cppqt – Framework zur Vorlesung „Bildgenerierung“ im WS 2015/2016

Erzeugt von Doxygen 1.8.14

Inhaltsverzeichnis

1 Hierarchie-Verzeichnis

1.1 Klassenhierarchie

Die Liste der Ableitungen ist -mit Einschränkungen- alphabetisch sortiert:

2 Klassen-Verzeichnis

2.1 Auflistung der Klassen

Hier folgt die Aufzählung aller Klassen, Strukturen, Varianten und Schnittstellen mit einer Kurzbeschreibung:

3 Datei-Verzeichnis

3.1 Auflistung der Dateien

Hier folgt die Aufzählung aller Dateien mit einer Kurzbeschreibung:

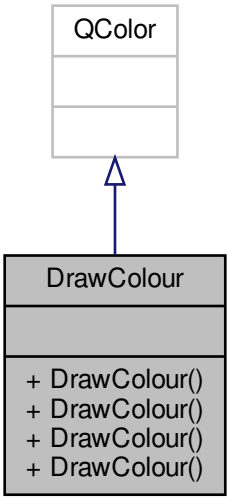
4 Klassen-Dokumentation

4.1 DrawColour Klassenreferenz

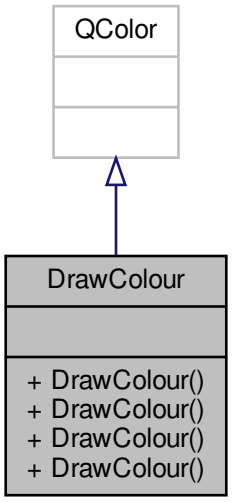
Farbklasse.

`#include <drawcolour.h>`

Klassendiagramm für DrawColour:



Zusammengehörigkeiten von DrawColour:



Öffentliche Methoden

- DrawColour ()
  Default-Konstruktor: Farbe schwarz.
- DrawColour (const QColor &c2)
  Kopier-Konstruktor.
- DrawColour (int r, int g, int b)
  Konstruktur für RGB-Werte.
- DrawColour (int gr)
  Konstruktor für Grauwerte.

4.1.1 Ausführliche Beschreibung

Farbklasse.

Definiert in Zeile 13 der Datei drawcolour.h.

4.1.2 Beschreibung der Konstruktoren und Destruktoren

4.1.2.1 DrawColour() [1/4]

```
DrawColour::DrawColour ( )  [inline]
```

Default-Konstruktor: Farbe schwarz.

Definiert in Zeile 17 der Datei drawcolour.h.

4.1.2.2 DrawColour() [2/4]

```
DrawColour::DrawColour (
            const QColor & c2 )  [inline]
```

Kopier-Konstruktor.

Definiert in Zeile 20 der Datei drawcolour.h.

4.1.2.3 DrawColour() [3/4]

```
DrawColour::DrawColour (
            int r,
            int g,
            int b )  [inline]
```

Konstruktur für RGB-Werte.

Definiert in Zeile 23 der Datei drawcolour.h.

4.1.2.4 DrawColour() [4/4]

```
DrawColour::DrawColour (
            int gr )  [inline]
```

Konstruktor für Grauwerte.

Definiert in Zeile 26 der Datei drawcolour.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

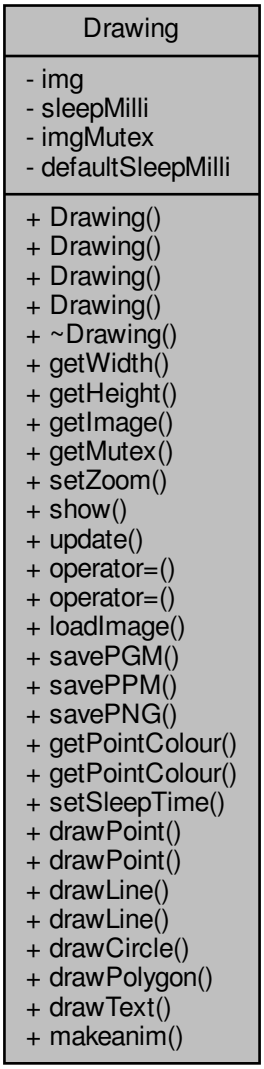
- drawcolour.h

4.2 Drawing Klassenreferenz

Ein Bild.

```
#include <drawing.h>
```

Zusammengehörigkeiten von Drawing:



Öffentliche Methoden

- Drawing ()

  Default-Konstruktor.
- Drawing (const Drawing &d2)

  Kopier-Konstruktor.
- Drawing (int w, int h)

  Konstruktor für vorgegebene Breite `w` *und Höhe* `h`*.*
- Drawing (int w, int h, const DrawColour &colour)

  Konstruktor für vorgegebene Breite `w`*, Höhe* `h` *und Hintergrundfarbe* `colour`*.*
- ~Drawing ()

  Destruktor. Informiert DrawWindow über das Ableben dieser Instanz.

- int getWidth () const

  Gibt die Breite des Bildes aus.
- int getHeight () const

  Gibt die Hoehe des Bildes aus.
- const QImage * getImage () const

  Liefert Zeiger auf das enthaltene Image.
- QMutex * getMutex () const

  Liefert Zeiger auf Mutex zum Sperren des Bildzugriffs.
- void setZoom (int z)

  Ändert den Zoomfaktor für des Fensters, falls es angezeigt wird.
- void show () const

  Zeigt dieses Bild im Graphikfenster an, ersetzt bisherigen Fensterinhalt.
- void update () const

  Informiert das Zeichenfenster über Veränderungen.
- Drawing & operator= (const Drawing &d2)

  Zuweisungsoperator.
- Drawing & operator= (DrawColour colour)

  Setzt das ganze Bild auf die Farbe `colour`.
- void loadImage (const std::string &filename)

  Lädt die Bilddatei `filename`.
- void savePGM (const std::string &filename) const

  Speichert das Bild im pgm-Format in der Datei `filename`.
- void savePPM (const std::string &filename) const

  Speichert das Bild im ppm-Format in der Datei `filename`.
- void savePNG (const std::string &filename) const

  Speichert das Bild im png-Format in der Datei `filename`.
- DrawColour getPointColour (int x, int y) const

  Liefert die Farbe des Pixels an Position (`x, y`*)*
- DrawColour getPointColour (IPoint2D p) const

  Liefert die Farbe des Pixels an Position `p`.
- void setSleepTime (int milli)

  Setzt die Wartezeit für verzögertes Malen.
- void drawPoint (int x, int y, DrawColour colour=0, bool drawslow=false, bool drawXOR=false)

  Zeichnet einen Punkt der Farbe `colour` *an Position (*`x, y`*) ins Bild.*
- void drawPoint (IPoint2D p, DrawColour colour=0, bool drawslow=false, bool drawXOR=false)

  Zeichnet einen Punkt der Farbe `colour` *an Position* `p` *ins Bild.*
- void drawLine (int x1, int y1, int x2, int y2, DrawColour colour=0, bool drawslow=false)

  Zeichnet eine Linie der Farbe `colour` *von (*`x1, y1`*) nach (*`x2 y2`*) ins Bild.*
- void drawLine (IPoint2D p1, IPoint2D p2, DrawColour colour=0, bool drawslow=false)

  Zeichnet eine Linie der Farbe `colour` *von* `p1` *nach* `p2` *ins Bild.*
- void drawCircle (IPoint2D c, int r, bool filled=false, DrawColour fg=0, DrawColour bg=255, bool drawslow=false)

  Zeichnet einen Kreis mit Mittelpunkt `c` *und Radius* `r` *ins Bild.*
- void drawPolygon (const std::vector< IPoint2D > &e, bool filled=false, DrawColour fg=0, DrawColour bg=255, bool drawslow=false)

  Zeichnet ein Polygon mit Ecken `e` *ins Bild.*
- void drawText (IPoint2D pos, const std::string &text, DrawColour colour=0)

  Zeichnet den Text `text` *mit der Frabe* `colour` *an die Position* `pos` *ins Bild.*

Öffentliche, statische Methoden

- static void makeanim (const std::vector< Drawing > &pics, const std::string &filename, const std::string &type, int delay=4)

  Erzeugt eine Animation namens `filename.type` aus den Einzelbildern in pics.

Private Attribute

- QImage img
- int sleepMilli
- QMutex imgMutex

Statische, private Attribute

- static const int defaultSleepMilli = 10

4.2.1 Ausführliche Beschreibung

Ein Bild.

Definiert in Zeile 21 der Datei drawing.h.

4.2.2 Beschreibung der Konstruktoren und Destruktoren

4.2.2.1 Drawing() [1/4]

```
Drawing::Drawing ( )  [inline]
```

Default-Konstruktor.

Erzeugt ein leeres weißes Bild der Standardgröße 200×100

Definiert in Zeile 31 der Datei drawing.h.

Benutzt img.

4.2.2.2 Drawing() [2/4]

```
Drawing::Drawing (
            const Drawing & d2 )  [inline]
```

Kopier-Konstruktor.

Definiert in Zeile 35 der Datei drawing.h.

4.2.2.3 Drawing() [3/4]

```
Drawing::Drawing (
            int w,
            int h )  [inline]
```

Konstruktor für vorgegebene Breite `w` und Höhe `h`.

Der Hintergrund wird automatisch auf weiß gesetzt.

Definiert in Zeile 39 der Datei drawing.h.

Benutzt img.

4.2.2.4 Drawing() [4/4]

```
Drawing::Drawing (
            int w,
            int h,
            const DrawColour & colour )  [inline]
```

Konstruktor für vorgegebene Breite `w`, Höhe `h` und Hintergrundfarbe `colour`.

Definiert in Zeile 45 der Datei drawing.h.

Benutzt img.

4.2.2.5 ~Drawing()

```
Drawing::~Drawing ( )
```

Destruktor. Informiert DrawWindow über das Ableben dieser Instanz.

Definiert in Zeile 21 der Datei drawing.cc.

Benutzt DrawOps::getInstance() und DrawOps::obituary().

4.2.3 Dokumentation der Elementfunktionen

4.2.3.1 drawCircle()

```
void Drawing::drawCircle (
            IPoint2D c,
            int r,
            bool filled = false,
            DrawColour fg = 0,
            DrawColour bg = 255,
            bool drawslow = false )
```

Zeichnet einen Kreis mit Mittelpunkt `c` und Radius `r` ins Bild.

Der Rand hat die Farbe `fg`; falls `filled` auf `true` gesetzt ist, wird mit Farbe `bg` gefüllt.

Definiert in Zeile 115 der Datei drawing.cc.

Benutzt IOThread::msleep(), Point2D< T >::x und Point2D< T >::y.

4.2.3.2 drawLine() [1/2]

```
void Drawing::drawLine (
            int x1,
            int y1,
            int x2,
            int y2,
            DrawColour colour = 0,
            bool drawslow = false )
```

Zeichnet eine Linie der Farbe `colour` von (`x1`, `y1`) nach (`x2` `y2`) ins Bild.

Definiert in Zeile 100 der Datei drawing.cc.

Benutzt IOThread::msleep().

Wird benutzt von drawLine() und maindraw().

4.2.3.3 drawLine() [2/2]

```
void Drawing::drawLine (
            IPoint2D p1,
            IPoint2D p2,
            DrawColour colour = 0,
            bool drawslow = false )  [inline]
```

Zeichnet eine Linie der Farbe `colour` von `p1` nach `p2` ins Bild.

Definiert in Zeile 125 der Datei drawing.h.

Benutzt drawLine(), Point2D< T >::x und Point2D< T >::y.

4.2.3.4 drawPoint() [1/2]

```
void Drawing::drawPoint (
            int x,
            int y,
            DrawColour colour = 0,
            bool drawslow = false,
            bool drawXOR = false )
```

Zeichnet einen Punkt der Farbe `colour` an Position (`x`, `y`) ins Bild.

Falls `drawslow` auf `true` gesetzt ist, werden die Punkte verzögert gezeichnet. Falls `drawXOR` auf `true` gesetzt ist, werden die Punkte im XOR-Modus gefärbt, die Pixelfarbe also mit der Hintergrundfarbe kombiniert.

Definiert in Zeile 82 der Datei drawing.cc.

Benutzt IOThread::msleep().

Wird benutzt von drawPoint() und maindraw().

4.2.3.5 drawPoint() [2/2]

```
void Drawing::drawPoint (
            IPoint2D p,
            DrawColour colour = 0,
            bool drawslow = false,
            bool drawXOR = false )  [inline]
```

Zeichnet einen Punkt der Farbe `colour` an Position `p` ins Bild.

Falls `drawslow` auf `true` gesetzt ist, werden die Punkte verzögert gezeichnet. Falls `drawXOR` auf `true` gesetzt ist, werden die Punkte im XOR-Modus gefärbt, die Pixelfarbe also mit der Hintergrundfarbe kombiniert.

Definiert in Zeile 116 der Datei drawing.h.

Benutzt drawPoint(), Point2D< T >::x und Point2D< T >::y.

4.2.3.6 drawPolygon()

```
void Drawing::drawPolygon (
            const std::vector< IPoint2D > & e,
            bool filled = false,
            DrawColour fg = 0,
            DrawColour bg = 255,
            bool drawslow = false )
```

Zeichnet ein Polygon mit Ecken `e` ins Bild.

Der Rahmen hat die Farbe `fg`; falls `filled` auf `true` gesetzt ist, wird mit Farbe `bg` gefüllt.

Definiert in Zeile 134 der Datei drawing.cc.

Benutzt IOThread::msleep().

4.2.3.7 drawText()

```
void Drawing::drawText (
            IPoint2D pos,
            const std::string & text,
            DrawColour colour = 0 )
```

Zeichnet den Text `text` mit der Frabe `colour` an die Position `pos` ins Bild.

Definiert in Zeile 156 der Datei drawing.cc.

Benutzt Point2D< T >::x und Point2D< T >::y.

4.2.3.8 getHeight()

```
int Drawing::getHeight ( ) const  [inline]
```

Gibt die Hoehe des Bildes aus.

Definiert in Zeile 56 der Datei drawing.h.

Benutzt img.

Wird benutzt von DrawWindow::changeDrawing(), DrawWindow::changeSize() und DrawWindow::setZoom().

4.2.3.9 getImage()

```
const QImage* Drawing::getImage ( ) const  [inline]
```

Liefert Zeiger auf das enthaltene Image.

Definiert in Zeile 59 der Datei drawing.h.

Benutzt img.

Wird benutzt von DrawWindow::paintEvent().

4.2.3.10 getMutex()

```
QMutex* Drawing::getMutex ( ) const  [inline]
```

Liefert Zeiger auf Mutex zum Sperren des Bildzugriffs.

Definiert in Zeile 62 der Datei drawing.h.

Benutzt imgMutex.

Wird benutzt von DrawWindow::paintEvent().

4.2.3.11 getPointColour() [1/2]

```
DrawColour Drawing::getPointColour (
            int x,
            int y ) const
```

Liefert die Farbe des Pixels an Position (x, y)

Definiert in Zeile 71 der Datei drawing.cc.

Wird benutzt von getPointColour().

4.2.3.12 getPointColour() [2/2]

```
DrawColour Drawing::getPointColour (
            IPoint2D p ) const  [inline]
```

Liefert die Farbe des Pixels an Position p.

Definiert in Zeile 95 der Datei drawing.h.

Benutzt getPointColour(), Point2D< T >::x und Point2D< T >::y.

4.2.3.13 getWidth()

```
int Drawing::getWidth ( ) const  [inline]
```

Gibt die Breite des Bildes aus.

Definiert in Zeile 53 der Datei drawing.h.

Benutzt img.

Wird benutzt von DrawWindow::changeDrawing(), DrawWindow::changeSize() und DrawWindow::setZoom().

4.2.3.14 loadImage()

```
void Drawing::loadImage (
            const std::string & filename )
```

Lädt die Bilddatei filename.

Definiert in Zeile 56 der Datei drawing.cc.

Benutzt DrawOps::changeSize() und DrawOps::getInstance().

Wird benutzt von maindraw().

4.2.3.15 makeanim()

```
void Drawing::makeanim (
            const std::vector< Drawing > & pics,
            const std::string & filename,
            const std::string & type,
            int delay = 4 )  [static]
```

Erzeugt eine Animation namens filename.type aus den Einzelbildern in pics.

Falls type = "gif", wird eine animierte gif-Datei erzeugt; falls type = "mpg" eine MPEG-Datei. delay ist bei gif-Animtionen der Zeitabstand zwischen zwei Bildern.

Definiert in Zeile 168 der Datei drawing.cc.

4.2.3.16 operator=() [1/2]

```
Drawing & Drawing::operator= (
            const Drawing & d2 )
```

Zuweisungsoperator.

Definiert in Zeile 36 der Datei drawing.cc.

Benutzt DrawOps::changeSize(), DrawOps::getInstance() und img.

4.2.3.17 operator=() [2/2]

```
Drawing & Drawing::operator= (
            DrawColour colour )
```

Setzt das ganze Bild auf die Farbe `colour`.

Definiert in Zeile 46 der Datei drawing.cc.

4.2.3.18 savePGM()

```
void Drawing::savePGM (
            const std::string & filename ) const
```

Speichert das Bild im pgm-Format in der Datei `filename`.

Definiert in Zeile 62 der Datei drawing.cc.

4.2.3.19 savePNG()

```
void Drawing::savePNG (
            const std::string & filename ) const
```

Speichert das Bild im png-Format in der Datei `filename`.

Definiert in Zeile 68 der Datei drawing.cc.

Wird benutzt von maindraw().

4.2.3.20 savePPM()

```
void Drawing::savePPM (
            const std::string & filename ) const
```

Speichert das Bild im ppm-Format in der Datei `filename`.

Definiert in Zeile 65 der Datei drawing.cc.

4.2.3.21 setSleepTime()

```
void Drawing::setSleepTime (
            int milli )  [inline]
```

Setzt die Wartezeit für verzögertes Malen.

Definiert in Zeile 99 der Datei drawing.h.

Benutzt sleepMilli.

4.2.3.22 setZoom()

```
void Drawing::setZoom (
            int z )
```

Ändert den Zoomfaktor für des Fensters, falls es angezeigt wird.

Definiert in Zeile 24 der Datei drawing.cc.

Benutzt DrawOps::getInstance(), DrawWindow::getInstance() und DrawOps::setZoom().

Wird benutzt von maindraw().

4.2.3.23 show()

```
void Drawing::show ( ) const
```

Zeigt dieses Bild im Graphikfenster an, ersetzt bisherigen Fensterinhalt.

Definiert in Zeile 30 der Datei drawing.cc.

Benutzt DrawOps::getInstance() und DrawOps::makeActive().

Wird benutzt von maindraw().

4.2.3.24 update()

```
void Drawing::update ( ) const
```

Informiert das Zeichenfenster über Veränderungen.

Definiert in Zeile 33 der Datei drawing.cc.

Benutzt DrawOps::getInstance() und DrawOps::updateIfActive().

4.2.4 Dokumentation der Datenelemente

4.2.4.1 defaultSleepMilli

```
const int Drawing::defaultSleepMilli = 10  [static], [private]
```

Definiert in Zeile 26 der Datei drawing.h.

4.2.4.2 img

```
QImage Drawing::img  [private]
```

Definiert in Zeile 24 der Datei drawing.h.

Wird benutzt von Drawing(), getHeight(), getImage(), getWidth() und operator=().

4.2.4.3 imgMutex

```
QMutex Drawing::imgMutex  [mutable], [private]
```

Definiert in Zeile 27 der Datei drawing.h.

Wird benutzt von getMutex().

4.2.4.4 sleepMilli

```
int Drawing::sleepMilli  [private]
```

Definiert in Zeile 25 der Datei drawing.h.

Wird benutzt von setSleepTime().

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

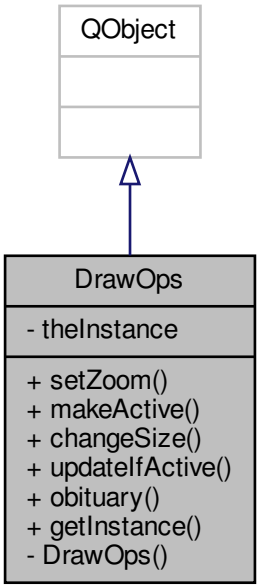
- drawing.h
- drawing.cc

4.3 DrawOps Klassenreferenz

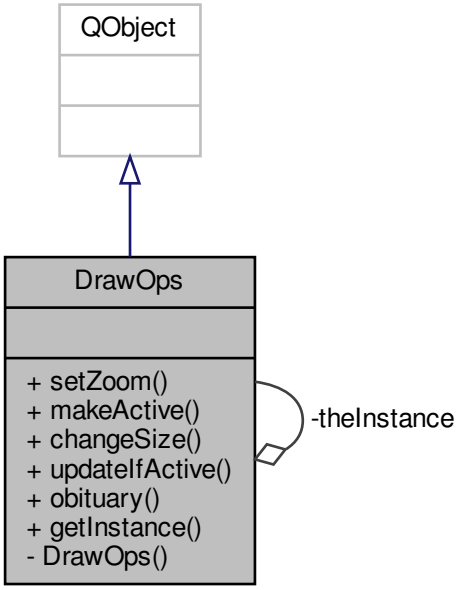
Klasse zur Bereitstellung von Zeichenoperationen und zur Kommunikation mit dem Zeichenfenster.

```
#include <drawops.h>
```

Klassendiagramm für DrawOps:



Zusammengehörigkeiten von DrawOps:



Signale

- void sigZoom (int z)
- void sigActivate (const Drawing ∗drw, QWaitCondition ∗madeActive)
- void sigSize (QWaitCondition ∗changedSize)
- void sigUpdate ()
- void sigDead (const Drawing ∗drw, QWaitCondition ∗forDeletion)

Öffentliche Methoden

- void setZoom (int z)

  Ändert den Zoomfaktor für des Fensters.
- void makeActive (const Drawing ∗img)

  Aktiviert eine Zeichnung, d.h. Anzeige im Graphikfenster.
- void changeSize ()

  Informiert das Zeichenfenster über eine geänderte Größe.
- void updateIfActive (const Drawing ∗drw)

  Informiert das Zeichenfenster über Veränderungen, falls `pm` *gerade angezeigt wird.*
- void obituary (const Drawing ∗drw)

  Informiert das Zeichenfenster über das Ableben einer Drawing-Instanz.

Öffentliche, statische Methoden

- static DrawOps ∗ getInstance ()

  Liefert Zeiger auf die einzige DrawOps-Instanz.

Private Methoden

- DrawOps ()

Statische, private Attribute

- static DrawOps * theInstance = nullptr

4.3.1 Ausführliche Beschreibung

Klasse zur Bereitstellung von Zeichenoperationen und zur Kommunikation mit dem Zeichenfenster.

Verwendet Singleton-Pattern

Definiert in Zeile 18 der Datei drawops.h.

4.3.2 Beschreibung der Konstruktoren und Destruktoren

4.3.2.1 DrawOps()

```
DrawOps::DrawOps ( )  [inline], [private]
```

Definiert in Zeile 25 der Datei drawops.h.

Wird benutzt von getInstance().

4.3.3 Dokumentation der Elementfunktionen

4.3.3.1 changeSize()

```
void DrawOps::changeSize ( )
```

Informiert das Zeichenfenster über eine geänderte Größe.

Definiert in Zeile 30 der Datei drawops.cc.

Benutzt sigSize().

Wird benutzt von Drawing::loadImage() und Drawing::operator=().

4.3.3.2 getInstance()

```
DrawOps * DrawOps::getInstance ( )  [static]
```

Liefert Zeiger auf die einzige DrawOps-Instanz.

Definiert in Zeile 14 der Datei drawops.cc.

Benutzt DrawOps() und theInstance.

Wird benutzt von Drawing::loadImage(), main(), Drawing::operator=(), Drawing::setZoom(), Drawing::show(), Drawing::update() und Drawing::∼Drawing().

4.3.3.3 makeActive()

```
void DrawOps::makeActive (
            const Drawing * img )
```

Aktiviert eine Zeichnung, d.h. Anzeige im Graphikfenster.

Definiert in Zeile 21 der Datei drawops.cc.

Benutzt sigActivate().

Wird benutzt von Drawing::show().

4.3.3.4 obituary()

```
void DrawOps::obituary (
            const Drawing * drw )
```

Informiert das Zeichenfenster über das Ableben einer Drawing-Instanz.

Blockiert, bis das Fenster das Signal bestätigt hat.

Definiert in Zeile 45 der Datei drawops.cc.

Benutzt sigDead().

Wird benutzt von Drawing::∼Drawing().

4.3.3.5 setZoom()

```
void DrawOps::setZoom (
            int z )  [inline]
```

Ändert den Zoomfaktor für des Fensters.

Definiert in Zeile 32 der Datei drawops.h.

Benutzt sigZoom().

Wird benutzt von Drawing::setZoom().

4.3.3.6 sigActivate

```
void DrawOps::sigActivate (
            const Drawing * drw,
            QWaitCondition * madeActive )  [signal]
```

Wird benutzt von main() und makeActive().

4.3.3.7 sigDead

```
void DrawOps::sigDead (
            const Drawing * drw,
            QWaitCondition * forDeletion )  [signal]
```

Wird benutzt von main() und obituary().

4.3.3.8 sigSize

```
void DrawOps::sigSize (
            QWaitCondition * changedSize )  [signal]
```

Wird benutzt von changeSize() und main().

4.3.3.9 sigUpdate

```
void DrawOps::sigUpdate ( )  [signal]
```

Wird benutzt von main() und updateIfActive().

4.3.3.10 sigZoom

```
void DrawOps::sigZoom (
            int z )  [signal]
```

Wird benutzt von main() und setZoom().

4.3.3.11 updateIfActive()

```
void DrawOps::updateIfActive (
            const Drawing * drw )
```

Informiert das Zeichenfenster über Veränderungen, falls `pm` gerade angezeigt wird.

Definiert in Zeile 39 der Datei drawops.cc.

Benutzt DrawWindow::getInstance() und sigUpdate().

Wird benutzt von Drawing::update().

4.3.4 Dokumentation der Datenelemente

4.3.4.1 theInstance

```
DrawOps * DrawOps::theInstance = nullptr  [static], [private]
```

Definiert in Zeile 23 der Datei drawops.h.

Wird benutzt von getInstance().

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

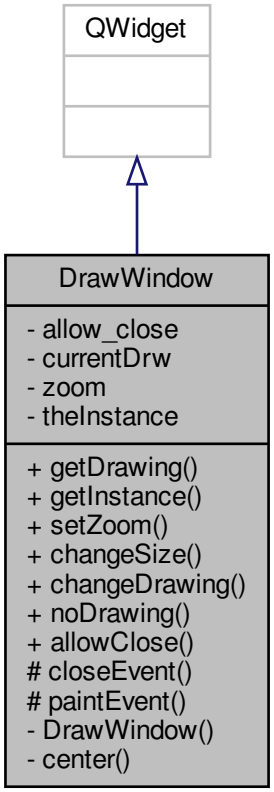
- drawops.h
- drawops.cc

4.4 DrawWindow Klassenreferenz

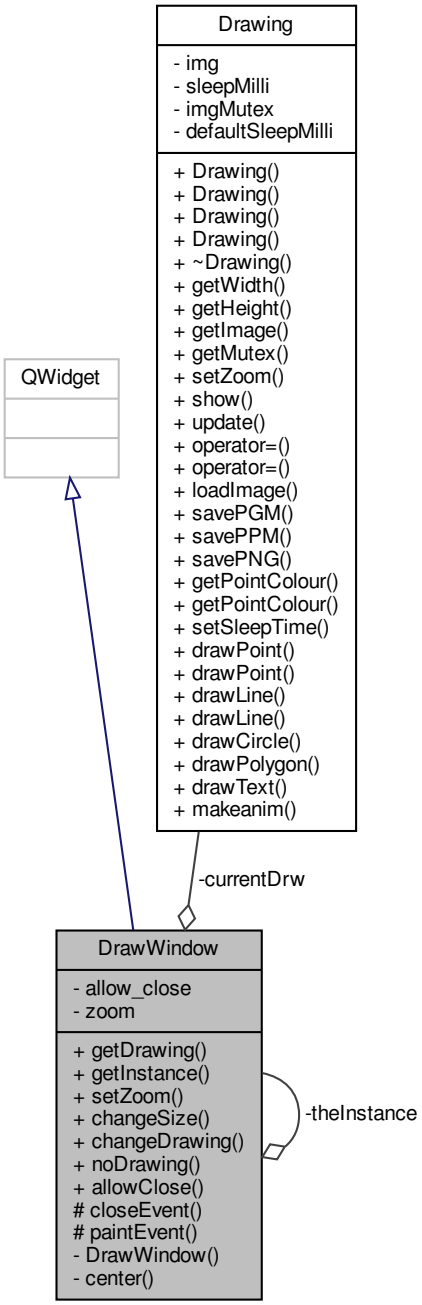
Das Fenster zum Anzeigen eines Bildes, verwendet Singleton-Pattern.

```
#include <drawwindow.h>
```

Klassendiagramm für DrawWindow:



Zusammengehörigkeiten von DrawWindow:



Öffentliche Slots

- void setZoom (int z)

  Ändert den Zoomfaktor für die Anzeige.
- void changeSize (QWaitCondition ∗changedSize)

  Passt das Fenster an eine geänderte Bildgröße an.
- void changeDrawing (const Drawing ∗drw, QWaitCondition ∗madeActive)

Verknüpft das Fenster mit einer anderen Zeichnung.

- void noDrawing (const Drawing ∗drw, QWaitCondition ∗forDeletion)

  Löst die Verknüpfung des Fensters mit einer Zeichnung, wenn diese gerade angezeigt wird.
- void allowClose (QWaitCondition ∗closeWindow)

  Aktiviert die Möglichkeit zum Schließen des Fensters.

Öffentliche Methoden

- const Drawing ∗ getDrawing ()

  Liefert Zeiger auf die aktuell angezeigte Zeichnung.

Öffentliche, statische Methoden

- static DrawWindow ∗ getInstance ()

  Liefert Zeiger auf das einzige DrawWindow.

Geschützte Methoden

- virtual void closeEvent (QCloseEvent ∗ce) override
- virtual void paintEvent (QPaintEvent ∗pe) override

Private Methoden

- DrawWindow ()

  Default-Konstruktor.
- void center ()

  Fenster auf Bildschirm zentrieren.

Private Attribute

- QWaitCondition ∗ allow_close = nullptr

  Signalisiert dem IOThread, dass das Fenster geschlossen wurde.
- const Drawing ∗ currentDrw

  Das momentan angezeigte Drawing.
- int zoom

  Zoomfaktor.

Statische, private Attribute

- static DrawWindow ∗ theInstance = nullptr

  Die DrawWindow Instanz.

4.4.1 Ausführliche Beschreibung

Das Fenster zum Anzeigen eines Bildes, verwendet Singleton-Pattern.

Definiert in Zeile 23 der Datei drawwindow.h.

4.4.2 Beschreibung der Konstruktoren und Destruktoren

4.4.2.1 DrawWindow()

```
DrawWindow::DrawWindow ( )  [inline], [private]
```

Default-Konstruktor.

Definiert in Zeile 38 der Datei drawwindow.h.

Benutzt center().

Wird benutzt von getInstance().

4.4.3 Dokumentation der Elementfunktionen

4.4.3.1 allowClose

```
void DrawWindow::allowClose (
            QWaitCondition * closeWindow )  [slot]
```

Aktiviert die Möglichkeit zum Schließen des Fensters.

Definiert in Zeile 112 der Datei drawwindow.cc.

Benutzt allow_close.

Wird benutzt von main().

4.4.3.2 center()

```
void DrawWindow::center ( )  [private]
```

Fenster auf Bildschirm zentrieren.

Definiert in Zeile 15 der Datei drawwindow.cc.

Wird benutzt von changeDrawing() und DrawWindow().

4.4.3.3 changeDrawing

```
void DrawWindow::changeDrawing (
            const Drawing * drw,
            QWaitCondition * madeActive )  [slot]
```

Verknüpft das Fenster mit einer anderen Zeichnung.

Definiert in Zeile 87 der Datei drawwindow.cc.

Benutzt center(), currentDrw, Drawing::getHeight(), Drawing::getWidth() und zoom.

Wird benutzt von main().

4.4.3.4 changeSize

```
void DrawWindow::changeSize (
            QWaitCondition * changedSize )  [slot]
```

Passt das Fenster an eine geänderte Bildgröße an.

Definiert in Zeile 77 der Datei drawwindow.cc.

Benutzt currentDrw, Drawing::getHeight(), Drawing::getWidth() und zoom.

Wird benutzt von main().

4.4.3.5 closeEvent()

```
void DrawWindow::closeEvent (
            QCloseEvent * ce )  [override], [protected], [virtual]
```

Definiert in Zeile 27 der Datei drawwindow.cc.

Benutzt allow_close.

4.4.3.6 getDrawing()

```
const Drawing* DrawWindow::getDrawing ( )  [inline]
```

Liefert Zeiger auf die aktuell angezeigte Zeichnung.

Definiert in Zeile 61 der Datei drawwindow.h.

Benutzt currentDrw.

4.4.3.7 getInstance()

```
DrawWindow * DrawWindow::getInstance ( )  [static]
```

Liefert Zeiger auf das einzige DrawWindow.

Definiert in Zeile 55 der Datei drawwindow.cc.

Benutzt DrawWindow() und theInstance.

Wird benutzt von main(), Drawing::setZoom() und DrawOps::updateIfActive().

4.4.3.8 noDrawing

```
void DrawWindow::noDrawing (
            const Drawing * drw,
            QWaitCondition * forDeletion )  [slot]
```

Löst die Verknüpfung des Fensters mit einer Zeichnung, wenn diese gerade angezeigt wird.

Definiert in Zeile 102 der Datei drawwindow.cc.

Benutzt currentDrw.

Wird benutzt von main().

4.4.3.9 paintEvent()

```
void DrawWindow::paintEvent (
            QPaintEvent * pe )  [override], [protected], [virtual]
```

Definiert in Zeile 35 der Datei drawwindow.cc.

Benutzt currentDrw, Drawing::getImage(), Drawing::getMutex() und zoom.

4.4.3.10 setZoom

```
void DrawWindow::setZoom (
            int z )  [slot]
```

Ändert den Zoomfaktor für die Anzeige.

Definiert in Zeile 62 der Datei drawwindow.cc.

Benutzt currentDrw, Drawing::getHeight(), Drawing::getWidth() und zoom.

Wird benutzt von main().

4.4.4 Dokumentation der Datenelemente

4.4.4.1 allow_close

```
QWaitCondition* DrawWindow::allow_close = nullptr  [private]
```

Signalisiert dem IOThread, dass das Fenster geschlossen wurde.

Definiert in Zeile 31 der Datei drawwindow.h.

Wird benutzt von allowClose() und closeEvent().

4.4.4.2 currentDrw

```
const Drawing* DrawWindow::currentDrw  [private]
```

Das momentan angezeigte Drawing.

Definiert in Zeile 33 der Datei drawwindow.h.

Wird benutzt von changeDrawing(), changeSize(), getDrawing(), noDrawing(), paintEvent() und setZoom().

4.4.4.3 theInstance

```
DrawWindow * DrawWindow::theInstance = nullptr  [static], [private]
```

Die DrawWindow Instanz.

Definiert in Zeile 29 der Datei drawwindow.h.

Wird benutzt von getInstance().

4.4.4.4 zoom

```
int DrawWindow::zoom  [private]
```

Zoomfaktor.

Definiert in Zeile 35 der Datei drawwindow.h.

Wird benutzt von changeDrawing(), changeSize(), paintEvent() und setZoom().

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

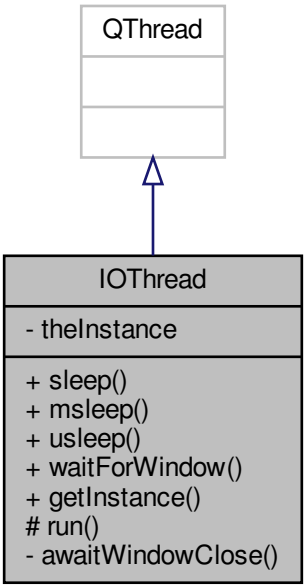
- drawwindow.h
- drawwindow.cc

4.5 IOThread Klassenreferenz

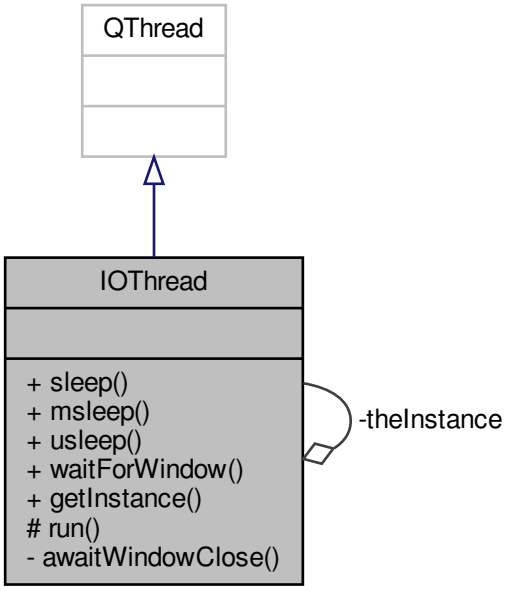
Der Thread, der in der Kommandoshell läuft; verwendet Singleton-Pattern.

```
#include <iothread.h>
```

Klassendiagramm für IOThread:



Zusammengehörigkeiten von IOThread:



Signale

- void sigAllowClose (QWaitCondition *closeWindow)

  Signalisiert dem Fenster das manuelle Schließen zuzulassen.

Öffentliche, statische Methoden

- static void sleep (int i)

  Schlafe für `i` Sekunden.
- static void msleep (int i)

  Schlafe für `i` Milliosekunden.
- static void usleep (int i)

  Schlafe für `i` Mikrosekunden.
- static void waitForWindow (unsigned time=UINT_MAX)

  Veranlasst die IOThread Instanz auf das schließen des Fensters zu warten.
- static IOThread * getInstance ()

  Gibt die IOThread Instanz zurück.

Geschützte Methoden

- virtual void run () override

  Ruft die Methode maindraw auf.

Private Methoden

- void awaitWindowClose (unsigned time=UINT_MAX)

  Veranlasst die IOThread Instanz auf das schließen des Fensters zu warten.

Statische, private Attribute

- static IOThread * theInstance

  Die IOThread Instanz.

4.5.1 Ausführliche Beschreibung

Der Thread, der in der Kommandoshell läuft; verwendet Singleton-Pattern.

Definiert in Zeile 19 der Datei iothread.h.

4.5.2 Dokumentation der Elementfunktionen

4.5.2.1 awaitWindowClose()

```
void IOThread::awaitWindowClose (
            unsigned time = UINT_MAX )  [inline], [private]
```

Veranlasst die IOThread Instanz auf das schließen des Fensters zu warten.

Definiert in Zeile 28 der Datei iothread.h.

Benutzt sigAllowClose().

Wird benutzt von waitForWindow().

4.5.2.2 getInstance()

```
static IOThread* IOThread::getInstance ( )  [inline], [static]
```

Gibt die IOThread Instanz zurück.

Definiert in Zeile 58 der Datei iothread.h.

Benutzt theInstance.

Wird benutzt von main() und waitForWindow().

4.5.2.3 msleep()

```
static void IOThread::msleep (
            int i )  [inline], [static]
```

Schlafe für i Milliosekunden.

Definiert in Zeile 47 der Datei iothread.h.

Wird benutzt von Drawing::drawCircle(), Drawing::drawLine(), Drawing::drawPoint() und Drawing::drawPolygon().

4.5.2.4 run()

```
virtual void IOThread::run ( )  [inline], [override], [protected], [virtual]
```

Ruft die Methode maindraw auf.

Definiert in Zeile 67 der Datei iothread.h.

Benutzt maindraw().

4.5.2.5 sigAllowClose

```
void IOThread::sigAllowClose (
            QWaitCondition * closeWindow )  [signal]
```

Signalisiert dem Fenster das manuelle Schließen zuzulassen.

Wird benutzt von awaitWindowClose() und main().

4.5.2.6 sleep()

```
static void IOThread::sleep (
            int i )  [inline], [static]
```

Schlafe für i Sekunden.

Definiert in Zeile 45 der Datei iothread.h.

4.5.2.7 usleep()

```
static void IOThread::usleep (
            int i )  [inline], [static]
```

Schlafe für i Mikrosekunden.

Definiert in Zeile 49 der Datei iothread.h.

4.5.2.8 waitForWindow()

```
static void IOThread::waitForWindow (
            unsigned time = UINT_MAX )  [inline], [static]
```

Veranlasst die IOThread Instanz auf das schließen des Fensters zu warten.

Definiert in Zeile 52 der Datei iothread.h.

Benutzt awaitWindowClose() und getInstance().

Wird benutzt von maindraw().

4.5.3 Dokumentation der Datenelemente

4.5.3.1 theInstance

```
IOThread* IOThread::theInstance  [static], [private]
```

Die IOThread Instanz.

Definiert in Zeile 25 der Datei iothread.h.

Wird benutzt von getInstance().

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

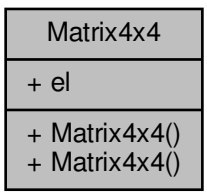
- iothread.h

4.6 Matrix4x4 Klassenreferenz

4x4-Matrix, dient auch als Transformationsmatrix für 3D-Vektoren mit homogenen Koordinaten

```
#include <matrix4x4.h>
```

Zusammengehörigkeiten von Matrix4x4:



Öffentliche Methoden

- Matrix4x4 ()
  Nullmatrix Konstruktor.
- Matrix4x4 (double m[4][4])
  Matrix aus 2D-Array.

Öffentliche Attribute

- double el [4][4]
  Matrixelemente.

4.6.1 Ausführliche Beschreibung

4x4-Matrix, dient auch als Transformationsmatrix für 3D-Vektoren mit homogenen Koordinaten

Definiert in Zeile 13 der Datei matrix4x4.h.

4.6.2 Beschreibung der Konstruktoren und Destruktoren

4.6.2.1 Matrix4x4() [1/2]

```
Matrix4x4::Matrix4x4 ( )
```

Nullmatrix Konstruktor.

4.6.2.2 Matrix4x4() [2/2]

```
Matrix4x4::Matrix4x4 (
            double m[4][4] )
```

Matrix aus 2D-Array.

4.6.3 Dokumentation der Datenelemente

4.6.3.1 el

```
double Matrix4x4::el[4][4]
```

Matrixelemente.

Definiert in Zeile 17 der Datei matrix4x4.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

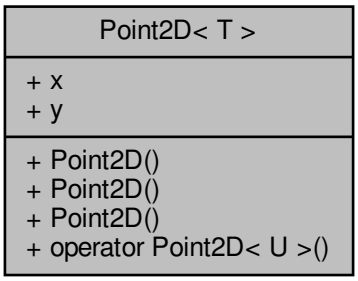
- matrix4x4.h

4.7 Point2D< T > Template-Klassenreferenz

Punkt in der Ebene.

`#include <point2d.h>`

Zusammengehörigkeiten von Point2D< T >:



Öffentliche Methoden

- Point2D ()
  Default-Konstruktor: Punkt (0, 0)
- Point2D (T xx, T yy)
  Konstruktor für einen Punkt (x, y)
- Point2D (const Point2D< T > &p2)
  Kopier-Konstruktor.
- template<class U >
  operator Point2D< U > () const
  Typumwandlung.

Öffentliche Attribute

- T x
  x-Koordinate
- T y
  y-Koordinate

4.7.1 Ausführliche Beschreibung

template<class T = int>
class Point2D< T >

Punkt in der Ebene.

Kann auch für Vektoren in der Ebene verwendet werden.

Definiert in Zeile 14 der Datei point2d.h.

4.7.2 Beschreibung der Konstruktoren und Destruktoren

4.7.2.1 Point2D() [1/3]

```
template<class T = int>
Point2D< T >::Point2D ( )  [inline]
```

Default-Konstruktor: Punkt (0, 0)

Definiert in Zeile 24 der Datei point2d.h.

4.7.2.2 Point2D() [2/3]

```
template<class T = int>
Point2D< T >::Point2D (
            T xx,
            T yy )  [inline]
```

Konstruktor für einen Punkt (x, y)

Definiert in Zeile 27 der Datei point2d.h.

4.7.2.3 Point2D() [3/3]

```
template<class T = int>
Point2D< T >::Point2D (
            const Point2D< T > & p2 )  [inline]
```

Kopier-Konstruktor.

Definiert in Zeile 30 der Datei point2d.h.

4.7.3 Dokumentation der Elementfunktionen

4.7.3.1 operator Point2D< U >()

```
template<class T = int>
template<class U >
Point2D< T >::operator Point2D< U > ( ) const  [inline]
```

Typumwandlung.

Definiert in Zeile 33 der Datei point2d.h.

Benutzt Point2D< T >::x und Point2D< T >::y.

4.7.4 Dokumentation der Datenelemente

4.7.4.1 x

```
template<class T = int>
T Point2D< T >::x
```

x-Koordinate

Definiert in Zeile 18 der Datei point2d.h.

Wird benutzt von Drawing::drawCircle(), Drawing::drawLine(), Drawing::drawPoint(), Drawing::drawText(), Drawing::getPointColour(), norm(), Point2D< T >::operator Point2D< U >(), operator∗(), operator+(), operator-(), operator/(), operator>>() und round().

4.7.4.2 y

```
template<class T = int>
T Point2D< T >::y
```

y-Koordinate

Definiert in Zeile 21 der Datei point2d.h.

Wird benutzt von Drawing::drawCircle(), Drawing::drawLine(), Drawing::drawPoint(), Drawing::drawText(), Drawing::getPointColour(), norm(), Point2D< T >::operator Point2D< U >(), operator∗(), operator+(), operator-(), operator/(), operator>>() und round().

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

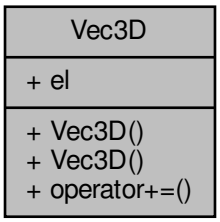
- point2d.h

4.8 Vec3D Klassenreferenz

Koordinaten eines 3D-Vektors.

```
#include <vec3d.h>
```

Zusammengehörigkeiten von Vec3D:



Öffentliche Methoden

- Vec3D ()

  Default-Konstruktor: Vektor (0, 0, 0)
- Vec3D (double x, double y, double z)

  Konstruktor für einen Vektor (x, y, z)
- Vec3D & operator+= (const Vec3D &v)

  Additionszuweisung.

Öffentliche Attribute

- double el [3]

  x-, y-, und z-Koordinate

4.8.1 Ausführliche Beschreibung

Koordinaten eines 3D-Vektors.

Definiert in Zeile 12 der Datei vec3d.h.

4.8.2 Beschreibung der Konstruktoren und Destruktoren

4.8.2.1 Vec3D() [1/2]

```
Vec3D::Vec3D ( )
```

Default-Konstruktor: Vektor (0, 0, 0)

4.8.2.2 Vec3D() [2/2]

```
Vec3D::Vec3D (
            double x,
            double y,
            double z )
```

Konstruktor für einen Vektor (x, y, z)

4.8.3 Dokumentation der Elementfunktionen

4.8.3.1 operator+=()

```
Vec3D& Vec3D::operator+= (
            const Vec3D & v )
```

Additionszuweisung.

4.8.4 Dokumentation der Datenelemente

4.8.4.1 el

```
double Vec3D::el[3]
```

x-, y-, und z-Koordinate

Definiert in Zeile 16 der Datei vec3d.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

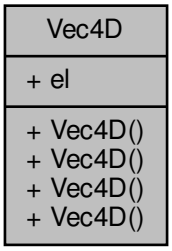
- vec3d.h

4.9 Vec4D Klassenreferenz

Homogene Koordinaten eines 3D-Vektors oder ein 4D-Vektor.

```
#include <vec4d.h>
```

Zusammengehörigkeiten von Vec4D:



Öffentliche Methoden

- Vec4D ()

  Default-Konstruktor: Vektor (0, 0, 0, 0)
- Vec4D (double x, double y, double z)

  Konstruktor für einen Vektor (x, y, z, 1)
- Vec4D (double x, double y, double z, double w)

  Konstruktor für einen Vektor (x, y, z, w)
- Vec4D (const Vec3D &v)

  Konstruktor für einen 4D-Vektor aus einem 3D-Vektor.

Öffentliche Attribute

- double el [4]
  x-, y-, z-, und w-Koordinate

4.9.1 Ausführliche Beschreibung

Homogene Koordinaten eines 3D-Vektors oder ein 4D-Vektor.

Definiert in Zeile 13 der Datei vec4d.h.

4.9.2 Beschreibung der Konstruktoren und Destruktoren

4.9.2.1 Vec4D() [1/4]

```
Vec4D::Vec4D ( )
```

Default-Konstruktor: Vektor (0, 0, 0, 0)

4.9.2.2 Vec4D() [2/4]

```
Vec4D::Vec4D (
            double x,
            double y,
            double z )
```

Konstruktor für einen Vektor (x, y, z, 1)

4.9.2.3 Vec4D() [3/4]

```
Vec4D::Vec4D (
            double x,
            double y,
            double z,
            double w )
```

Konstruktor für einen Vektor (x, y, z, w)

4.9.2.4 Vec4D() [4/4]

```
Vec4D::Vec4D (
            const Vec3D & v )
```

Konstruktor für einen 4D-Vektor aus einem 3D-Vektor.

4.9.3 Dokumentation der Datenelemente

4.9.3.1 el

```
double Vec4D::el[4]
```

x-, y-, z-, und w-Koordinate

Definiert in Zeile 17 der Datei vec4d.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

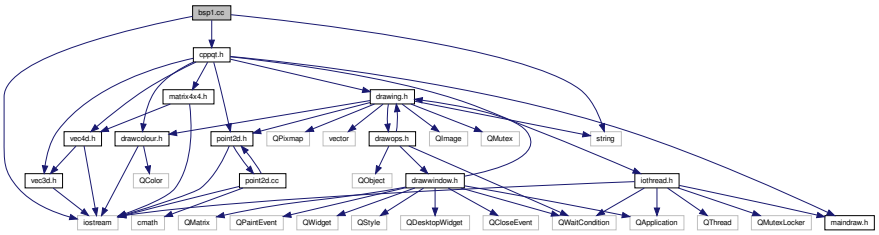
- vec4d.h

5 Datei-Dokumentation

5.1 bsp1.cc-Dateireferenz

```
#include <iostream>
#include <string>
#include <cppqt.h>
```

Include-Abhängigkeitsdiagramm für bsp1.cc:



Funktionen

- int maindraw ()

  Beispiel-Programm, das die wichtigstes Funktionen zum Zeichnen von Bildern enthält.

5.1.1 Ausführliche Beschreibung

Autor

Holger Arndt, Martin Galgon

Version

0.3.3

Datum

11.11.2016

5.1.2 Dokumentation der Funktionen

5.1.2.1 maindraw()

```
int maindraw ( )
```

Beispiel-Programm, das die wichtigstes Funktionen zum Zeichnen von Bildern enthält.

Die Funktion `maindraw` dient als Ersatz für das normale Hauptprogramm `main`.

Definiert in Zeile 15 der Datei bsp1.cc.

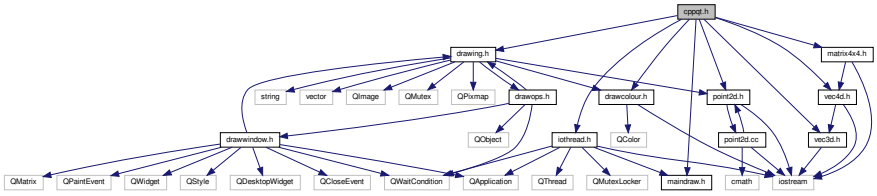
Benutzt Drawing::drawLine(), Drawing::drawPoint(), Drawing::loadImage(), Drawing::savePNG(), Drawing::setZoom(), Drawing::show() und IOThread::waitForWindow().

Wird benutzt von IOThread::run().

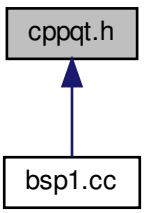
5.2 cppqt.h-Dateireferenz

```
#include <drawcolour.h>
#include <drawing.h>
#include <iothread.h>
#include <maindraw.h>
#include <point2d.h>
#include <vec3d.h>
#include <vec4d.h>
#include <matrix4x4.h>
```

Include-Abhängigkeitsdiagramm für cppqt.h:



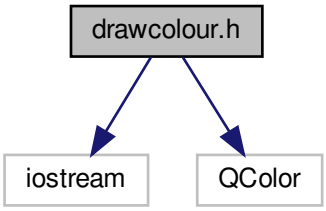
Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



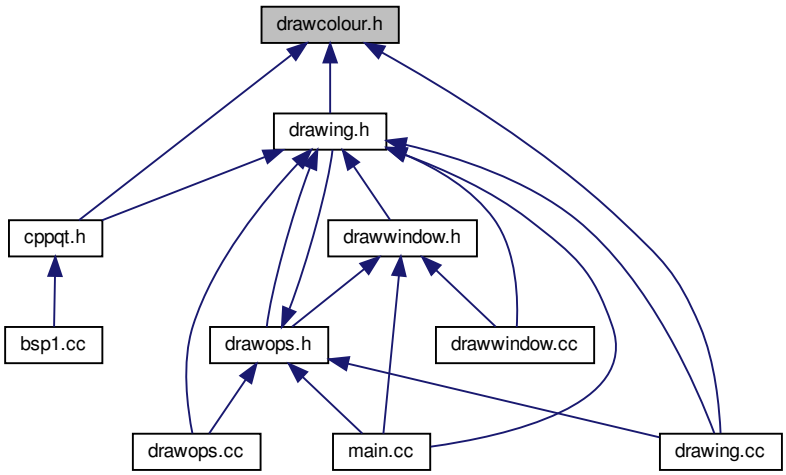
5.3 drawcolour.h-Dateireferenz

```
#include <iostream>
#include <QColor>
```

Include-Abhängigkeitsdiagramm für drawcolour.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class DrawColour

  Farbklasse.

Funktionen

- std::ostream & operator<< (std::ostream &os, const DrawColour &dc)

  Ausgabe der Farbe in der Form (0,0,255)
- std::istream & operator>> (std::istream &is, DrawColour &dc)

  Eingabe der Farbe in der Form (0,0,255)

5.3.1 Ausführliche Beschreibung

Autor

Holger Arndt, Sebastian Birk

Version

0.3

Datum

16.09.2015

5.3.2 Dokumentation der Funktionen

5.3.2.1 operator<<()

```
std::ostream& operator<< (
            std::ostream & os,
            const DrawColour & dc )
```

Ausgabe der Farbe in der Form (0,0,255)

5.3.2.2 operator>>()

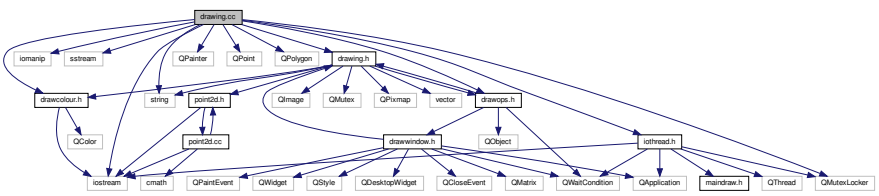
```
std::istream& operator>> (
            std::istream & is,
            DrawColour & dc )
```

Eingabe der Farbe in der Form (0,0,255)

5.4 drawing.cc-Dateireferenz

```
#include <iostream>
#include <iomanip>
#include <sstream>
#include <string>
#include <QMutexLocker>
#include <QPainter>
#include <QPoint>
#include <QPolygon>
#include "drawcolour.h"
#include "drawing.h"
#include "drawops.h"
#include "iothread.h"
```

Include-Abhängigkeitsdiagramm für drawing.cc:



5.4.1 Ausführliche Beschreibung

Autor

Holger Arndt, Sebastian Birk, Martin Galgon

Version

0.2.2

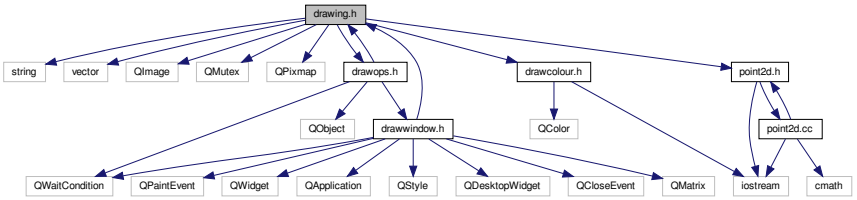
Datum

16.09.2015

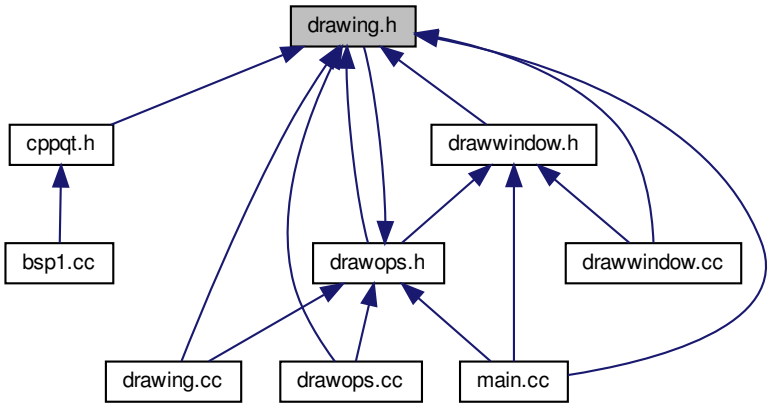
5.5 drawing.h-Dateireferenz

```
#include <string>
#include <vector>
#include <QImage>
#include <QMutex>
#include <QPixmap>
#include "drawcolour.h"
#include "drawops.h"
#include "point2d.h"
```

Include-Abhängigkeitsdiagramm für drawing.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class Drawing

  Ein Bild.

5.5.1 Ausführliche Beschreibung

Autor

Holger Arndt, Sebastian Birk, Martin Galgon

Version

0.2.3

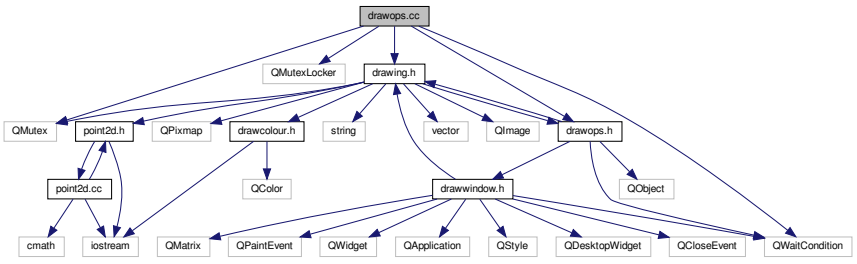
Datum

16.09.2015

5.6 drawops.cc-Dateireferenz

```
#include <QMutex>
#include <QMutexLocker>
#include <QWaitCondition>
#include "drawing.h"
#include "drawops.h"
```

Include-Abhängigkeitsdiagramm für drawops.cc:



5.6.1 Ausführliche Beschreibung

Autor

Holger Arndt, Martin Galgon

Version

0.7

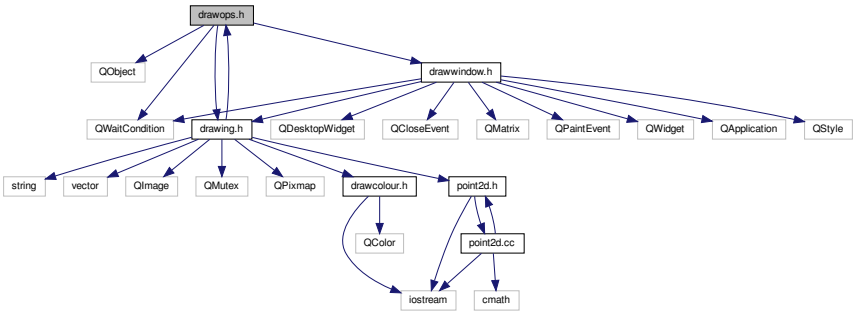
Datum

16.09.2015

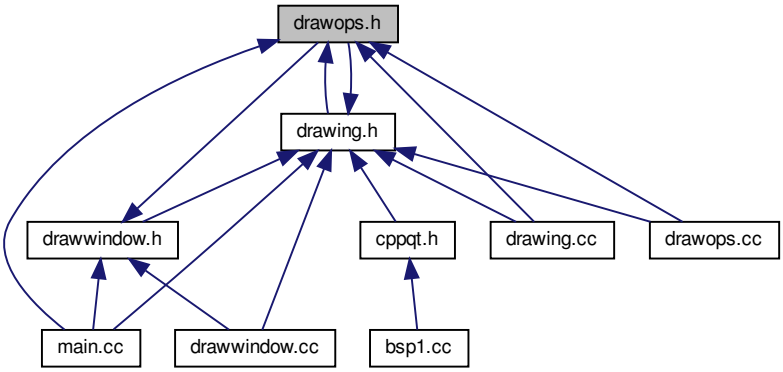
5.7 drawops.h-Dateireferenz

```
#include <QObject>
#include <QWaitCondition>
#include "drawing.h"
#include "drawwindow.h"
```

Include-Abhängigkeitsdiagramm für drawops.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class DrawOps

  Klasse zur Bereitstellung von Zeichenoperationen und zur Kommunikation mit dem Zeichenfenster.

5.7.1 Ausführliche Beschreibung

Autor

Holger Arndt, Martin Galgon

Version

0.7

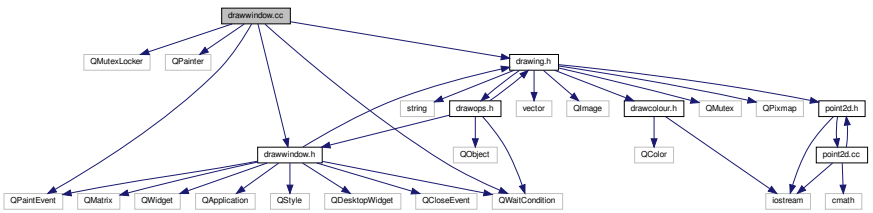
Datum

16.09.2015

5.8 drawwindow.cc-Dateireferenz

```
#include <QMutexLocker>
#include <QPainter>
#include <QPaintEvent>
#include <QWaitCondition>
#include "drawing.h"
#include "drawwindow.h"
```

Include-Abhängigkeitsdiagramm für drawwindow.cc:



5.8.1 Ausführliche Beschreibung

Autor

Holger Arndt, Martin Galgon

Version

0.3

Datum

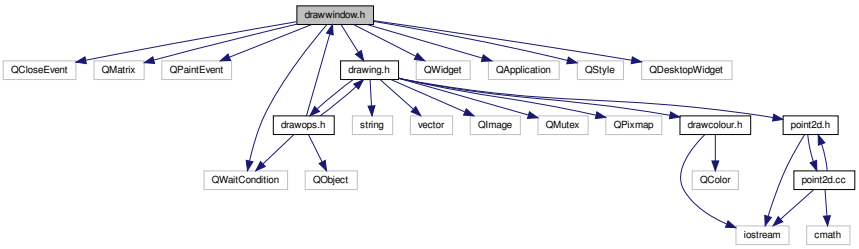
11.11.2016

5.9 drawwindow.h-Dateireferenz

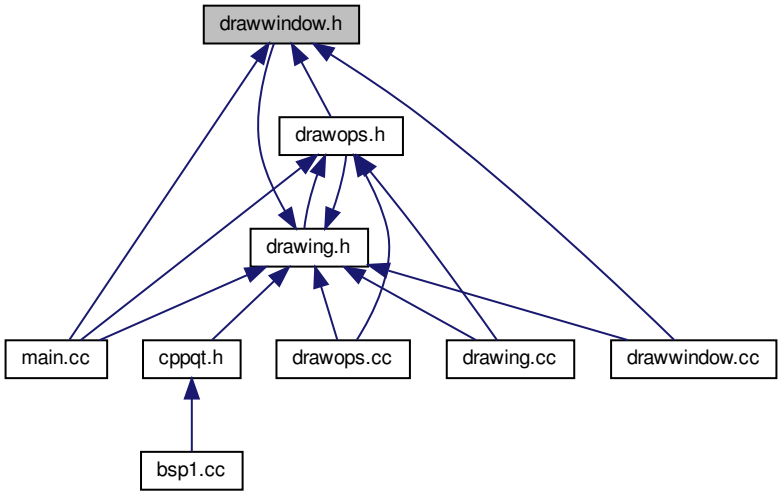
```
#include <QCloseEvent>
#include <QMatrix>
#include <QPaintEvent>
#include <QWaitCondition>
#include <QWidget>
#include "drawing.h"
#include <QApplication>
#include <QStyle>
```

```
#include <QDesktopWidget>
```

Include-Abhängigkeitsdiagramm für drawwindow.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class DrawWindow

  Das Fenster zum Anzeigen eines Bildes, verwendet Singleton-Pattern.

5.9.1 Ausführliche Beschreibung

Autor

Holger Arndt, Martin Galgon

Version

0.3

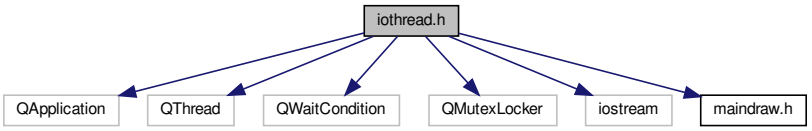
Datum

11.11.2016

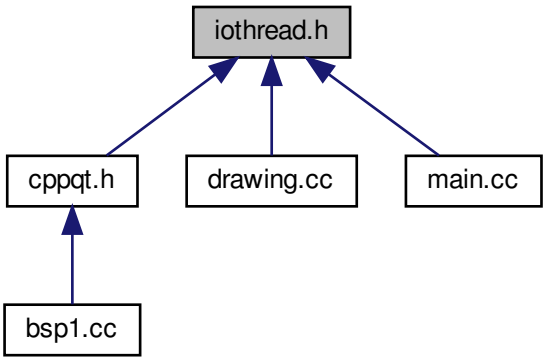
5.10 iothread.h-Dateireferenz

```
#include <QApplication>
#include <QThread>
#include <QWaitCondition>
#include <QMutexLocker>
#include <iostream>
#include "maindraw.h"
```

Include-Abhängigkeitsdiagramm für iothread.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class IOThread

  Der Thread, der in der Kommandoshell läuft; verwendet Singleton-Pattern.

5.10.1 Ausführliche Beschreibung

Autor

Holger Arndt, Martin Galgon

Version

0.4

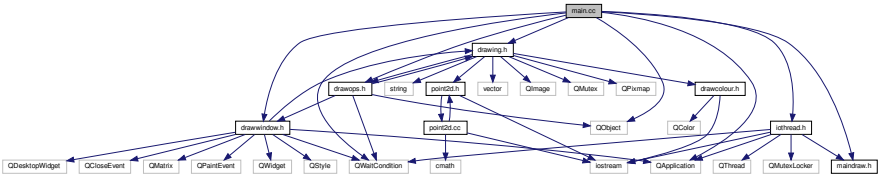
Datum

11.11.2016

5.11 main.cc-Dateireferenz

```
#include <QApplication>
#include <QObject>
#include <QWaitCondition>
#include "drawing.h"
#include "drawops.h"
#include "drawwindow.h"
#include "iothread.h"
#include "maindraw.h"
```

Include-Abhängigkeitsdiagramm für main.cc:



Funktionen

- int main (int argc, char ∗∗argv)

  Dieses Hauptprogramm ist für alle Draw-Programme fest.

5.11.1 Ausführliche Beschreibung

Autor

Holger Arndt, Martin Galgon

Version

0.4

Datum

11.11.2016

5.11.2 Dokumentation der Funktionen

5.11.2.1 main()

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

Dieses Hauptprogramm ist für alle Draw-Programme fest.

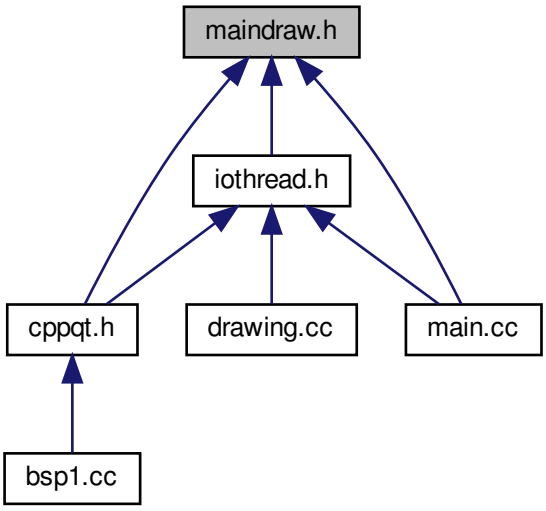
Als Pseudo-Hauptprogramm dient die Funktion `maindraw`.

Definiert in Zeile 17 der Datei main.cc.

Benutzt DrawWindow::allowClose(), DrawWindow::changeDrawing(), DrawWindow::changeSize(), DrawOps::getInstance(), IOThread::getInstance(), DrawWindow::getInstance(), DrawWindow::noDrawing(), DrawWindow::setZoom(), DrawOps::sigActivate(), IOThread::sigAllowClose(), DrawOps::sigDead(), DrawOps::sigSize(), DrawOps::sigUpdate() und DrawOps::sigZoom().

5.12 maindraw.h-Dateireferenz

Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Funktionen

- int maindraw ()

  Die Funktion `maindraw` dient als Ersatz für das normale Hauptprogramm `main`.

5.12.1 Ausführliche Beschreibung

Autor

Holger Arndt

Version

0.2

Datum

15.09.2015

5.12.2 Dokumentation der Funktionen

5.12.2.1 maindraw()

```
int maindraw ( )
```

Die Funktion `maindraw` dient als Ersatz für das normale Hauptprogramm `main`.

Sie läuft als eigener Thread.

Die Funktion `maindraw` dient als Ersatz für das normale Hauptprogramm `main`.

Definiert in Zeile 15 der Datei bsp1.cc.

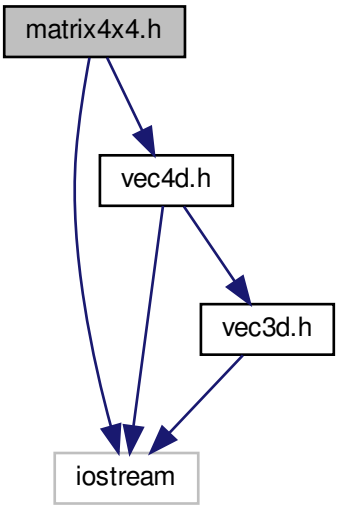
Benutzt Drawing::drawLine(), Drawing::drawPoint(), Drawing::loadImage(), Drawing::savePNG(), Drawing::setZoom(), Drawing::show() und IOThread::waitForWindow().

Wird benutzt von IOThread::run().

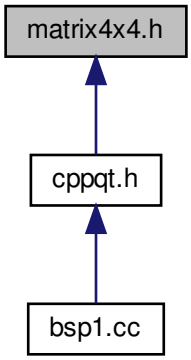
5.13 matrix4x4.h-Dateireferenz

```
#include <iostream>
#include <vec4d.h>
```

Include-Abhängigkeitsdiagramm für matrix4x4.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class Matrix4x4

  4x4-Matrix, dient auch als Transformationsmatrix für 3D-Vektoren mit homogenen Koordinaten

Funktionen

- std::ostream & operator<< (std::ostream &os, const Matrix4x4 &t)

  Zeilenweise Ausgabe einer Matrix.
- Matrix4x4 operator* (const Matrix4x4 &t1, const Matrix4x4 &t2)

  Matrix-Matrix-Multiplikation.
- Vec4D operator* (const Matrix4x4 &t, const Vec4D &v)

  Matrix-Vektor-Multiplikation.

5.13.1 Ausführliche Beschreibung

Autor

Holger Arndt, Sebastian Birk, Martin Galgon

Version

0.1

Datum

10.11.2016

5.13.2 Dokumentation der Funktionen

5.13.2.1 operator*() [1/2]

```
Matrix4x4 operator* (
            const Matrix4x4 & t1,
            const Matrix4x4 & t2 )
```

Matrix-Matrix-Multiplikation.

5.13.2.2 operator*() [2/2]

```
Vec4D operator* (
            const Matrix4x4 & t,
            const Vec4D & v )
```

Matrix-Vektor-Multiplikation.

5.13.2.3 operator<<()

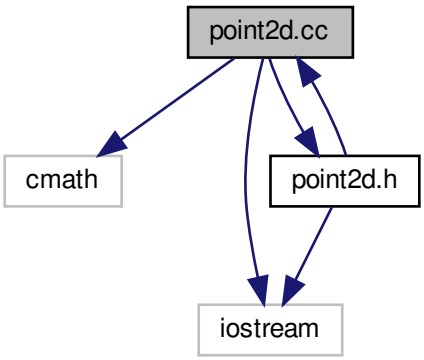
```
std::ostream& operator<< (
            std::ostream & os,
            const Matrix4x4 & t )
```

Zeilenweise Ausgabe einer Matrix.

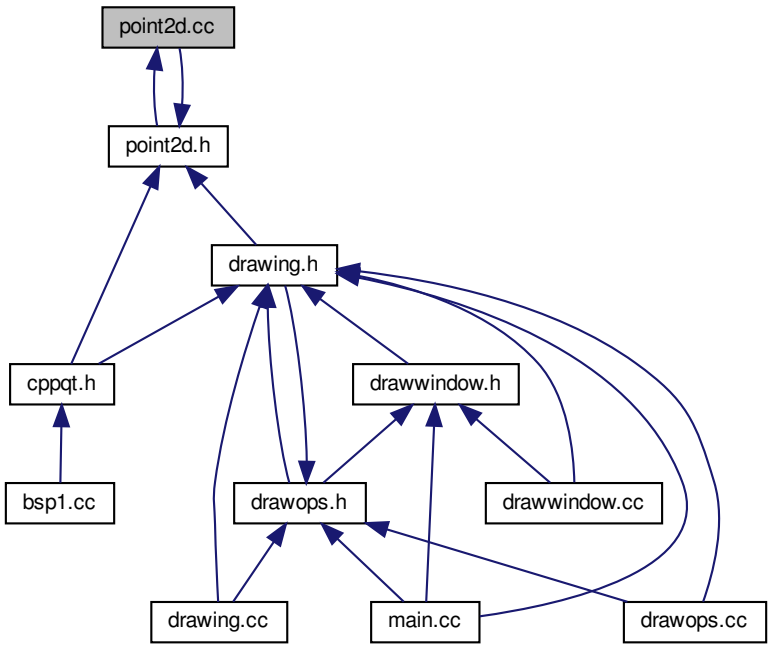
5.14 point2d.cc-Dateireferenz

```
#include <cmath>
#include <iostream>
#include "point2d.h"
```

Include-Abhängigkeitsdiagramm für point2d.cc:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Makrodefinitionen

- #define POINT2D_CC

Funktionen

- template<class T >
  std::ostream & operator<< (std::ostream &os, const Point2D< T > &p)
  Ausgabe eines Punktes in der Form (2,3)
- template<class T >
  std::istream & operator>> (std::istream &is, Point2D< T > &p)
  Eingabe eines Punktes in der Form (2,3)
- template<class T >
  Point2D< int > round (const Point2D< T > &p)
  Rundung auf Integer-Punkte.
- template<class T >
  Point2D< T > operator+ (const Point2D< T > &p1, const Point2D< T > &p2)
  Vektor plus Vektor.
- template<class T >
  Point2D< T > operator- (const Point2D< T > &p1, const Point2D< T > &p2)
  Vektor minus Vektor.
- template<class T >
  Point2D< T > operator* (T a, const Point2D< T > &p)
  Skalar mal Vektor.
- template<class T >
  Point2D< T > operator* (const Point2D< T > &p, T a)
  Vektor mal Skalar.
- template<class T >
  Point2D< T > operator/ (const Point2D< T > &p, T a)
  Vektor durch Skalar.
- template<class T >
  double norm (const Point2D< T > &p)
  Vektornorm.
- template<>
  double norm (const Point2D< int > &p)
  Vektornorm, Variante für int-Punkte.

5.14.1 Ausführliche Beschreibung

Autor

Holger Arndt

Version

0.4

Datum

15.09.2015

5.14.2 Makro-Dokumentation

5.14.2.1 POINT2D_CC

```
#define POINT2D_CC
```

Definiert in Zeile 7 der Datei point2d.cc.

5.14.3 Dokumentation der Funktionen

5.14.3.1 norm() [1/2]

```
template<class T >
double norm (
            const Point2D< T > & p )
```

Vektornorm.

Definiert in Zeile 59 der Datei point2d.cc.

Benutzt Point2D< T >::x und Point2D< T >::y.

5.14.3.2 norm() [2/2]

```
template<>
double norm (
            const Point2D< int > & p )  [inline]
```

Vektornorm, Variante für int-Punkte.

Definiert in Zeile 63 der Datei point2d.cc.

Benutzt Point2D< T >::x und Point2D< T >::y.

5.14.3.3 operator∗() [1/2]

```
template<class T >
Point2D<T> operator* (
            T a,
            const Point2D< T > & p )
```

Skalar mal Vektor.

Definiert in Zeile 47 der Datei point2d.cc.

Benutzt Point2D< T >::x und Point2D< T >::y.

5.14.3.4 operator∗() [2/2]

```
template<class T >
Point2D<T> operator* (
            const Point2D< T > & p,
            T a )
```

Vektor mal Skalar.

Definiert in Zeile 51 der Datei point2d.cc.

5.14.3.5 operator+()

```
template<class T >
Point2D<T> operator+ (
            const Point2D< T > & p1,
            const Point2D< T > & p2 )
```

Vektor plus Vektor.

Definiert in Zeile 39 der Datei point2d.cc.

Benutzt Point2D< T >::x und Point2D< T >::y.

5.14.3.6 operator-()

```
template<class T >
Point2D<T> operator- (
            const Point2D< T > & p1,
            const Point2D< T > & p2 )
```

Vektor minus Vektor.

Definiert in Zeile 43 der Datei point2d.cc.

Benutzt Point2D< T >::x und Point2D< T >::y.

5.14.3.7 operator/()

```
template<class T >
Point2D<T> operator/ (
            const Point2D< T > & p,
            T a )
```

Vektor durch Skalar.

Definiert in Zeile 55 der Datei point2d.cc.

Benutzt Point2D< T >::x und Point2D< T >::y.

5.14.3.8 operator<<()

```
template<class T >
std::ostream& operator<< (
            std::ostream & os,
            const Point2D< T > & p )
```

Ausgabe eines Punktes in der Form (2,3)

Definiert in Zeile 14 der Datei point2d.cc.

5.14.3.9 operator>>()

```
template<class T >
std::istream& operator>> (
            std::istream & is,
            Point2D< T > & p )
```

Eingabe eines Punktes in der Form (2,3)

Definiert in Zeile 21 der Datei point2d.cc.

Benutzt Point2D< T >::x und Point2D< T >::y.

5.14.3.10 round()

```
template<class T >
Point2D<int> round (
            const Point2D< T > & p )
```

Rundung auf Integer-Punkte.

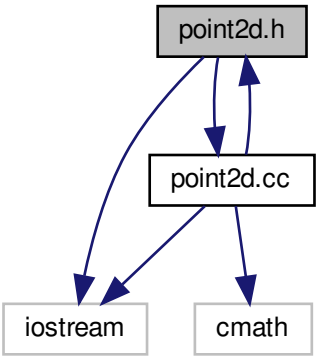
Definiert in Zeile 32 der Datei point2d.cc.

Benutzt Point2D< T >::x und Point2D< T >::y.

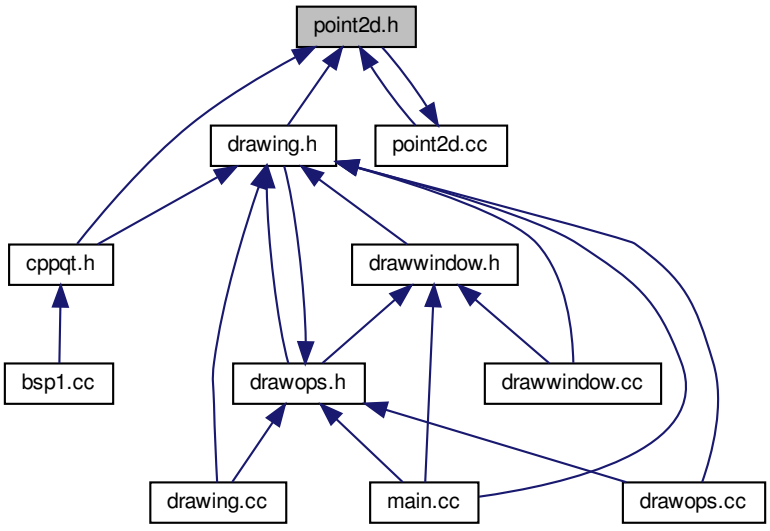
5.15 point2d.h-Dateireferenz

```
#include <iostream>
#include "point2d.cc"
```

Include-Abhängigkeitsdiagramm für point2d.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class Point2D< T >

  Punkt in der Ebene.

Typdefinitionen

- typedef Point2D< int > IPoint2D

  Ein Punkt mit ganzzahligen Koordinaten.
- typedef Point2D< double > DPoint2D

  Ein Punkt mit double-wertigen Koordinaten.

Funktionen

- template<class T >
  std::ostream & operator<< (std::ostream &os, const Point2D< T > &p)

  Ausgabe eines Punktes in der Form (2,3)
- template<class T >
  std::istream & operator>> (std::istream &is, Point2D< T > &p)

  Eingabe eines Punktes in der Form (2,3)
- template<class T >
  Point2D< int > round (const Point2D< T > &p)

  Rundung auf Integer-Punkte.
- template<class T >
  Point2D< T > operator+ (const Point2D< T > &p1, const Point2D< T > &p2)

  Vektor plus Vektor.
- template<class T >
  Point2D< T > operator- (const Point2D< T > &p1, const Point2D< T > &p2)

  Vektor minus Vektor.
- template<class T >
  Point2D< T > operator* (T a, const Point2D< T > &p)

  Skalar mal Vektor.
- template<class T >
  Point2D< T > operator* (const Point2D< T > &p, T a)

  Vektor mal Skalar.
- template<class T >
  Point2D< T > operator/ (const Point2D< T > &p, T a)

  Vektor durch Skalar.
- template<class T >
  double norm (const Point2D< T > &p)

  Vektornorm.
- template<>
  double norm (const Point2D< int > &p)

  Vektornorm, Variante für int-Punkte.

5.15.1 Ausführliche Beschreibung

Autor

Holger Arndt

Version

0.4

Datum

15.09.2015

5.15.2 Dokumentation der benutzerdefinierten Typen

5.15.2.1 DPoint2D

```
typedef Point2D<double> DPoint2D
```

Ein Punkt mit double-wertigen Koordinaten.

Definiert in Zeile 74 der Datei point2d.h.

5.15.2.2 IPoint2D

```
typedef Point2D<int> IPoint2D
```

Ein Punkt mit ganzzahligen Koordinaten.

Definiert in Zeile 71 der Datei point2d.h.

5.15.3 Dokumentation der Funktionen

5.15.3.1 norm() [1/2]

```
template<class T >
double norm (
            const Point2D< T > & p )
```

Vektornorm.

Definiert in Zeile 59 der Datei point2d.cc.

Benutzt Point2D< T >::x und Point2D< T >::y.

5.15.3.2 norm() [2/2]

```
template<>
double norm (
            const Point2D< int > & p )  [inline]
```

Vektornorm, Variante für int-Punkte.

Definiert in Zeile 63 der Datei point2d.cc.

Benutzt Point2D< T >::x und Point2D< T >::y.

5.15.3.3 operator∗() [1/2]

```
template<class T >
Point2D<T> operator* (
            T a,
            const Point2D< T > & p )
```

Skalar mal Vektor.

Definiert in Zeile 47 der Datei point2d.cc.

Benutzt Point2D< T >::x und Point2D< T >::y.

5.15.3.4 operator∗() [2/2]

```
template<class T >
Point2D<T> operator* (
            const Point2D< T > & p,
            T a )
```

Vektor mal Skalar.

Definiert in Zeile 51 der Datei point2d.cc.

5.15.3.5 operator+()

```
template<class T >
Point2D<T> operator+ (
            const Point2D< T > & p1,
            const Point2D< T > & p2 )
```

Vektor plus Vektor.

Definiert in Zeile 39 der Datei point2d.cc.

Benutzt Point2D< T >::x und Point2D< T >::y.

5.15.3.6 operator-()

```
template<class T >
Point2D<T> operator- (
            const Point2D< T > & p1,
            const Point2D< T > & p2 )
```

Vektor minus Vektor.

Definiert in Zeile 43 der Datei point2d.cc.

Benutzt Point2D< T >::x und Point2D< T >::y.

5.15.3.7 operator/()

```
template<class T >
Point2D<T> operator/ (
            const Point2D< T > & p,
            T a )
```

Vektor durch Skalar.

Definiert in Zeile 55 der Datei point2d.cc.

Benutzt Point2D< T >::x und Point2D< T >::y.

5.15.3.8 operator<<()

```
template<class T >
std::ostream& operator<< (
            std::ostream & os,
            const Point2D< T > & p )
```

Ausgabe eines Punktes in der Form (2,3)

Definiert in Zeile 14 der Datei point2d.cc.

5.15.3.9 operator>>()

```
template<class T >
std::istream& operator>> (
            std::istream & is,
            Point2D< T > & p )
```

Eingabe eines Punktes in der Form (2,3)

Definiert in Zeile 21 der Datei point2d.cc.

Benutzt Point2D< T >::x und Point2D< T >::y.

5.15.3.10 round()

```
template<class T >
Point2D<int> round (
            const Point2D< T > & p )
```

Rundung auf Integer-Punkte.

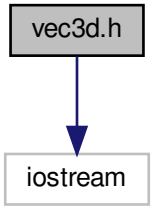
Definiert in Zeile 32 der Datei point2d.cc.

Benutzt Point2D< T >::x und Point2D< T >::y.

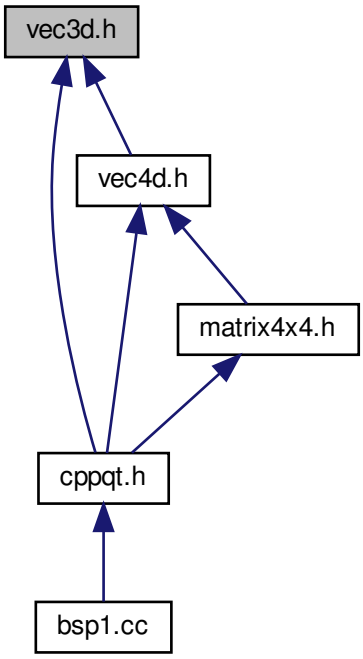
5.16 vec3d.h-Dateireferenz

```
#include <iostream>
```

Include-Abhängigkeitsdiagramm für vec3d.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class Vec3D

  Koordinaten eines 3D-Vektors.

Funktionen

- Vec3D operator∗ (double a, const Vec3D &v)

  Skalar mal Vektor.
- Vec3D operator∗ (const Vec3D &v1, const Vec3D &v2)

  Elementweise Multiplikation Vektor mal Vektor.
- Vec3D operator/ (const Vec3D &v, double a)

  Elementweise Division Vektor durch Skalar.
- Vec3D operator+ (const Vec3D &v1, const Vec3D &v2)

  Vektor plus Vektor.
- Vec3D operator- (const Vec3D &v1, const Vec3D &v2)

  Vektor minus Vektor.
- std::ostream & operator<< (std::ostream &os, const Vec3D &v)

  Ausgabe eines Vektors in der Form (x y z)
- double norm (const Vec3D &v)

  Vektornorm.
- double skalarprod (const Vec3D &v1, const Vec3D &v2)

  Skalarprodukt.
- Vec3D kreuzprod (const Vec3D &v1, const Vec3D &v2)

  Kreuzprodukt.

5.16.1 Ausführliche Beschreibung

Autor

Holger Arndt, Sebastian Birk, Martin Galgon

Version

0.1

Datum

10.11.2016

5.16.2 Dokumentation der Funktionen

5.16.2.1 kreuzprod()

```
Vec3D kreuzprod (
            const Vec3D & v1,
            const Vec3D & v2 )
```

Kreuzprodukt.

5.16.2.2 norm()

```
double norm (
            const Vec3D & v )
```

Vektornorm.

5.16.2.3 operator∗() [1/2]

```
Vec3D operator* (
            double a,
            const Vec3D & v )
```

Skalar mal Vektor.

5.16.2.4 operator∗() [2/2]

```
Vec3D operator* (
            const Vec3D & v1,
            const Vec3D & v2 )
```

Elementweise Multiplikation Vektor mal Vektor.

5.16.2.5 operator+()

```
Vec3D operator+ (
            const Vec3D & v1,
            const Vec3D & v2 )
```

Vektor plus Vektor.

5.16.2.6 operator-()

```
Vec3D operator- (
            const Vec3D & v1,
            const Vec3D & v2 )
```

Vektor minus Vektor.

5.16.2.7 operator/()

```
Vec3D operator/ (
            const Vec3D & v,
            double a )
```

Elementweise Division Vektor durch Skalar.

5.16.2.8 operator<<()

```
std::ostream& operator<< (
            std::ostream & os,
            const Vec3D & v )
```

Ausgabe eines Vektors in der Form (x y z)

5.16.2.9 skalarprod()

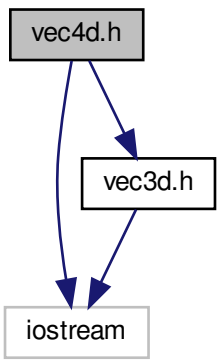
```
double skalarprod (
            const Vec3D & v1,
            const Vec3D & v2 )
```

Skalarprodukt.

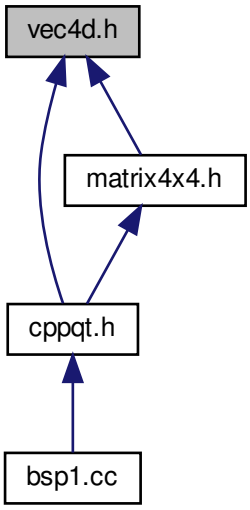
5.17 vec4d.h-Dateireferenz

```
#include <iostream>
#include <vec3d.h>
```

Include-Abhängigkeitsdiagramm für vec4d.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class Vec4D

  Homogene Koordinaten eines 3D-Vektors oder ein 4D-Vektor.

Funktionen

- std::ostream & operator<< (std::ostream &os, const Vec4D &v)

  Ausgabe eines Vektors in der Form (x y z w)
- double skalarprod (const Vec4D &v1, const Vec4D &v2)

  Skalarprodukt.

5.17.1 Ausführliche Beschreibung

Autor

Holger Arndt, Sebastian Birk, Martin Galgon

Version

0.1

Datum

10.11.2016

5.17.2 Dokumentation der Funktionen

5.17.2.1 operator<<()

```
std::ostream& operator<< (
            std::ostream & os,
            const Vec4D & v )
```

Ausgabe eines Vektors in der Form (x y z w)

5.17.2.2 skalarprod()

```
double skalarprod (
            const Vec4D & v1,
            const Vec4D & v2 )
```

Skalarprodukt.

Index