate (HSLf_Space const &hsl, RGBi_Space ∗rgb)

  HSLf_Space to RGBi_Space
- void meow::colorTransformate (RGBi_Space const &rgb, HSLf_Space ∗hsl)

  RGBi_Space to HSLf_Space

7.5 meowpp/colors/HSV.h File Reference

```
#include "RGB.h"
#include "YUV.h"
#include "HSL.h"
#include "HSV.hpp"
```

Classes

- class meow::HSV< T >
- class meow::HSVf

Namespaces

- meow

Functions

- template<class RGB_T , class HSV_T >
  void meow::RGB_to_HSV (RGB< RGB_T > const &rgb, HSV< HSV_T > ∗hsv)
- template<class HSV_T , class RGB_T >
  void meow::HSV_to_RGB (HSV< HSV_T > const &hsv, RGB< RGB_T > ∗rgb)
- template<class YUV_T , class HSV_T >
  void meow::YUV_to_HSV (YUV< YUV_T > const &yuv, HSV< HSV_T > ∗hsv)
- template<class HSV_T , class YUV_T >
  void meow::HSV_to_YUV (HSV< HSV_T > const &hsv, YUV< YUV_T > ∗yuv)
- template<class HSL_T , class HSV_T >
  void meow::HSL_to_HSV (HSL< HSL_T > const &hsl, HSV< HSV_T > ∗hsv)
- template<class HSV_T , class HSL_T >
  void meow::HSV_to_HSL (HSV< HSV_T > const &hsv, HSL< HSL_T > ∗hsl)

7.6 meowpp/colors/HSV.hpp File Reference

```
#include "HSV.h"
#include "RGB.h"
#include "YUV.h"
#include "HSL.h"
#include "../utility.h"
```

Namespaces

- meow

Functions

- template<class RGB_T , class HSV_T >
  void meow::RGB_to_HSV (RGB< RGB_T > const &rgb, HSV< HSV_T > ∗hsv)
- template<class HSV_T , class RGB_T >
  void meow::HSV_to_RGB (HSV< HSV_T > const &hsv, RGB< RGB_T > ∗rgb)
- template<class YUV_T , class HSV_T >
  void meow::YUV_to_HSV (YUV< YUV_T > const &yuv, HSV< HSV_T > ∗hsv)
- template<class HSV_T , class YUV_T >
  void meow::HSV_to_YUV (HSV< HSV_T > const &hsv, YUV< YUV_T > ∗yuv)
- template<class HSL_T , class HSV_T >
  void meow::HSL_to_HSV (HSL< HSL_T > const &hsl, HSV< HSV_T > ∗hsv)
- template<class HSV_T , class HSL_T >
  void meow::HSV_to_HSL (HSV< HSV_T > const &hsv, HSL< HSL_T > ∗hsl)

7.7 meowpp/colors/HSV_Space.h File Reference

```
#include "Color3_Space.h"
#include "../geo/Vectors.h"
#include "RGB_Space.h"
#include "YUV_Space.h"
#include "HSL_Space.h"
#include "../math/utility.h"
#include <cstdlib>
```

Classes

- class meow::HSVf_Space

  ***Y**(), **U**(), **V**()*

Namespaces

- meow

Functions

- void meow::colorTransformate (RGBf_Space const &rgb, HSVf_Space ∗hsv)

  RGBf_Space to HSVf_Space
- void meow::colorTransformate (YUVf_Space const &yuv, HSVf_Space ∗hsv)

  YUVf_Space to HSVf_Space
- void meow::colorTransformate (HSLf_Space const &hsl, HSVf_Space ∗hsv)

  HSLf_Space to HSVf_Space
- void meow::colorTransformate (HSVf_Space const &hsv, RGBf_Space ∗rgb)

  HSVf_Space to RGBf_Space
- void meow::colorTransformate (HSVf_Space const &hsv, YUVf_Space ∗yuv)

  HSVf_Space to YUVf_Space
- void meow::colorTransformate (HSVf_Space const &hsv, HSLf_Space ∗hsl)

  HSVf_Space to HSLf_Space
- void meow::colorTransformate (HSVf_Space const &hsv, RGBi_Space ∗rgb)

  HSVf_Space to RGBi_Space
- void meow::colorTransformate (RGBi_Space const &rgb, HSVf_Space ∗hsv)

  RGBi_Space to HSVf_Space

7.8 meowpp/colors/RGB.h File Reference

```
#include "RGB.hpp"
```

Classes

- class meow::RGB< T >
- class meow::RGBf
- class meow::RGBi

Namespaces

- meow

7.9 meowpp/colors/RGB.hpp File Reference

```
#include <algorithm>
#include <cstdint>
```

Namespaces

- meow

7.10 meowpp/colors/RGB_Space.h File Reference

```
#include "Color3_Space.h"
#include "../geo/Vectors.h"
#include "../math/utility.h"
#include <cstdlib>
```

Classes

- class meow::RGBi_Space

    Red**, **Green**, **Blue
- class meow::RGBf_Space

    Red**, **Green**, **Blue

Namespaces

- meow

Functions

- void meow::colorTransformate (RGBi_Space const &a, RGBf_Space ∗b)

    `RGBi_Space` to `RGBf_Space`
- void meow::colorTransformate (RGBf_Space const &a, RGBi_Space ∗b)

    `RGBf_Space` to `RGBi_Space`

7.11 meowpp/colors/YUV.h File Reference

```
#include "RGB.h"
#include "YUV.hpp"
```

Classes

- class meow::YUV< T >
- class meow::YUVf

Namespaces

- meow

Functions

- template<class RGB_T , class YUV_T >
  void meow::RGB_to_YUV (RGB< RGB_T > const &rgb, YUV< YUV_T > *yuv)
- template<class YUV_T , class RGB_T >
  void meow::YUV_to_RGB (YUV< YUV_T > const &yuv, RGB< RGB_T > *rgb)

7.12 meowpp/colors/YUV.hpp File Reference

```
#include <algorithm>
#include "RGB.h"
#include "../utility.h"
```

Namespaces

- meow

Functions

- template<class RGB_T , class YUV_T >
  void meow::RGB_to_YUV (RGB< RGB_T > const &rgb, YUV< YUV_T > *yuv)
- template<class YUV_T , class RGB_T >
  void meow::YUV_to_RGB (YUV< YUV_T > const &yuv, RGB< RGB_T > *rgb)

7.13 meowpp/colors/YUV_Space.h File Reference

```
#include "Color3_Space.h"
#include "../geo/Vectors.h"
#include "RGB_Space.h"
#include "../math/utility.h"
#include <cstdlib>
```

Classes

- class meow::YUVf_Space

  Y(), U(), V()

Namespaces

- meow

Functions

- void meow::colorTransformate (RGBf_Space const &rgb, YUVf_Space ∗yuv)

  RGBf_Space to YUVf_Space
- void meow::colorTransformate (YUVf_Space const &yuv, RGBf_Space ∗rgb)

  YUVf_Space to RGBf_Space
- void meow::colorTransformate (RGBi_Space const &rgb, YUVf_Space ∗yuv)

  RGBi_Space to YUVf_Space
- void meow::colorTransformate (YUVf_Space const &yuv, RGBi_Space ∗rgb)

  YUVf_Space to RGBi_Space

7.14 meowpp/dsa/BinaryIndexTree.h File Reference

```
#include <cstdlib>
#include <vector>
#include <algorithm>
```

Classes

- class meow::BinaryIndexTree< Value >

  SegmentTree

Namespaces

- meow

7.15 meowpp/dsa/DisjointSet.h File Reference

```
#include <vector>
#include <cstdlib>
#include <cstdio>
```

Classes

- class meow::DisjointSet

Namespaces

- meow

7.16 meowpp/dsa/HashTable.h File Reference

```
#include <vector>
#include <list>
```

Classes

- class meow::HashTableList< Data, HashFunc >

  keylisthash_table

Namespaces

- meow

7.17 meowpp/dsa/KD_Tree.h File Reference

```
#include "../utility.h"
#include "../math/utility.h"
#include <cstdlib>
#include <vector>
#include <algorithm>
#include <queue>
```

Classes

- class meow::KD_Tree< Vector, Scalar >

  k-dimension tree

Namespaces

- meow

7.18 meowpp/dsa/MergeableHeap.h File Reference

```
#include <cstdlib>
#include <algorithm>
```

Classes

- class meow::MergeableHeap< Element >

  Maximum-Heap , heap, merge

Namespaces

- meow

7.19 meowpp/dsa/SegmentTree.h File Reference

```
#include "../math/utility.h"
#include <vector>
#include <algorithm>
#include <cstdlib>
```

Classes

- class meow::SegmentTree< Value >

Namespaces

- meow

7.20 meowpp/dsa/SplayTree.h File Reference

```
#include <cstdlib>
#include <utility>
#include "../math/utility.h"
```

Classes

- class meow::SplayTree< Key, Value >

  , Key->Value . std::map , split , merge , keyOffset
- class meow::SplayTree< Key, Value >::Element

  stl iterator ,Element
- class meow::SplayTree_Range< Key, Value >

  SplayTree, , (operator SegmentTree)
- class meow::SplayTree_Range< Key, Value >::Element

  stl iterator ,Element

Namespaces

- meow

7.21 meowpp/dsa/VP_Tree.h File Reference

```
#include "../math/utility.h"
#include <cstdlib>
#include <list>
#include <vector>
#include <stack>
#include <queue>
```

Classes

- class meow::VP_Tree< Vector, Scalar >

  KD_Tree

Namespaces

- meow

7.22 meowpp/geo/Vectors.h File Reference

```
#include "../math/utility.h"
#include "../math/Vector.h"
#include "../math/Matrix.h"
#include <cmath>
```

Classes

- class meow::Vector2D< Scalar >

  2D's vector
- class meow::Vector3D< Scalar >

  3D's vector

Namespaces

- meow

7.23 meowpp/gra/Bitmap.h File Reference

```
#include "../math/utility.h"
#include "../math/Matrix.h"
#include "../oo/ObjBase.h"
#include <vector>
#include <cmath>
#include <string>
#include <typeinfo>
#include <cstdlib>
```

Classes

- class meow::Bitmap< Pixel >

Namespaces

- meow

7.24 meowpp/gra/BundleAdjustment.h File Reference

```
#include "Eye.h"
#include "../oo/ObjBase.h"
```

Classes

- struct meow::SceneInfo< Pixel >
- class meow::BundleAdjustment< Pixel >

Namespaces

- meow

Enumerations

- enum meow::SceneInfoFlags { meow::CAN_OFFSET = 0x01, meow::CAN_ROTATE = 0x02, meow::CAN_ZOOM = 0x04 }

7.25 meowpp/gra/BundleAdjustment_LM.h File Reference

```
#include "Eye.h"
#include "BundleAdjustment.h"
#include "../math/methods.h"
#include "../math/Vector.h"
#include "../math/Matrix.h"
#include "../math/utility.h"
#include "../oo/ObjBase.h"
#include <algorithm>
```

Classes

- class meow::BundleAdjustment_LM< Pixel >

Namespaces

- meow

7.26 meowpp/gra/Camera.h File Reference

```
#include "Photo.h"
#include "IdentityPoints.h"
#include "../Self.h"
#include "../math/utility.h"
#include "../math/LinearTransformations.h"
#include "../math/methods.h"
#include "../oo/ObjBase.h"
```

Classes

- class meow::Camera< Pixel >

  Camera.

Namespaces

- meow

7.27 meowpp/gra/Eye.h File Reference

```
#include "Camera.h"
#include "../Self.h"
#include "../oo/ObjBase.h"
```

Classes

- class meow::Eye< Pixel >

  `Camera` *offset transformation*

Namespaces

- meow

7.28 meowpp/gra/FeaturePoint.h File Reference

```
#include "../oo/ObjBase.h"
#include "../math/Vector.h"
#include <string>
#include <typeinfo>
#include <cstdlib>
#include <cstdio>
```

Classes

- class meow::FeaturePoint< Scalar, Description >

Namespaces

- meow

7.29 meowpp/gra/FeaturePointsDetector.h File Reference

```
#include "../oo/ObjBase.h"
#include "FeaturePoint.h"
#include "Bitmap.h"
#include <vector>
```

Classes

- class meow::FeaturePointsDetector< Pixel >

Namespaces

- meow

7.30 meowpp/gra/FeaturePointsDetector_Harris.h File Reference

```
#include "FeaturePointsDetector.h"
#include "Bitmap.h"
#include "FeaturePoint.h"
#include "../dsa/DisjointSet.h"
#include "../Self.h"
#include <vector>
```

Classes

- class meow::FeaturePointsDetector_Harris< Pixel >

  Harris corner detect.

Namespaces

- meow

Macros

- #define FPD_Harris FeaturePointsDetector_Harris

7.30.1 Macro Definition Documentation

#define FPD_Harris FeaturePointsDetector_Harris

Definition at line 25 of file FeaturePointsDetector_Harris.h.

7.31 meowpp/gra/FeaturePointsMatch.h File Reference

```
#include "FeaturePoint.h"
#include "../utility.h"
#include "../oo/ObjBase.h"
#include <cstdlib>
```

Classes

- class meow::FeaturePointsMatch< Scalar, Description >

Namespaces

- meow

Typedefs

- typedef PairToPair< size_t,
  size_t, size_t, size_t > meow::FeaturePointIndexPair
- typedef std::vector
  < FeaturePointIndexPair > meow::FeaturePointIndexPairs

7.32 meowpp/gra/FeaturePointsMatch_K_Match.h File Reference

```
#include "FeaturePointsMatch.h"
#include "../Self.h"
#include "../dsa/VP_Tree.h"
#include "../oo/ObjBase.h"
#include <cstdlib>
```

Classes

- class meow::FeaturePointsMatch_K_Match< Scalar, Description >

Namespaces

- meow

Macros

- #define FPMKM FeaturePointsMatch_K_Match

7.32.1 Macro Definition Documentation

#define FPMKM FeaturePointsMatch_K_Match

Definition at line 17 of file FeaturePointsMatch_K_Match.h.

7.33 meowpp/gra/IdentityPoints.h File Reference

```
#include "../Self.h"
#include "../math/Vector.h"
#include "../oo/ObjBase.h"
#include <map>
#include <set>
#include <cstdlib>
```

Classes

- class meow::IdentityPoints< ID, Scalar >

  std::map<ID,Vector<Scalar> >

Namespaces

- meow

7.34 meowpp/gra/Photo.h File Reference

```
#include "Bitmap.h"
#include "../Self.h"
#include "../geo/Vectors.h"
#include "../math/utility.h"
#include "../math/Matrix.h"
#include "../math/Transformations.h"
#include "../oo/ObjBase.h"
#include <vector>
#include <cmath>
#include <string>
#include <typeinfo>
#include <cstdlib>
```

Classes

- class meow::Photo< Pixel >

Namespaces

- meow

7.35 meowpp/gra/ViewPort.h File Reference

```
#include "../oo/ObjBase.h"
```

Classes

- class meow::ViewPort

Namespaces

- meow

7.36 meowpp/gra/WatchBall.h File Reference

```
#include "Camera.h"
#include "../Self.h"
#include "../geo/Vectors.h"
#include "../math/LinearTransformations.h"
#include "../oo/ObjBase.h"
#include <cmath>
#include <vector>
```

Classes

- class meow::WatchBall< Pixel >

  ***camera**, offset, rotation*

Namespaces

- meow

7.37 meowpp/math/LinearTransformation.h File Reference

```
#include "Transformation.h"
#include "Matrix.h"
#include <cstdlib>
```

Classes

- class meow::LinearTransformation< Scalar >

  A base class for implementing kinds of linear transformations.

Namespaces

- meow

7.38 meowpp/math/LinearTransformations.h File Reference

```
#include "LinearTransformation.h"
#include "Matrix.h"
#include "utility.h"
#include "../Self.h"
#include "../geo/Vectors.h"
#include <cstdlib>
```

Classes

- class meow::Rotation3D< Scalar >

  Rotation a point/vector alone an axis with given angle in 3D world.

Namespaces

- meow

7.39 meowpp/math/Matrix.h File Reference

```
#include "../Self.h"
#include <vector>
#include <algorithm>
#include <cstdlib>
```

Classes

- class meow::Matrix< Entry >

  matrix

Namespaces

- meow

7.40 meowpp/math/methods.h File Reference

```
#include "Matrix.h"
#include "Vector.h"
#include "utility.h"
#include <cstdlib>
#include <vector>
```

Namespaces

- meow

Functions

- template< class Data , class WeightingClass >
  std::vector< Data > meow::ransac (std::vector< Data > const &data, WeightingClass const &w, size_t N, double p0, double P)

  *Run the **RANSAC** method to approach the best solution.*
- template< class Scalar , class Function >
  Vector< Scalar > meow::levenbergMarquardt (Function const &f, Vector< Scalar > const &init, int counter=-1)
- template< class Scalar , class Function >
  Vector< Scalar > meow::levenbergMarquardtTraining (Function &f, Vector< Scalar > const &init, Scalar const &init_mu, Scalar const &mu_pow, Scalar const &er_max, int retry_number, int counter)

7.41 meowpp/math/Transformation.h File Reference

```
#include "Matrix.h"
#include "../Self.h"
#include <list>
#include <cstdlib>
```

Classes

- class meow::Transformation< Scalar >

  A base class for implementing kinds of transformations.

Namespaces

- meow

7.42 meowpp/math/Transformations.h File Reference

```
#include "Transformation.h"
#include "Matrix.h"
#include "utility.h"
#include "../Self.h"
#include <cstdlib>
```

Classes

- class meow::BallProjection< Scalar >

  A ball projection is to project the given vector to a hyper-sphere.
- class meow::PhotoProjection< Scalar >

  *A **photo projection** is a kind of transformation that project point/vector to a flat **photo**.*

Namespaces

- meow

7.43 meowpp/math/utility.h File Reference

```
#include <cstdlib>
#include <vector>
#include <algorithm>
#include <cmath>
```

Namespaces

- meow

Functions

- template<class T >
  T meow::noEPS (T value, T eps=1e-9)

  abs() < eps, 0,
- template<class T >
  T meow::normalize (T lower, T upper, T value)

  (value-lower)/(upper-lower)
- template<class T >
  T meow::denormalize (T lower, T upper, T _ratio)

  (lower+_ratio∗(upper-lower))
- template<class T >
  T meow::ratioMapping (T l1, T u1, T m1, T l2, T u2)

  `denormalize(l2,u2,normalize(l1,u1,m1))`

- template<class T >
  T meow::inRange (T const &mn, T const &mx, T const &v)

  std::min(mx,std::max(mn,v))
- template<class T >
  T meow::isInRange (T const &mn, T const &mx, T const &x)

  (mn <= x && x <= mx)
- template<class T >
  T meow::squ (T const &x)

  *x*x*
- template<class T >
  T meow::cub (T const &x)

  *x*x*x*
- template<class T >
  double meow::average (T const &beg, T const &end, double sigs)

  sigs
- template<class T >
  double meow::average (T const &beg, T const &end, T const &p, double sigs)

  sigs, p
- template<class T >
  T meow::tAbs (T const &t)

Variables

- static const double meow::PI = 3.14159265358979323846264338327950288

  ...

7.44 meowpp/utility.h File Reference

```
#include <cstdlib>
#include <cstring>
#include <cstdio>
#include <cstdarg>
#include <string>
```

Classes

- struct meow::PairToPair< F1, F2, T1, T2 >

  .from.first, .from.second, .to.first, .to.second

Namespaces

- meow

Macros

- #define debugPrintf(str)

  DEBUGdefine, stderr,

Functions

- std::string meow::stringPrintf (char const ∗fmt,...)

  Cprintf, `std::string`
- std::string meow::stringReplace (std::string str, std::string const &from, std::string const &to)

  patternpattern
- bool meow::cstringEndWith (char const ∗str, int n,...)

  patterns
- void meow::debugPrintf_ (char const ∗file, char const ∗func, size_t line, char const ∗msg)
- void meow::messagePrintf (int level_change, char const ∗fmt,...)

- bool meow::filenameCompare (std::string const &f1, std::string const &f2)

7.44.1 Macro Definition Documentation

#define debugPrintf(*str*)

Value:

```
debugPrintf_(\
             __FILE__,\
             __FUNCTION__,\
             __LINE__,\
             str)
```

DEBUGdefine, stderr,

Parameters

`in`	*str*	, c string, `char const∗`

Returns

Note

macro

Definition at line 103 of file utility.h.

7.45 meowpp/math/Vector.h File Reference

```
#include "../Self.h"
#include "Matrix.h"
#include <vector>
#include <cmath>
```

Classes

- class meow::Vector< Scalar >

  vector

Namespaces

- meow

7.46 meowpp/oo/ObjArray.h File Reference

```
#include "ObjBase.h"
#include "../Self.h"
#include <vector>
#include <string>
#include <typeinfo>
#include <cstdio>
#include <cstdlib>
```

Classes

- class meow::ObjArray< T >

    `std::vector`, ObjBase

Namespaces

- meow

7.47 meowpp/oo/ObjBase.h File Reference

```
#include <cstdio>
#include <typeinfo>
#include <string>
```

Classes

- class meow::ObjBase

    Base, read, write, create, ...

Namespaces

- meow

7.48 meowpp/oo/ObjDictionary.h File Reference

```
#include "ObjBase.h"
#include "../Self.h"
#include <string>
#include <typeinfo>
#include <map>
#include <cstdio>
#include <cstdlib>
```

Classes

- class meow::ObjDictionary< Key, Value >

    `std::map`, ObjBase

Namespaces

- meow

7.49 meowpp/oo/ObjProperties.h File Reference

```
#include "ObjBase.h"
#include <cstdlib>
```

Classes

- class meow::ObjProperties< SID >

Namespaces

- meow

7.50 meowpp/oo/ObjSelector.h File Reference

```
#include "ObjBase.h"
#include <utility>
#include <vector>
#include <string>
#include <map>
#include <cstdlib>
#include <cstdio>
```

Classes

- class meow::ObjSelector< id >

  register, runtimestringnewclass

Namespaces

- meow

Variables

- static const size_t meow::kGlobalSeletorID = 0

7.51 meowpp/oo/ObjTypes.h File Reference

```
#include "../Self.h"
#include "ObjBase.h"
#include <cstdlib>
#include <cstdio>
```

Classes

- class meow::ObjType< Type, ReaderWriter >

  `Type` , *ObjBase*
- class meow::ReaderWriter_int
- class meow::ReaderWriter_size_t
- class meow::ReaderWriter_double
- class meow::ReaderWriter_string

Namespaces

- meow

Typedefs

- typedef ObjType< int,
  ReaderWriter_int > meow::ObjInt
- typedef ObjType< size_t,
  ReaderWriter_size_t > meow::ObjSizeT
- typedef ObjType< double,
  ReaderWriter_double > meow::ObjDouble
- typedef ObjType< std::string,
  ReaderWriter_string > meow::ObjString

7.52 meowpp/oo/Register_Implement.h File Reference

```
#include <map>
#include "Register_Implement.hpp"
```

Classes

- class meow::ImplementInterface< T >
- class meow::RegisterInterface< T >

Namespaces

- meow

7.53 meowpp/oo/Register_Implement.hpp File Reference

```
#include <map>
```

Namespaces

- meow

7.54 meowpp/Self.h File Reference

```
#include <cstdlib>
```

Classes

- class meow::Self< Data >

  A little class use for packing the data part of another class. With this technique, it can achieve Copy-On-Write(COR) mechanism at background and have a reference mechanism which much more flexible then the one C++ has.

Namespaces

- meow

7.55 meowpp/Usage.h File Reference

```
#include "utility.h"
#include <cstdlib>
#include <algorithm>
#include <string>
#include <vector>
#include <map>
```

Classes

- class meow::Usage

  , usage document, argc, argv

Namespaces

- meow

7.56 meowpp/Usage.hpp File Reference

```
#include <string>
#include <cstdint>
#include <vector>
#include <map>
#include <algorithm>
#include <cstdlib>
#include "utility.h"
#include <unistd.h>
```

Namespaces

- meow

7.57 meowpp/utility.hpp File Reference

```
#include <string>
#include <stack>
#include <cstdio>
#include <cstdarg>
#include <algorithm>
#include <cstdint>
#include <cctype>
#include <cstring>
#include <cmath>
```

Namespaces

- meow

Functions

- std::string meow::stringPrintf (char const ∗fmt,...)

  Cprintf, `std::string`

- std::string meow::stringReplace (std::string str, std::string const &from, std::string const &to)

    patternpattern
- bool meow::cstringEndWith (char const ∗str, int n,...)

    patterns
- void meow::debugPrintf_ (char const ∗file, char const ∗func, int32_t line, char const ∗msg)
- void meow::messagePrintf (int32_t level_change, char const ∗fmt,...)
- double meow::noEPS (double value, double eps)
- double meow::normalize (double lower, double upper, double value)
- double meow::denormalize (double lower, double upper, double ratio)
- double meow::ratioMapping (double l1, double u1, double m1, double l2, double u2)
- bool meow::filenameCompare (std::string const &f1, std::string const &f2)

- template<class T >
  T meow::inRange (T const &mn, T const &mx, T const &v)

    `std::min(mx,std::max(mn,v))`
- template<class T >
  double meow::average (T const &beg, T const &end, double sigs)

    `sigs`
- template<class T >
  double meow::average (T const &beg, T const &end, T const &p, double sigs)

    `sigs, p`