CF_CG-Lib

1.0

Generated by Doxygen 1.8.12

Contents

Chapter 1

CF_CG-Lib

This library is inteted to be used in 'Chaos und Fraktale' and 'Computer Geometry', lessons from 'Hochschule Darmstadt'. If you want to use it in differnt way, you may do so under the terms of the MIT license.

The best way to find ALL functions is by going to 'namespaces cf' (Note: register 'classes' doesn't show 'namespace global' functions)

The MIT License (MIT)

3rd party licenses:

- GLM: MIT
- OpenCV: 3-clause BSD License
- InfInt: LGPL
- FreeGlut: X-Consortium

Note: OpenCV utilizes 3rd party libraries like libpng, these licenes are NOT covered in this segment

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 cf Namespace Reference

Classes

- struct Color

  The Color struct offers a class for rgb access.
- struct Console

  The Console struct offers utility functions for 'console'.
- struct Direction

  The Direction struct for getting absolute directions from a current direction and a relative direction.
- struct Interval

  The Interval struct provides functionallity to translate values from one interval into another.
- struct IteratedFunctionSystem

  The IteratedFunctionSystem class lazy people (like myself) may use the IFS tyepdef.
- struct LindenmayerSystem

  The LindenmayerSystem class lazy people (like myself) may use the IFS tyepdef.
- struct Orbit

  The Orbit class lazy people (like myself) may use the ORB tyepdef.
- struct Point

  The Point struct is a simple class for positon access on 2D images (imilar to cv::Point, but uses floats instead of integer)
- struct Vec3

  The Vec3 struct General class for vector operations.
- class Window2D

  The Window2D struct offers advanced features used by WindowRasterized/WindowVertorized.
- struct Window3D

  The Window3D struct is the default class for accessing 3D content, creating more than 1 instance results in undefined behavior.
- struct WindowCoordinateSystem

  The WindowCoordinateSystem struct Default class for images and raster operations.
- struct WindowRasterized

  The WindowRasterized struct Default struct for verctorized operations within a custom interval.
- struct WindowVectorized

  The WindowVectorized struct Default class for images and raster operations.

Typedefs

- typedef Vec3< true > PointVector

  PointVector Specialiaztion of general Vec3.
- typedef Vec3< false > DirectionVector

  DirectionVector Specialiaztion of general Vec3, where component 'w' may not be written to.
- typedef IteratedFunctionSystem IFS
- typedef LindenmayerSystem LSystem
- typedef Orbit ORB

Functions

- void _removeWindowsSpecificCarriageReturn (std::string &str)

  _removeWindowsSpecificCarriageReturn Removes 'carriage return' characters in strings ('carriage return' may be read from unix system by providing windows files)
- std::vector< Color > readPaletteFromFile (const std::string &filePath)

  readPaletteFromFile
- std::string readAntString (const std::string &filePath)

  readAntString
- template<typename _VectorType = glm::vec3>
  std::vector< _VectorType > readDATFile (const std::string &filePath)

  readDATFile Reads a $*$.dat file
- float radian2degree (float radianValue)

  radian2degree Converts a radian value to a degree value
- float degree2radian (float degreeValue)

  degree2radian Converts a degree value to a radian value

6.1.1 Typedef Documentation

6.1.1.1 DirectionVector

```
typedef Vec3<false> cf::DirectionVector
```

DirectionVector Specialiaztion of general Vec3, where component 'w' may not be written to.

6.1.1.2 IFS

```
typedef IteratedFunctionSystem cf::IFS
```

6.1.1.3 LSystem

```
typedef LindenmayerSystem cf::LSystem
```

6.1.1.4 ORB

```
typedef Orbit cf::ORB
```

6.1.1.5 PointVector

```
typedef Vec3<true > cf::PointVector
```

PointVector Specialiaztion of general Vec3.

6.1.2 Function Documentation

6.1.2.1 _removeWindowsSpecificCarriageReturn()

```
void cf::_removeWindowsSpecificCarriageReturn (
             std::string & str )
```

_removeWindowsSpecificCarriageReturn Removes 'carriage return' characters in strings ('carriage return' may be read from unix system by providing windows files)

Parameters

str	string containing 'carriage return', which will be removed

6.1.2.2 degree2radian()

```
float cf::degree2radian (
             float degreeValue )
```

degree2radian Converts a degree value to a radian value

Parameters

degreeValue	Degree value to be converted

Returns

Converted radian value

6.1.2.3 radian2degree()

```
float cf::radian2degree (
             float radianValue )
```

radian2degree Converts a radian value to a degree value

Parameters

radianValue	Radian value to be converted

Returns

Converted degree value

6.1.2.4 readAntString()

```
std::string cf::readAntString (
             const std::string & filePath )
```

readAntString

Parameters

filePath	Read ∗.ant file from path

Returns

6.1.2.5 readDATFile()

```
template<typename _VectorType = glm::vec3>
std::vector<_VectorType> cf::readDATFile (
             const std::string & filePath )
```

readDATFile Reads a ∗.dat file

Parameters

filePath	Read ∗.dat file from path

Returns

6.1.2.6 readPaletteFromFile()

```
std::vector<Color> cf::readPaletteFromFile (
             const std::string & filePath )
```

readPaletteFromFile

Parameters

filePath	Read ∗.pal file from path

Returns

Chapter 7

Class Documentation

7.1 cf::Color Struct Reference

The Color struct offers a class for rgb access.

```
#include <utils.h>
```

Public Member Functions

- Color (uint8_t red=0, uint8_t green=0, uint8_t blue=0)
- Color operator* (float value)
- Color operator/ (float value)
- Color & operator*= (float value)
- Color & operator/= (float value)
- Color operator+ (const Color &c)
- Color operator- (const Color &c)
- Color & operator+= (const Color &c)
- Color & operator-= (const Color &c)
- bool operator== (const Color &c)
- bool operator!= (const Color &c)
- Color invert () const

Public Attributes

- uint8_t b
- uint8_t g
- uint8_t r

Static Public Attributes

- static const Color MAGENTA
- static const Color YELLOW
- static const Color ORANGE
- static const Color WHITE
- static const Color BLACK
- static const Color GREEN
- static const Color GREY
- static const Color BLUE
- static const Color CYAN
- static const Color PINK
- static const Color RED

Friends

- cf::Color operator∗ (float value, const cf::Color &c)
- cf::Color operator/ (float value, const cf::Color &c)
- std::ostream & operator<< (std::ostream &os, const Color &c)

7.1.1 Detailed Description

The Color struct offers a class for rgb access.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 Color()

```
cf::Color::Color (
             uint8_t red = 0,
             uint8_t green = 0,
             uint8_t blue = 0 )  [inline]
```

7.1.3 Member Function Documentation

7.1.3.1 invert()

```
Color cf::Color::invert ( ) const
```

7.1.3.2 operator"!=()

```
bool cf::Color::operator!= (
             const Color & c )
```

7.1.3.3 operator∗()

```
Color cf::Color::operator* (
             float value )
```

7.1.3.4 operator∗=()

```
Color& cf::Color::operator*= (
             float value )
```

7.1.3.5 operator+()

```
Color cf::Color::operator+ (
             const Color & c )
```

7.1.3.6 operator+=()

```
Color& cf::Color::operator+= (
             const Color & c )
```

7.1.3.7 operator-()

```
Color cf::Color::operator- (
             const Color & c )
```

7.1.3.8 operator-=()

```
Color& cf::Color::operator-= (
             const Color & c )
```

7.1.3.9 operator/()

```
Color cf::Color::operator/ (
             float value )
```

7.1.3.10 operator/=()

```
Color& cf::Color::operator/= (
             float value )
```

7.1.3.11 operator==()

```
bool cf::Color::operator== (
             const Color & c )
```

7.1.4 Friends And Related Function Documentation

7.1.4.1 operator*

```
cf::Color operator* (
             float value,
             const cf::Color & c )  [friend]
```

7.1.4.2 operator/

```
cf::Color operator/ (
             float value,
             const cf::Color & c )  [friend]
```

7.1.4.3 operator<<

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

7.1.5 Member Data Documentation

7.1.5.1 b

```
uint8_t cf::Color::b
```

7.1.5.2 BLACK

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

7.1.5.3 BLUE

```
const Color cf::Color::BLUE  [static]
```

7.1.5.4 CYAN

```
const Color cf::Color::CYAN  [static]
```

7.1.5.5 g

```
uint8_t cf::Color::g
```

7.1.5.6 GREEN

```
const Color cf::Color::GREEN  [static]
```

7.1.5.7 GREY

```
const Color cf::Color::GREY  [static]
```

7.1.5.8 MAGENTA

```
const Color cf::Color::MAGENTA  [static]
```

7.1.5.9 ORANGE

```
const Color cf::Color::ORANGE [static]
```

7.1.5.10 PINK

```
const Color cf::Color::PINK [static]
```

7.1.5.11 r

```
uint8_t cf::Color::r
```

7.1.5.12 RED

```
const Color cf::Color::RED [static]
```

7.1.5.13 WHITE

```
const Color cf::Color::WHITE [static]
```

7.1.5.14 YELLOW

```
const Color cf::Color::YELLOW [static]
```

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

- include/utils.h

7.2 cf::Console Struct Reference

The Console struct offers utility functions for 'console'.

```
#include <utils.h>
```

Static Public Member Functions

- static std::string readString ()

  readString Read a line into a std::string (includes spaces)
- static float readFloat ()

  readFloat Reads a floatingpoint value
- static int readInt ()

  readInt Reads a integer value
- static void waitKey ()

  waitKey Wait until key input (on windows also sets the console window active)
- static void clearConsole ()

  clearConsole Clears the console

7.2.1 Detailed Description

The Console struct offers utility functions for 'console'.

7.2.2 Member Function Documentation

7.2.2.1 clearConsole()

```
static void cf::Console::clearConsole ( )  [static]
```

clearConsole Clears the console

7.2.2.2 readFloat()

```
static float cf::Console::readFloat ( )  [static]
```

readFloat Reads a floatingpoint value

Returns

Read value

7.2.2.3 readInt()

```
static int cf::Console::readInt ( )  [static]
```

readInt Reads a integer value

Returns

Read value

7.2.2.4 readString()

```
static std::string cf::Console::readString ( )  [static]
```

readString Read a line into a std::string (includes spaces)

Returns

Read line

7.2.2.5 waitKey()

```
static void cf::Console::waitKey ( )  [static]
```

waitKey Wait until key input (on windows also sets the console window active)

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

- include/utils.h

7.3 cf::Direction Struct Reference

The Direction struct for getting absolute directions from a current direction and a relative direction.

```
#include <utils.h>
```

Public Types

- enum AbsoluteDirection {
  AbsoluteDirection::NORTH, AbsoluteDirection::EAST, AbsoluteDirection::SOUTH, AbsoluteDirection::WEST,
  AbsoluteDirection::NUM_ABS_DIRS }
- enum RelativeDirection { RelativeDirection::LEFT, RelativeDirection::FORWARD, RelativeDirection::RIGHT, RelativeDirection::NUM_REL_DIRS }

Static Public Member Functions

- static AbsoluteDirection getNextiDirection (AbsoluteDirection currentDirection, RelativeDirection relativeMovement)

  getNextiDirection receive absolute direction by providing a relative directon
- static std::string toString (AbsoluteDirection absDir)
- static std::string toString (RelativeDirection relDir)

7.3.1 Detailed Description

The Direction struct for getting absolute directions from a current direction and a relative direction.

7.3.2 Member Enumeration Documentation

7.3.2.1 AbsoluteDirection

```
enum cf::Direction::AbsoluteDirection  [strong]
```

Enumerator

NORTH	
EAST	
SOUTH	
WEST	
NUM_ABS_DIRS	

7.3.2.2 RelativeDirection

```
enum cf::Direction::RelativeDirection  [strong]
```

Enumerator

LEFT	
FORWARD	
RIGHT	
NUM_REL_DIRS	

7.3.3 Member Function Documentation

7.3.3.1 getNextiDirection()

```
static AbsoluteDirection cf::Direction::getNextiDirection (
             AbsoluteDirection currentDirection,
             RelativeDirection relativeMovement )  [static]
```

getNextiDirection receive absolute direction by providing a relative directon

Parameters

currentDirection	current absolute direction
relativeMovement	relative movement

Returns

7.3.3.2 toString() [1/2]

```
static std::string cf::Direction::toString (
             AbsoluteDirection absDir )  [static]
```

7.3.3.3 toString() [2/2]

```
static std::string cf::Direction::toString (
             RelativeDirection relDir )  [static]
```

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

- include/utils.h

7.4 cf::Interval Struct Reference

The Interval struct provides functionallity to translate values from one interval into another.

```
#include <utils.h>
```

Public Member Functions

- Interval (float _min=0, float _max=0)

Static Public Member Functions

- static float translateIntervalPostion (const Interval &originalInterval, const Interval &newInterval, float originalPosition)

Public Attributes

- float min
- float max

Friends

- std::ostream & operator<< (std::ostream &os, const Interval &interval)

7.4.1 Detailed Description

The Interval struct provides functionallity to translate values from one interval into another.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 Interval()

```
cf::Interval::Interval (
             float _min = 0,
             float _max = 0 )  [inline]
```

7.4.3 Member Function Documentation

7.4.3.1 translateIntervalPostion()

```
static float cf::Interval::translateIntervalPostion (
             const Interval & originalInterval,
             const Interval & newInterval,
             float originalPosition )  [static]
```

7.4.4 Friends And Related Function Documentation

7.4.4.1 operator<<

```
std::ostream& operator<< (
            std::ostream & os,
            const Interval & interval ) [friend]
```

7.4.5 Member Data Documentation

7.4.5.1 max

```
float cf::Interval::max
```

7.4.5.2 min

```
float cf::Interval::min
```

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

- include/utils.h

7.5 cf::IteratedFunctionSystem Struct Reference

The IteratedFunctionSystem class lazy people (like myself) may use the IFS tyepdef.

```
#include <IFS.h>
```

Public Member Functions

- void read (const std::string &fiilePath)

  read a ∗.ifs file from path
- std::size_t getNumTransformations () const
- const glm::mat3x3 & getTransformation (std::size_t pos) const
- const Interval & getRangeX () const
- const Interval & getRangeY () const
- const std::string & getName () const
- const std::vector< glm::mat3x3 > & getAllTransformation () const

7.5.1 Detailed Description

The IteratedFunctionSystem class lazy people (like myself) may use the IFS tyepdef.

7.5.2 Member Function Documentation

7.5.2.1 getAllTransformation()

```
const std::vector<glm::mat3x3>& cf::IteratedFunctionSystem::getAllTransformation ( ) const
```

7.5.2.2 getName()

```
const std::string& cf::IteratedFunctionSystem::getName ( ) const
```

7.5.2.3 getNumTransformations()

```
std::size_t cf::IteratedFunctionSystem::getNumTransformations ( ) const
```

7.5.2.4 getRangeX()

```
const Interval& cf::IteratedFunctionSystem::getRangeX ( ) const
```

7.5.2.5 getRangeY()

```
const Interval& cf::IteratedFunctionSystem::getRangeY ( ) const
```

7.5.2.6 getTransformation()

```
const glm::mat3x3& cf::IteratedFunctionSystem::getTransformation (
             std::size_t pos ) const
```

7.5.2.7 read()

```
void cf::IteratedFunctionSystem::read (
             const std::string & fiilePath )
```

read a ∗.ifs file from path

Parameters

fiilePath	Path to a ∗.ifs file

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

- include/IFS.h

7.6 cf::LindenmayerSystem Struct Reference

The LindenmayerSystem class lazy people (like myself) may use the IFS tyepdef.

```
#include <LSystem.h>
```

Public Member Functions

- void read (const std::string &filePath)

  read a ∗.lin file from path
- const std::string & getName () const
- char getAxiom () const
- const std::string ∗ getProduction (char symbol) const
- std::size_t getNumProductions () const
- bool clearWindowEachTime () const
- const Interval & getRangeX () const
- const Interval & getRangeY () const
- float getScale () const
- float getStartAngle () const
- float getAdjustmentAngel () const
- const std::vector< std::pair< const char, const std::string > > & getAllProductions () const

7.6.1 Detailed Description

The LindenmayerSystem class lazy people (like myself) may use the IFS tyepdef.

7.6.2 Member Function Documentation

7.6.2.1 clearWindowEachTime()

```
bool cf::LindenmayerSystem::clearWindowEachTime ( ) const
```

7.6.2.2 getAdjustmentAngel()

```
float cf::LindenmayerSystem::getAdjustmentAngel ( ) const
```

7.6.2.3 getAllProductions()

```
const std::vector<std::pair<const char, const std::string> >& cf::LindenmayerSystem::getAll↩
Productions ( ) const
```

7.6.2.4 getAxiom()

```
char cf::LindenmayerSystem::getAxiom ( ) const
```

7.6.2.5 getName()

```
const std::string& cf::LindenmayerSystem::getName ( ) const
```

7.6.2.6 getNumProductions()

```
std::size_t cf::LindenmayerSystem::getNumProductions ( ) const
```

7.6.2.7 getProduction()

```
const std::string* cf::LindenmayerSystem::getProduction (
             char symbol ) const
```

7.6.2.8 getRangeX()

```
const Interval& cf::LindenmayerSystem::getRangeX ( ) const
```

7.6.2.9 getRangeY()

```
const Interval& cf::LindenmayerSystem::getRangeY ( ) const
```

7.6.2.10 getScale()

```
float cf::LindenmayerSystem::getScale ( ) const
```

7.6.2.11 getStartAngle()

```
float cf::LindenmayerSystem::getStartAngle ( ) const
```

7.6.2.12 read()

```
void cf::LindenmayerSystem::read (
             const std::string & filePath )
```

read a ∗.lin file from path

Parameters

filePath	Path to a ∗.lin file

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

- include/LSystem.h

7.7 cf::Orbit Struct Reference

The Orbit class lazy people (like myself) may use the ORB tyepdef.

```
#include <ORB.h>
```

Public Member Functions

- void read (const std::string &filePath)

  read a ∗.orb file from path
- const Interval & getRangeX () const
- const Interval & getRangeY () const
- const std::string & getName () const
- const std::vector< glm::vec3 > & getAllStartingPoints () const
- const std::vector< float > & getAllFactors () const
- std::size_t getNumFactors () const
- std::size_t getNumStartingPoints () const

7.7.1 Detailed Description

The Orbit class lazy people (like myself) may use the ORB tyepdef.

7.7.2 Member Function Documentation

7.7.2.1 getAllFactors()

```
const std::vector<float>& cf::Orbit::getAllFactors ( ) const
```

7.7.2.2 getAllStartingPoints()

```
const std::vector<glm::vec3>& cf::Orbit::getAllStartingPoints ( ) const
```

7.7.2.3 getName()

```
const std::string& cf::Orbit::getName ( ) const
```

7.7.2.4 getNumFactors()

```
std::size_t cf::Orbit::getNumFactors ( ) const
```

7.7.2.5 getNumStartingPoints()

```
std::size_t cf::Orbit::getNumStartingPoints ( ) const
```

7.7.2.6 getRangeX()

```
const Interval& cf::Orbit::getRangeX ( ) const
```

7.7.2.7 getRangeY()

```
const Interval& cf::Orbit::getRangeY ( ) const
```

7.7.2.8 read()

```
void cf::Orbit::read (
             const std::string & filePath )
```

read a ∗.orb file from path

Parameters

filePath	Path to a ∗.orb file

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

- include/ORB.h

7.8 cf::Point Struct Reference

The Point struct is a simple class for positon access on 2D images (imilar to cv::Point, but uses floats instead of integer)

```
#include <window2D.h>
```

Public Member Functions

- Point (float val_x=0.f, float val_y=0.f)
- bool operator== (const Point &p) const
- bool operator!= (const Point &p) const
- Point operator+ (const Point &p) const
- Point & operator+= (const Point &p)
- Point operator- (const Point &p) const
- Point & operator-= (const Point &p)
- Point operator∗ (float factor) const
- Point & operator∗= (float factor)
- Point operator/ (float rhs) const
- Point & operator/= (float rhs)
- operator cv::Point () const

Public Attributes

- float x
- float y

Friends

- Point operator∗ (float lhs, const Point &p)
- Point operator/ (float lhs, const Point &p)

7.8.1 Detailed Description

The Point struct is a simple class for positon access on 2D images (imilar to cv::Point, but uses floats instead of integer)

7.8.2 Constructor & Destructor Documentation

7.8.2.1 Point()

```
cf::Point::Point (
             float val_x = 0.f,
             float val_y = 0.f )  [inline]
```

7.8.3 Member Function Documentation

7.8.3.1 operator cv::Point()

```
cf::Point::operator cv::Point ( ) const
```

7.8.3.2 operator"!=()

```
bool cf::Point::operator!= (
             const Point & p ) const
```

7.8.3.3 operator∗()

```
Point cf::Point::operator* (
             float factor ) const
```

7.8.3.4 operator∗=()

```
Point& cf::Point::operator*= (
             float factor )
```

7.8.3.5 operator+()

```
Point cf::Point::operator+ (
             const Point & p ) const
```

7.8.3.6 operator+=()

```
Point& cf::Point::operator+= (
             const Point & p )
```

7.8.3.7 operator-()

```
Point cf::Point::operator- (
             const Point & p ) const
```

7.8.3.8 operator-=()

```
Point& cf::Point::operator-= (
             const Point & p )
```

7.8.3.9 operator/()

```
Point cf::Point::operator/ (
             float rhs ) const
```

7.8.3.10 operator/=()

```
Point& cf::Point::operator/= (
             float rhs )
```

7.8.3.11 operator==()

```
bool cf::Point::operator== (
             const Point & p ) const
```

7.8.4 Friends And Related Function Documentation

7.8.4.1 operator∗

```
Point operator* (
             float lhs,
             const Point & p )  [friend]
```

7.8.4.2 operator/

```
Point operator/ (
            float lhs,
            const Point & p )  [friend]
```

7.8.5 Member Data Documentation

7.8.5.1 x

```
float cf::Point::x
```

7.8.5.2 y

```
float cf::Point::y
```

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

- include/window2D.h

7.9 cf::Vec3< POINTVECTOR > Struct Template Reference

The Vec3 struct General class for vector operations.

```
#include <computerGeometry.hpp>
```

Public Member Functions

- Vec3 (float x=0.f, float y=0.f)
- Vec3 (float x, float y, float w)
- Vec3 (const cf::Point &p)
- template< bool RHS >
  Vec3< RHS|POINTVECTOR > operator+ (const Vec3< RHS > &rhs) const
- template< bool RHS >
  Vec3< POINTVECTOR > & operator+= (const Vec3< RHS > &rhs)
- template< bool RHS >
  Vec3< RHS|POINTVECTOR > operator- (const Vec3< RHS > &rhs) const
- template< bool RHS >
  Vec3< POINTVECTOR > & operator-= (const Vec3< RHS > &rhs)
- cf::Vec3< POINTVECTOR > operator∗ (float rhs) const

  operator∗ Multiplys each component of the vector with a factor
- cf::Vec3< POINTVECTOR > & operator∗= (float rhs)
- template< bool RHS >
  Vec3< RHS|POINTVECTOR > operator% (const Vec3< RHS > &rhs) const

  operator% Performs the cross product between two vectors
- template< bool RHS >
  Vec3< POINTVECTOR > & operator%= (const Vec3< RHS > &rhs)
- void normalize ()

normalize Normalizes the PointVector (division by the 'w' component), compile error on DirectionVecotrs

- bool isPointVector () const

  isPointVector Checks wether a Vector is a PointVector or DirectionVector
- template<bool RHS>
  float operator∗ (const Vec3< RHS > &rhs) const

  operator∗ Performs the dot product between two vectors
- float getX () const

  getX Read access to component 'x'
- float getY () const

  getY Read access to component 'y'
- float getW () const

  getW Read access to component 'w'
- void setX (float value)

  setX Write to component 'x'
- void setY (float value)

  setY Write to component 'y'
- void setW (float value)

  setW Write to component 'w', compile error on DirectionVectors
- float operator[ ] (int idx) const

  operator[] Access to each component of the Vector, Note: read access is granted to all components (including index 2)
- float & operator[ ] (int idx)

  operator[] Access to each component of the Vector, Note: no write access for index 2 on DirectionVectors
- operator glm::vec3 () const
- operator const glm::vec3 & () const
- operator cf::Point () const

  operator cf::Point Conversion operator to cf::Point, compile error on DirectionVectors
- cf::PointVector & operator= (const cf::Point &p)
- cf::Vec3< POINTVECTOR > & operator= (const glm::vec3 &rhs)
- operator cf::Vec3< false > () const

  operator cf::DirectionVector Conversion operator from PointVector to DirectionVector, exception if 'w' is not 0

Friends

- struct Vec3<!POINTVECTOR >
- cf::Vec3< POINTVECTOR > operator∗ (float lhs, const cf::Vec3< POINTVECTOR > &vec)
- template<bool b>
  std::ostream & operator<<) (std::ostream &, const Vec3< b > &)

7.9.1 Detailed Description

template<bool POINTVECTOR>
struct cf::Vec3< POINTVECTOR >

The Vec3 struct General class for vector operations.

it porvieds:

- conversions from/to cf::Point and glm::vec3
- Cross product ('operator') and dot product ('operator∗') with other vectors
- Support for DirectionVectors and PointVectors (see typedef 'PointVector' and 'DirectionVector')

7.9.2 Constructor & Destructor Documentation

7.9.2.1 Vec3() [1/3]

```
template<bool POINTVECTOR>
cf::Vec3< POINTVECTOR >::Vec3 (
             float x = 0.f,
             float y = 0.f )  [inline]
```

7.9.2.2 Vec3() [2/3]

```
template<bool POINTVECTOR>
cf::Vec3< POINTVECTOR >::Vec3 (
             float x,
             float y,
             float w )  [inline]
```

7.9.2.3 Vec3() [3/3]

```
template<bool POINTVECTOR>
cf::Vec3< POINTVECTOR >::Vec3 (
             const cf::Point & p )  [inline]
```

7.9.3 Member Function Documentation

7.9.3.1 getW()

```
template<bool POINTVECTOR>
float cf::Vec3< POINTVECTOR >::getW ( ) const  [inline]
```

getW Read access to component 'w'

Returns

7.9.3.2 getX()

```
template<bool POINTVECTOR>
float cf::Vec3< POINTVECTOR >::getX ( ) const  [inline]
```

getX Read access to component 'x'

Returns

7.9.3.3 getY()

```
template<bool POINTVECTOR>
float cf::Vec3< POINTVECTOR >::getY ( ) const  [inline]
```

getY Read access to component 'y'

Returns

7.9.3.4 isPointVector()

```
template<bool POINTVECTOR>
bool cf::Vec3< POINTVECTOR >::isPointVector ( ) const  [inline]
```

isPointVector Checks wether a Vector is a PointVector or DirectionVector

Returns

7.9.3.5 normalize()

```
template<bool POINTVECTOR>
void cf::Vec3< POINTVECTOR >::normalize ( )  [inline]
```

normalize Normalizes the PointVector (division by the 'w' component), compile error on DirectionVecotrs

7.9.3.6 operator cf::Point()

```
template<bool POINTVECTOR>
cf::Vec3< POINTVECTOR >::operator cf::Point ( ) const  [inline]
```

operator cf::Point Conversion operator to cf::Point, compile error on DirectionVectors

7.9.3.7 operator cf::Vec3< false >()

```
template<bool POINTVECTOR>
cf::Vec3< POINTVECTOR >::operator cf::Vec3< false > ( ) const  [inline]
```

operator cf::DirectionVector Conversion operator from PointVector to DirectionVector, exception if 'w' is not 0

7.9.3.8 operator const glm::vec3 &()

```
template<bool POINTVECTOR>
cf::Vec3< POINTVECTOR >::operator const glm::vec3 & ( ) const  [inline]
```

7.9.3.9 operator glm::vec3()

```
template<bool POINTVECTOR>
cf::Vec3< POINTVECTOR >::operator glm::vec3 ( ) const  [inline]
```

7.9.3.10 operator%()

```
template<bool POINTVECTOR>
template<bool RHS>
Vec3<RHS | POINTVECTOR> cf::Vec3< POINTVECTOR >::operator% (
             const Vec3< RHS > & rhs ) const  [inline]
```

operator% Performs the cross product between two vectors

Parameters

rhs	Second operand for cross product

Returns

7.9.3.11 operator%=()

```
template<bool POINTVECTOR>
template<bool RHS>
Vec3<POINTVECTOR>& cf::Vec3< POINTVECTOR >::operator%= (
             const Vec3< RHS > & rhs )  [inline]
```

7.9.3.12 operator*() [1/2]

```
template<bool POINTVECTOR>
cf::Vec3<POINTVECTOR> cf::Vec3< POINTVECTOR >::operator* (
             float rhs ) const  [inline]
```

operator* Multiplys each component of the vector with a factor

Parameters

rhs	Factor for the multiplication

Returns

Multiplied vector

7.9.3.13 operator∗() [2/2]

```
template<bool POINTVECTOR>
template<bool RHS>
float cf::Vec3< POINTVECTOR >::operator* (
             const Vec3< RHS > & rhs ) const  [inline]
```

operator∗ Performs the dot product between two vectors

Parameters

rhs	Second operand for dot product

Returns

7.9.3.14 operator∗=()

```
template<bool POINTVECTOR>
cf::Vec3<POINTVECTOR>& cf::Vec3< POINTVECTOR >::operator*= (
             float rhs )  [inline]
```

7.9.3.15 operator+()

```
template<bool POINTVECTOR>
template<bool RHS>
Vec3<RHS | POINTVECTOR> cf::Vec3< POINTVECTOR >::operator+ (
             const Vec3< RHS > & rhs ) const  [inline]
```

7.9.3.16 operator+=()

```
template<bool POINTVECTOR>
template<bool RHS>
Vec3<POINTVECTOR>& cf::Vec3< POINTVECTOR >::operator+= (
             const Vec3< RHS > & rhs )  [inline]
```

7.9.3.17 operator-()

```
template<bool POINTVECTOR>
template<bool RHS>
Vec3<RHS | POINTVECTOR> cf::Vec3< POINTVECTOR >::operator- (
             const Vec3< RHS > & rhs ) const  [inline]
```

7.9.3.18 operator-=()

```
template<bool POINTVECTOR>
template<bool RHS>
Vec3<POINTVECTOR>& cf::Vec3< POINTVECTOR >::operator-= (
             const Vec3< RHS > & rhs )  [inline]
```

7.9.3.19 operator=() [1/2]

```
template<bool POINTVECTOR>
cf::PointVector& cf::Vec3< POINTVECTOR >::operator= (
            const cf::Point & p ) [inline]
```

7.9.3.20 operator=() [2/2]

```
template<bool POINTVECTOR>
cf::Vec3<POINTVECTOR>& cf::Vec3< POINTVECTOR >::operator= (
            const glm::vec3 & rhs ) [inline]
```

7.9.3.21 operator[]() [1/2]

```
template<bool POINTVECTOR>
float cf::Vec3< POINTVECTOR >::operator[] (
            int idx ) const [inline]
```

operator[] Access to each component of the Vector, Note: read access is granted to all components (including index 2)

Parameters

idx	Acess index

Returns

7.9.3.22 operator[]() [2/2]

```
template<bool POINTVECTOR>
float& cf::Vec3< POINTVECTOR >::operator[] (
            int idx ) [inline]
```

operator[] Access to each component of the Vector, Note: no write access for index 2 on DirectionVectors

Parameters

idx	Acess index, idx = 0 -> x, idx = 1 -> y, idx = 2 -> w

Returns

7.9.3.23 setW()

```
template<bool POINTVECTOR>
void cf::Vec3< POINTVECTOR >::setW (
            float value )  [inline]
```

setW Write to component 'w', compile error on DirectionVectors

Parameters

value	

7.9.3.24 setX()

```
template<bool POINTVECTOR>
void cf::Vec3< POINTVECTOR >::setX (
            float value )  [inline]
```

setX Write to component 'x'

Parameters

value	

7.9.3.25 setY()

```
template<bool POINTVECTOR>
void cf::Vec3< POINTVECTOR >::setY (
            float value )  [inline]
```

setY Write to component 'y'

Parameters

value	

7.9.4 Friends And Related Function Documentation

7.9.4.1 operator*

```
template<bool POINTVECTOR>
cf::Vec3<POINTVECTOR> operator* (
            float lhs,
            const cf::Vec3< POINTVECTOR > & vec )  [friend]
```

7.9.4.2 operator<<)

```
template<bool POINTVECTOR>
template<bool b>
std::ostream& operator<<) (
             std::ostream & ,
             const Vec3< b > &  )  [friend]
```

7.9.4.3 Vec3<"!POINTVECTOR >

```
template<bool POINTVECTOR>
friend struct Vec3<!POINTVECTOR >  [friend]
```

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

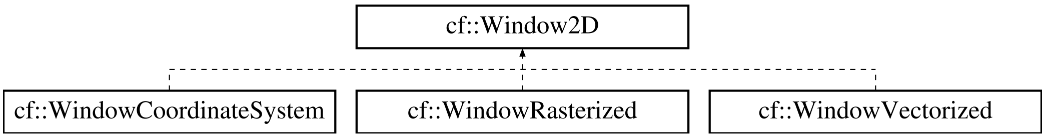
- include/computerGeometry.hpp

7.10 cf::Window2D Class Reference

The Window2D struct offers advanced features used by WindowRasterized/WindowVertorized.

```
#include <window2D.h>
```

Inheritance diagram for cf::Window2D:



Public Types

- enum LineType {
  LineType::DEFAULT = 0, LineType::DOT_0 = Window2D::DOT_VALUE | 1, LineType::DOT_1, LineType::DOT_2,
  LineType::DASH_0 = Window2D::DASH_VALUE | 1, LineType::DASH_1, LineType::DASH_2, LineType::DOT_DASH_0 = Window2D::DOT_VALUE | Window2D:: DASH_VALUE | 1,
  LineType::DOT_DASH_1, LineType::DOT_DASH_2 }

  The LineType enum Special line type used by one function of 'drawLine'.

Public Member Functions

- Window2D (int width=800, int height=600, const char ∗windowName="Chaos and Fractals", const cf::Color &startColor={0, 0, 0})
- Window2D (const char ∗filePath)
- virtual ∼Window2D ()
- void show () const

  show Show image, on first call it may require additional time to display content correctly (in those cases use wait←Key(1000))
- void clear (const cf::Color &color=cf::Color::WHITE)
- unsigned char waitKey (int delay=0) const

  waitKey Block access until key input on window
- void waitMouseInput (float &x, float &y)

  waitMouseInput Blocks until mouse input has been given
- void setWindowDisplayScale (float scale)

  setWindowDisplayScale Scales the image before displaying
- float getWindowDisplayScale () const
- void setInvertYAxis (bool invert)

  setInvertYAxis Invert y values on all 'cf::Point' functions
- bool getInvertYAxis () const
- void setColor (float x, float y, const Color &color)
- Color getColor (float x, float y) const
- void drawCircle (cf::Point center, int radius, int lineWidth, const cf::Color &color)

  drawCircle Draws a circle around the center
- void drawRectangle (cf::Point point1, cf::Point point2, int lineWidth, const cf::Color &color)

  drawRectangle Draws a rectangle from two diagonal points
- void drawLine (cf::Point point1, cf::Point point2, int lineWidth, const cf::Color &color)

  drawLine Draws a line from point1 to point2
- void drawSpecializedLine (cf::Point point1, cf::Point point2, LineType lineType, const cf::Color &color)

  drawSpecializedLine Draws specialized line of width 1 (dotted and/or dashed lines)
- void setNewInterval (const cf::Interval &intervalX, const cf::Interval &intervalY)

  setNewInterval Set new interval
- void resetInterval ()

  resetInterval Set default interval (interval x: [0, image widht - 1], interval y: [0, image height - 1])
- void saveImage (const char ∗filePath) const

  saveImage Saves current image to harddrive
- void resize (int pixelWidth, int pixelHeight)

  resize Resize underlying image
- void flippHorizontal ()

  flippHorizontal Flipp image horizontally
- void flippVertical ()

  flippHorizontal Flipp image vertically
- const cf::Interval & getIntervalX () const

  getIntervalX Const access to interval in x direction
- const cf::Interval & getIntervalY () const

  getIntervalY Const access to interval in y direction
- int getWidth () const

  getWidth Acess to underlying image width
- int getHeight () const

  getHeight Acess to underlying image height
- cv::Mat & getImage ()

  getImage Direct access to the underlying image

- void drawAxis (const cf::Color &color=cf::Color::BLACK, float stepSize_x=1.f, float stepSize_y=1.f, float interceptLength=3.f)

  drawAxis This function draws x and y axis based on Interval
- void drawCriclePart (cf::Point center, int radius, float startAngle, float endAngle, int lineWidth, const cf::Color &color=cf::Color::BLACK)

  drawCriclePart Draws a part of a circle
- void floodFill (cf::Point startingPoint, const cf::Color &color)

  floodFill Fills an area

Protected Member Functions

- void _correctYValue (float &y) const
- void _convertFromNewInterval (float &x, float &y) const
- void _convertToNewInterval (float &x, float &y) const
- void _window2foreground () const

Protected Attributes

- cv::Mat m_Image
- bool m_InvertYAxis
- const char * m_WindowName
- float m_WindowScale
- cf::Interval m_IntervalX
- cf::Interval m_IntervalY
- float m_MouseCallBackStorage [2]
- bool m_IntervalChanged = false
- bool m_FristShowCall = true

7.10.1 Detailed Description

The Window2D struct offers advanced features used by WindowRasterized/WindowVertorized.

7.10.2 Member Enumeration Documentation

7.10.2.1 LineType

```
enum cf::Window2D::LineType  [strong]
```

The LineType enum Special line type used by one function of 'drawLine'.

Enumerator

DEFAULT	
DOT_0	
DOT_1	
DOT_2	
DASH_0	
DASH_1	
DASH_2	
DOT_DASH↩ _0	
DOT_DASH↩ _1	
DOT_DASH↩	

7.10.3 Constructor & Destructor Documentation

7.10.3.1 Window2D() [1/2]

```
cf::Window2D::Window2D (
            int width = 800,
            int height = 600,
            const char * windowName = "Chaos and Fractals",
            const cf::Color & startColor = {0, 0, 0} )
```

7.10.3.2 Window2D() [2/2]

```
cf::Window2D::Window2D (
            const char * filePath )
```

7.10.3.3 ∼Window2D()

```
virtual cf::Window2D::∼Window2D ( )  [virtual]
```

7.10.4 Member Function Documentation

7.10.4.1 _convertFromNewInterval()

```
void cf::Window2D::_convertFromNewInterval (
            float & x,
            float & y ) const  [protected]
```

7.10.4.2 _convertToNewInterval()

```
void cf::Window2D::_convertToNewInterval (
            float & x,
            float & y ) const  [protected]
```

7.10.4.3 _correctYValue()

```
void cf::Window2D::_correctYValue (
            float & y ) const  [protected]
```

7.10.4.4 _window2foreground()

```
void cf::Window2D::_window2foreground ( ) const  [protected]
```

7.10.4.5 clear()

```
void cf::Window2D::clear (
            const cf::Color & color = cf::Color::WHITE )
```

7.10.4.6 drawAxis()

```
void cf::Window2D::drawAxis (
             const cf::Color & color = cf::Color::BLACK,
             float stepSize_x = 1.f,
             float stepSize_y = 1.f,
             float interceptLength = 3.f )
```

drawAxis This function draws x and y axis based on Interval

Parameters

color	Axis color, default is white
stepSize←↩_x	Dynamially set step size (x-axis), negative numbers indicate 10 steps for interval x
stepSize←↩_y	Dynamially set step size (y-axis), negative numbers indicate 10 steps for interval y

7.10.4.7 drawCircle()

```
void cf::Window2D::drawCircle (
             cf::Point center,
             int radius,
             int lineWidth,
             const cf::Color & color )
```

drawCircle Draws a circle around the center

Parameters

point	Point within interval_x and interval_y
radius	Circle radius in pixel (not effected by intervals)
lineWidth	Pixelwidth of line (not effected by intervals)
color	Circle color

7.10.4.8 drawCriclePart()

```
void cf::Window2D::drawCriclePart (
             cf::Point center,
             int radius,
             float startAngle,
             float endAngle,
             int lineWidth,
             const cf::Color & color = cf::Color::BLACK )
```

drawCriclePart Draws a part of a circle

Parameters

center	Center point of the circle

Parameters

radius	Radius of the circle
startAngle	Start position (in degrees)
endAngle	End position (in degrees)
color	Color of the drawn line

7.10.4.9 drawLine()

```
void cf::Window2D::drawLine (
             cf::Point point1,
             cf::Point point2,
             int lineWidth,
             const cf::Color & color )
```

drawLine Draws a line from point1 to point2

Parameters

point1	Point within interval_x and interval_y
point2	Point within interval_x and interval_y
lineWidth	Line width in pixel size
color	Line color

7.10.4.10 drawRectangle()

```
void cf::Window2D::drawRectangle (
             cf::Point point1,
             cf::Point point2,
             int lineWidth,
             const cf::Color & color )
```

drawRectangle Draws a rectangle from two diagonal points

Parameters

point1	Point within interval_x and interval_y, has to be the diagonal point to point2
point2	Point within interval_x and interval_y, has to be the diagonal point to point1
lineWidth	LineWidth pixelwidth of line (not effected by intervals)
color	Rectangle color

7.10.4.11 drawSpecializedLine()

```
void cf::Window2D::drawSpecializedLine (
             cf::Point point1,
             cf::Point point2,
             LineType lineType,
             const cf::Color & color )
```

drawSpecializedLine Draws specialized line of width 1 (dotted and/or dashed lines)

Parameters

point1	Point within interval_x and interval_y
point2	Point within interval_x and interval_y
lineType	Type of line to be drawn
color	Line color

7.10.4.12 flippHorizontal()

```
void cf::Window2D::flippHorizontal ( )
```

flippHorizontal Flipp image horizontally

7.10.4.13 flippVertical()

```
void cf::Window2D::flippVertical ( )
```

flippHorizontal Flipp image vertically

7.10.4.14 floodFill()

```
void cf::Window2D::floodFill (
             cf::Point startingPoint,
             const cf::Color & color )
```

floodFill Fills an area

Parameters

startingPoint	First point to be colored
color	Fill color

7.10.4.15 getColor()

```
Color cf::Window2D::getColor (
             float x,
             float y ) const
```

7.10.4.16 getHeight()

```
int cf::Window2D::getHeight ( ) const
```

getHeight Acess to underlying image height

Returns

Height

7.10.4.17 getImage()

```
cv::Mat& cf::Window2D::getImage ( )
```

getImage Direct access to the underlying image

Returns

Image handle

7.10.4.18 getIntervalX()

```
const cf::Interval& cf::Window2D::getIntervalX ( ) const
```

getIntervalX Const access to interval in x direction

Returns

7.10.4.19 getIntervalY()

```
const cf::Interval& cf::Window2D::getIntervalY ( ) const
```

getIntervalY Const access to interval in y direction

Returns

7.10.4.20 getInvertYAxis()

```
bool cf::Window2D::getInvertYAxis ( ) const
```

7.10.4.21 getWidth()

```
int cf::Window2D::getWidth ( ) const
```

getWidth Acess to underlying image width

Returns

Width

7.10.4.22 getWindowDisplayScale()

```
float cf::Window2D::getWindowDisplayScale ( ) const
```

7.10.4.23 resetInterval()

```
void cf::Window2D::resetInterval ( )
```

resetInterval Set default interval (interval x: [0, image widht - 1], interval y: [0, image height - 1])

7.10.4.24 resize()

```
void cf::Window2D::resize (
             int pixelWidth,
             int pixelHeight )
```

resize Resize underlying image

Parameters

pixelWidth	New width
pixelHeight	New height

7.10.4.25 saveImage()

```
void cf::Window2D::saveImage (
             const char * filePath ) const
```

saveImage Saves current image to harddrive

Parameters

filePath	File path and name, format will be determind based on file ending (∗.png, ∗.jpeg, ...)

7.10.4.26 setColor()

```
void cf::Window2D::setColor (
             float x,
             float y,
             const Color & color )
```

7.10.4.27 setInvertYAxis()

```
void cf::Window2D::setInvertYAxis (
             bool invert )
```

setInvertYAxis Invert y values on all 'cf::Point' functions

Parameters

invert	

7.10.4.28 setNewInterval()

```
void cf::Window2D::setNewInterval (
             const cf::Interval & intervalX,
             const cf::Interval & intervalY )
```

setNewInterval Set new interval

Parameters

intervalX	Interval in x direction
intervalY	Interval in y direction

7.10.4.29 setWindowDisplayScale()

```
void cf::Window2D::setWindowDisplayScale (
             float scale )
```

setWindowDisplayScale Scales the image before displaying

Parameters

scale	Window scale size

7.10.4.30 show()

```
void cf::Window2D::show ( ) const
```

show Show image, on first call it may require additional time to display content correctly (in those cases use waitKey(1000))

7.10.4.31 waitKey()

```
unsigned char cf::Window2D::waitKey (
             int delay = 0 ) const
```

waitKey Block access until key input on window

Parameters

delay	Value $>$ 0 -$>$ wait till key input on window or 'delay'ms else wait till user input

Returns

7.10.4.32 waitMouseInput()

```
void cf::Window2D::waitMouseInput (
             float & x,
             float & y )
```

waitMouseInput Blocks until mouse input has been given

Parameters

x	X-Window position
y	Y-Window position

7.10.5 Member Data Documentation

7.10.5.1 m_FristShowCall

```
bool cf::Window2D::m_FristShowCall = true  [mutable], [protected]
```

7.10.5.2 m_Image

```
cv::Mat cf::Window2D::m_Image  [protected]
```

7.10.5.3 m_IntervalChanged

```
bool cf::Window2D::m_IntervalChanged = false  [protected]
```

7.10.5.4 m_IntervalX

```
cf::Interval cf::Window2D::m_IntervalX  [protected]
```

7.10.5.5 m_IntervalY

```
cf::Interval cf::Window2D::m_IntervalY  [protected]
```

7.10.5.6 m_InvertYAxis

```
bool cf::Window2D::m_InvertYAxis  [protected]
```

7.10.5.7 m_MouseCallBackStorage

```
float cf::Window2D::m_MouseCallBackStorage[2]  [protected]
```

7.10.5.8 m_WindowName

```
const char* cf::Window2D::m_WindowName  [protected]
```

7.10.5.9 m_WindowScale

```
float cf::Window2D::m_WindowScale  [protected]
```

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

- include/window2D.h

7.11 cf::Window3D Struct Reference

The Window3D struct is the default class for accessing 3D content, creating more than 1 instance results in undefined behavior.

```
#include <window3D.h>
```

Public Types

- enum CameraType {
  CameraType::NONE, CameraType::ROTATION, CameraType::FREE_MOVEMENT, CameraType::STATI←
  C_X_AXIS,
  CameraType::STATIC_Y_AXIS, CameraType::STATIC_Z_AXIS }

  The CameraType enum providing access to camera types, default: 'CameraType::ROTATION'.

Public Member Functions

- Window3D (int ∗argc, char ∗∗argv, int width=800, int height=600, const char ∗title="chaos and fractals")
- virtual ∼Window3D ()
- void clear (const Color &color=Color::BLACK)
- virtual void draw ()=0

  draw Draw function, this has to be implemented
- virtual void handleKeyboardInput (unsigned char key, int x, int y)

  handleKeyboardInput Access key input by simple override this function
- int startDrawing ()

  startDrawing Start drawing, this function only returns afer 'ESC'-key press
- int getWindowWidth () const
- int getWindowHeight () const
- void setCamera (CameraType type, glm::vec3 lookAt=glm::vec3(0, 0, 0), float distance=10.f)

  setCamera Set or change current camera type
- void drawAxis (float length=10.f) const

  drawAxis Draw x-,y- and z-axis
- void forceDisplay () const

  forceDisplay Displays all content, it may be used for displaying the current process of the draw function
- void drawCylinder (const glm::vec3 &drawingDirection, const glm::vec3 &position, float diameter=1.f, const Color &color=Color::WHITE) const

  drawCylinder Draws a solid clynder
- void drawCylinder (const glm::vec4 &drawingDirection, const glm::vec3 &position, float diameter=1.f, const Color &color=Color::WHITE) const

  Type adjusted version of Window3D::drawCylinder.
- void drawCylinder (const glm::vec3 &drawingDirection, const glm::vec4 &position, float diameter=1.f, const Color &color=Color::WHITE) const

  Type adjusted version of Window3D::drawCylinder.
- void drawCylinder (const glm::vec4 &drawingDirection, const glm::vec4 &position, float diameter=1.f, const Color &color=Color::WHITE) const

  Type adjusted version of Window3D::drawCylinder.
- void setMaxFPS (float maxFPS=0.f)

  setMaxFPS Set maximum frames per second
- void enableLighting ()

  enableLighting Enable lightning (Default: lightning is enabled)
- void disableLighting ()

  disableLighting Disable lightning (Default: lightning is enabled)

Static Public Member Functions

- static void printWindowUsage ()

  printWindowUsage Print camera usage to console

Protected Member Functions

- void _AdjustCamera ()

Protected Attributes

- float m_DistAdjustment = 1.f
- float m_AngleAdjustment = 1.f
- float m_CameraAdjustment = 1.f
- glm::vec3 m_LookAt = glm::vec3(0.f, 0.f, 0.f)
- float m_LookAtDistance = 10.f
- float m_RotationAngle_Y = 0.f
- float m_RotationAngle_X = 0.f
- CameraType m_CameraType = Window3D::CameraType::ROTATION
- glm::vec3 m_FreeCamera_position = glm::vec3(0.f, 0.f, 0.f)

  CameraType::FREE_MOVEMENT specific member variables.
- glm::vec3 m_FreeCamera_UpVector = glm::vec3(0.f, 1.f, 0.f)
- glm::vec3 m_FreeCamera_LookDirection = glm::vec3(0.f, 0.f, 1.f)

Friends

- void _KeyboardCallbackFunction (unsigned char key, int x, int y)
- void _DrawingFunction ()

7.11.1 Detailed Description

The Window3D struct is the default class for accessing 3D content, creating more than 1 instance results in undefined behavior.

7.11.2 Member Enumeration Documentation

7.11.2.1 CameraType

```
enum cf::Window3D::CameraType  [strong]
```

The CameraType enum providing access to camera types, default: 'CameraType::ROTATION'.

Enumerator

NONE	
ROTATION	
FREE_MOVEMENT	
STATIC_X_AXIS	
STATIC_Y_AXIS	
STATIC_Z_AXIS	

7.11.3 Constructor & Destructor Documentation

7.11.3.1 Window3D()

```
cf::Window3D::Window3D (
                int * argc,
                char ** argv,
                int width = 800,
                int height = 600,
                const char * title = "chaos and fractals" )
```

7.11.3.2 ∼Window3D()

```
virtual cf::Window3D::∼Window3D ( ) [virtual]
```

7.11.4 Member Function Documentation

7.11.4.1 _AdjustCamera()

```
void cf::Window3D::_AdjustCamera ( ) [protected]
```

7.11.4.2 clear()

```
void cf::Window3D::clear (
             const Color & color = Color::BLACK )
```

7.11.4.3 disableLighting()

```
void cf::Window3D::disableLighting ( ) [inline]
```

disableLighting Disable lightning (Default: lightning is enabled)

7.11.4.4 draw()

```
virtual void cf::Window3D::draw ( ) [pure virtual]
```

draw Draw function, this has to be implemented

7.11.4.5 drawAxis()

```
void cf::Window3D::drawAxis (
             float length = 10.f ) const
```

drawAxis Draw x-,y- and z-axis

Parameters

length	Axis length

7.11.4.6 drawCylinder() [1/4]

```
void cf::Window3D::drawCylinder (
             const glm::vec3 & drawingDirection,
             const glm::vec3 & position,
             float diameter = 1.f,
             const Color & color = Color::WHITE ) const
```

drawCylinder Draws a solid clynder

Parameters

drawingDirection	Cylinder direction
position	Start position
diameter	Cylinder diamenter
color	Cylinder color

7.11.4.7 drawCylinder() [2/4]

```
void cf::Window3D::drawCylinder (
             const glm::vec4 & drawingDirection,
             const glm::vec3 & position,
             float diameter = 1.f,
             const Color & color = Color::WHITE ) const
```

Type adjusted version of Window3D::drawCylinder.

7.11.4.8 drawCylinder() [3/4]

```
void cf::Window3D::drawCylinder (
             const glm::vec3 & drawingDirection,
             const glm::vec4 & position,
             float diameter = 1.f,
             const Color & color = Color::WHITE ) const
```

Type adjusted version of Window3D::drawCylinder.

7.11.4.9 drawCylinder() [4/4]

```
void cf::Window3D::drawCylinder (
             const glm::vec4 & drawingDirection,
             const glm::vec4 & position,
             float diameter = 1.f,
             const Color & color = Color::WHITE ) const
```

Type adjusted version of Window3D::drawCylinder.

7.11.4.10 enableLighting()

```
void cf::Window3D::enableLighting ( )  [inline]
```

enableLighting Enable lightning (Default: lightning is enabled)

7.11.4.11 forceDisplay()

```
void cf::Window3D::forceDisplay ( ) const
```

forceDisplay Displays all content, it may be used for displaying the current process of the draw function

7.11.4.12 getWindowHeight()

```
int cf::Window3D::getWindowHeight ( ) const
```

7.11.4.13 getWindowWidth()

```
int cf::Window3D::getWindowWidth ( ) const
```

7.11.4.14 handleKeyboardInput()

```
virtual void cf::Window3D::handleKeyboardInput (
             unsigned char key,
             int x,
             int y )  [virtual]
```

handleKeyboardInput Access key input by simple override this function

Parameters

key	Key pressed
x	Mouse-x-position of the key press event
y	Mouse-y-position of the key press event

7.11.4.15 printWindowUsage()

```
static void cf::Window3D::printWindowUsage ( )  [static]
```

printWindowUsage Print camera usage to console

7.11.4.16 setCamera()

```
void cf::Window3D::setCamera (
             CameraType type,
             glm::vec3 lookAt = glm::vec3(0, 0, 0),
             float distance = 10.f )
```

setCamera Set or change current camera type

Parameters

type	Camera type
lookAt	
distance	

7.11.4.17 setMaxFPS()

```
void cf::Window3D::setMaxFPS (
            float maxFPS = 0.f )
```

setMaxFPS Set maximum frames per second

Parameters

maxFPS	values > 0 indicates capped fps, value of 0 indicates "only draw after key-input", 0 is default

7.11.4.18 startDrawing()

```
int cf::Window3D::startDrawing ( )
```

startDrawing Start drawing, this function only returns afer 'ESC'-key press

Returns

7.11.5 Friends And Related Function Documentation

7.11.5.1 _DrawingFunction

```
void _DrawingFunction ( ) [friend]
```

7.11.5.2 _KeyboardCallbackFunction

```
void _KeyboardCallbackFunction (
            unsigned char key,
            int x,
            int y ) [friend]
```

7.11.6 Member Data Documentation

7.11.6.1 m_AngleAdjustment

```
float cf::Window3D::m_AngleAdjustment = 1.f  [protected]
```

7.11.6.2 m_CameraAdjustment

```
float cf::Window3D::m_CameraAdjustment = 1.f  [protected]
```

7.11.6.3 m_CameraType

```
CameraType cf::Window3D::m_CameraType = Window3D::CameraType::ROTATION  [protected]
```

7.11.6.4 m_DistAdjustment

```
float cf::Window3D::m_DistAdjustment = 1.f  [protected]
```

7.11.6.5 m_FreeCamera_LookDirection

```
glm::vec3 cf::Window3D::m_FreeCamera_LookDirection = glm::vec3(0.f, 0.f, 1.f)  [protected]
```

7.11.6.6 m_FreeCamera_position

```
glm::vec3 cf::Window3D::m_FreeCamera_position = glm::vec3(0.f, 0.f, 0.f)  [protected]
```

CameraType::FREE_MOVEMENT specific member variables.

7.11.6.7 m_FreeCamera_UpVector

```
glm::vec3 cf::Window3D::m_FreeCamera_UpVector = glm::vec3(0.f, 1.f, 0.f)  [protected]
```

7.11.6.8 m_LookAt

```
glm::vec3 cf::Window3D::m_LookAt = glm::vec3(0.f, 0.f, 0.f)  [protected]
```

7.11.6.9 m_LookAtDistance

```
float cf::Window3D::m_LookAtDistance = 10.f  [protected]
```

7.11.6.10 m_RotationAngle_X

```
float cf::Window3D::m_RotationAngle_X = 0.f  [protected]
```

7.11.6.11 m_RotationAngle_Y

```
float cf::Window3D::m_RotationAngle_Y = 0.f  [protected]
```

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

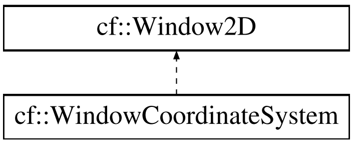
- include/window3D.h

7.12 cf::WindowCoordinateSystem Struct Reference

The WindowCoordinateSystem struct Default class for images and raster operations.

```
#include <windowCoordinateSystem.hpp>
```

Inheritance diagram for cf::WindowCoordinateSystem:



Public Types

- enum LineType

  The LineType enum Special line type used by one function of 'drawLine'.

Public Member Functions

- WindowCoordinateSystem (int width, const cf::Interval &range_x, const cf::Interval &range_y, const char ∗windowName="Computer Geometry", const cf::Color &startColor=cf::Color::WHITE)

  WindowCoordinateSystem Constructor.
- virtual ∼WindowCoordinateSystem ()=default
- void setInterval (const cf::Interval &range_x, const cf::Interval &range_y, int width)

  setInterval Set new interval
- void drawPoint (const cf::Point &pos, const cf::Color &color=cf::Color::BLACK)

  drawPoint Draws a cross-shaped point
- void drawLine (const cf::Point &p1, const cf::Point &p2, const cf::Color &color=cf::Color::BLACK, cf::Window2D::LineType type=cf::Window2D::LineType::DEFAULT, int lineWidth=1)

  drawLine Draw a simple line of width 1
- void drawLinearEquation (const cf::Point &pointVector, const glm::vec3 &drawingDirection, const cf::Color &color=cf::Color::BLACK, cf::Window2D::LineType type=cf::Window2D::LineType::DEFAULT, int lineWidth=1)

  drawLinearEquation Draws a line from a point on line and direction vector

- void drawLinearEquation (float a, float b, float c, const cf::Color &color=cf::Color::BLACK, cf::Window2D::↩ LineType type=cf::Window2D::LineType::DEFAULT, int lineWidth=1)

  drawLinearEquation Draw a line from a linear equation: ax + by + c = 0
- void drawLinearEquation (const glm::vec3 &vec, const cf::Color &color=cf::Color::BLACK, cf::Window2D::↩ LineType type=cf::Window2D::LineType::DEFAULT, int lineWidth=1)

  drawLinearEquation Draw line from linear equation: ax + by + c = 0, where a b and c are part of coefficent vector
- void drawLinearEquation (float slope, float yIntercept, const cf::Color &color=cf::Color::BLACK, cf::↩ Window2D::LineType type=cf::Window2D::LineType::DEFAULT, int lineWidth=1)

  drawLinearEquation Draw line from standard format y = m∗x + t
- void drawCircle (const cf::Point &center, float radius, const cf::Color &color=cf::Color::BLACK, int lineWidth=1)

  drawCircle Draws a circle with interval radius
- float convert_pixelLength_to_intervalLength (float pixelLength) const

  convert_pixelLength_to_intervalLength Converts length from pixel to interval
- float convert_intervalLength_to_pixelLength (float intervalLength) const

  convert_intervalLength_to_pixelLength Converts length from interval to pixel
- void drawCriclePart (const cf::Point &center, float radius, float startAngle, float endAngle, const cf::Color &color=cf::Color::BLACK, int lineWidth=1)

  drawCriclePart Draw a partition of a circle

Additional Inherited Members

7.12.1 Detailed Description

The WindowCoordinateSystem struct Default class for images and raster operations.

7.12.2 Member Enumeration Documentation

7.12.2.1 LineType

```
enum cf::Window2D::LineType  [strong]
```

The LineType enum Special line type used by one function of 'drawLine'.

7.12.3 Constructor & Destructor Documentation

7.12.3.1 WindowCoordinateSystem()

```
cf::WindowCoordinateSystem::WindowCoordinateSystem (
             int width,
             const cf::Interval & range_x,
             const cf::Interval & range_y,
             const char * windowName = "Computer Geometry",
             const cf::Color & startColor = cf::Color::WHITE )  [inline]
```

WindowCoordinateSystem Constructor.

Parameters

range↩ _x	Interval in x direction
range↩ _y	Interval in y direction
width	Image width in pixel (hight will be determind automatically)

7.12.3.2 ∼WindowCoordinateSystem()

```
virtual cf::WindowCoordinateSystem::∼WindowCoordinateSystem ( )  [virtual], [default]
```

7.12.4 Member Function Documentation

7.12.4.1 convert_intervalLength_to_pixelLength()

```
float cf::WindowCoordinateSystem::convert_intervalLength_to_pixelLength (
             float intervalLength ) const  [inline]
```

convert_intervalLength_to_pixelLength Converts length from interval to pixel

Parameters

intervalLength	

Returns

7.12.4.2 convert_pixelLength_to_intervalLength()

```
float cf::WindowCoordinateSystem::convert_pixelLength_to_intervalLength (
             float pixelLength ) const  [inline]
```

convert_pixelLength_to_intervalLength Converts length from pixel to interval

Parameters

pixelLength	

Returns

7.12.4.3 drawCircle()

```
void cf::WindowCoordinateSystem::drawCircle (
             const cf::Point & center,
             float radius,
             const cf::Color & color = cf::Color::BLACK,
             int lineWidth = 1 )  [inline]
```

drawCircle Draws a circle with interval radius

Parameters

center	Circle center
radius	Circle radius
color	Circle color
lineWidth	Width of the line, Note: only available on default line type

7.12.4.4 drawCriclePart()

```
void cf::WindowCoordinateSystem::drawCriclePart (
             const cf::Point & center,
             float radius,
             float startAngle,
             float endAngle,
             const cf::Color & color = cf::Color::BLACK,
             int lineWidth = 1 )  [inline]
```

drawCriclePart Draw a partition of a circle

Parameters

center	Circle center
radius	Circle radius (in intervall length)
startAngle	Starting angle for circle (0° -> positive x direction, 90° -> positive y direction)
endAngle	End angle for circle (0° -> positive x-axis, 90° -> positive y-axis)
color	Circle color
lineWidth	Line width of the circle

7.12.4.5 drawLine()

```
void cf::WindowCoordinateSystem::drawLine (
             const cf::Point & p1,
             const cf::Point & p2,
             const cf::Color & color = cf::Color::BLACK,
             cf::Window2D::LineType type = cf::Window2D::LineType::DEFAULT,
             int lineWidth = 1 )  [inline]
```

drawLine Draw a simple line of width 1

Parameters

p1	First point
p2	Second point
color	Line color
type	Line type
lineWidth	Width of the line, Note: only available on default line type

7.12.4.6 drawLinearEquation() [1/4]

```
void cf::WindowCoordinateSystem::drawLinearEquation (
            const cf::Point & pointVector,
            const glm::vec3 & drawingDirection,
            const cf::Color & color = cf::Color::BLACK,
            cf::Window2D::LineType type = cf::Window2D::LineType::DEFAULT,
            int lineWidth = 1 )  [inline]
```

drawLinearEquation Draws a line from a point on line and direction vector

Parameters

pointVector	Point on the line
drawingDirection	Line direction
color	Line color
type	Change line type to dot/dash/dot-dash
lineWidth	Width of the line, Note: only available on default line type

7.12.4.7 drawLinearEquation() [2/4]

```
void cf::WindowCoordinateSystem::drawLinearEquation (
            float a,
            float b,
            float c,
            const cf::Color & color = cf::Color::BLACK,
            cf::Window2D::LineType type = cf::Window2D::LineType::DEFAULT,
            int lineWidth = 1 )  [inline]
```

drawLinearEquation Draw a line from a linear equation: ax + by + c = 0

Parameters

a	Coefficent of x
b	Coefficent of y
c	Constant
color	Line color
type	Change line type to dot/dash/dot-dash
lineWidth	Width of the line, Note: only available on default line type

7.12.4.8 drawLinearEquation() [3/4]

```
void cf::WindowCoordinateSystem::drawLinearEquation (
            const glm::vec3 & vec,
            const cf::Color & color = cf::Color::BLACK,
            cf::Window2D::LineType type = cf::Window2D::LineType::DEFAULT,
            int lineWidth = 1 )  [inline]
```

drawLinearEquation Draw line from linear equation: ax + by + c = 0, where a b and c are part of coefficent vector

Parameters

vec	Vector of cooefficents a b and see
color	Line color
type	Change line type to dot/dash/dot-dash
lineWidth	Width of the line, Note: only available on default line type

7.12.4.9 drawLinearEquation() [4/4]

```
void cf::WindowCoordinateSystem::drawLinearEquation (
            float slope,
            float yIntercept,
            const cf::Color & color = cf::Color::BLACK,
            cf::Window2D::LineType type = cf::Window2D::LineType::DEFAULT,
            int lineWidth = 1 )  [inline]
```

drawLinearEquation Draw line from standard format y = m∗x + t

Parameters

slope	Slope m of equation y = m∗x + t
yIntercept	y-Intercept t of equation y = m∗x + t
color	Line color
type	Change line type to dot/dash/dot-dash
lineWidth	Width of the line, Note: only available on default line type

7.12.4.10 drawPoint()

```
void cf::WindowCoordinateSystem::drawPoint (
            const cf::Point & pos,
            const cf::Color & color = cf::Color::BLACK )  [inline]
```

drawPoint Draws a cross-shaped point

Parameters

pos	Cross position
color	Cross color

7.12.4.11 setInterval()

```
void cf::WindowCoordinateSystem::setInterval (
             const cf::Interval & range_x,
             const cf::Interval & range_y,
             int width )  [inline]
```

setInterval Set new interval

Parameters

range↵_x	Interval in x direction
range↵_y	Interval in y direction
width	Image width in pixel (hight will be determind automatically)

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

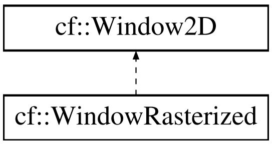
- include/windowCoordinateSystem.hpp

7.13 cf::WindowRasterized Struct Reference

The WindowRasterized struct Default struct for verctorized operations within a custom interval.

```
#include <windowRasterized.hpp>
```

Inheritance diagram for cf::WindowRasterized:



Public Types

- enum LineType

  The LineType enum Special line type used by one function of 'drawLine'.

Public Member Functions

- WindowRasterized (int width=800, int height=600, const char ∗windowName="Chaos and Fractals", const cf::Color &startColor={0, 0, 0})

  WindowRasterized Constructor.
- WindowRasterized (const char ∗filePath)

  WindowRasterized Load image from file path.
- virtual ∼WindowRasterized ()=default

Additional Inherited Members

7.13.1 Detailed Description

The WindowRasterized struct Default struct for verctorized operations within a custom interval.

7.13.2 Member Enumeration Documentation

7.13.2.1 LineType

```
enum cf::Window2D::LineType  [strong]
```

The LineType enum Special line type used by one function of 'drawLine'.

7.13.3 Constructor & Destructor Documentation

7.13.3.1 WindowRasterized() [1/2]

```
cf::WindowRasterized::WindowRasterized (
             int width = 800,
             int height = 600,
             const char * windowName = "Chaos and Fractals",
             const cf::Color & startColor = {0, 0, 0} )  [inline]
```

WindowRasterized Constructor.

Parameters

width	Pixel width of the image
height	Pixel height of the image
windowName	Name of the window
startColor	Background color

7.13.3.2 WindowRasterized() [2/2]

```
cf::WindowRasterized::WindowRasterized (
             const char * filePath )  [inline]
```

WindowRasterized Load image from file path.

Parameters

filePath	Path to file

7.13.3.3 ∼WindowRasterized()

```
virtual cf::WindowRasterized::∼WindowRasterized ( )  [virtual], [default]
```

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

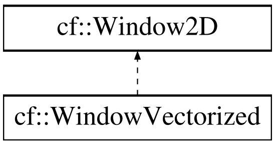
- include/windowRasterized.hpp

7.14 cf::WindowVectorized Struct Reference

The WindowVectorized struct Default class for images and raster operations.

```
#include <windowVercorized.hpp>
```

Inheritance diagram for cf::WindowVectorized:



Public Types

- enum LineType

  The LineType enum Special line type used by one function of 'drawLine'.

Public Member Functions

- WindowVectorized (int width, const cf::Interval &range_x, const cf::Interval &range_y, const char ∗window↩ Name="Chaos and Fractals", const cf::Color &startColor=cf::Color::BLACK)

  WindowVectorized Constructor.
- WindowVectorized (const char ∗filePath, int width, const cf::Interval &range_x, const cf::Interval &range_y)

  WindowVectorized Image reading constructoor.
- virtual ∼WindowVectorized ()=default
- void setInterval (const cf::Interval &range_x, const cf::Interval &range_y, int width)

  setInterval Set new interval
- cf::Point transformPoint_fromInterval_toImage (cf::Point point)

  transformPoint_fromInterval_toImage Transform point from interval position to pixel position
- cf::Point transformPoint_fromImage_toInterval (cf::Point point)

  transformPoint_fromImage_toInterval Transform point from pixel position to interval position
- float convert_pixelLength_to_intervalLength (float pixelLength) const

  convert_pixelLength_to_intervalLength Converts length from pixel to interval
- float convert_intervalLength_to_pixelLength (float intervalLength) const

  convert_intervalLength_to_pixelLength Converts length from interval to pixel
- cf::Color getColor_imageSpace (int i, int j) const

  getColor_imageSpace Get color from image i/j position
- void setColor_imageSpace (int i, int j, const cf::Color &color)

  setColor_imageSpace Set color from image i/j position

Additional Inherited Members

7.14.1 Detailed Description

The WindowVectorized struct Default class for images and raster operations.

7.14.2 Member Enumeration Documentation

7.14.2.1 LineType

```
enum cf::Window2D::LineType  [strong]
```

The LineType enum Special line type used by one function of 'drawLine'.

7.14.3 Constructor & Destructor Documentation

7.14.3.1 WindowVectorized() [1/2]

```
cf::WindowVectorized::WindowVectorized (
             int width,
             const cf::Interval & range_x,
             const cf::Interval & range_y,
             const char * windowName = "Chaos and Fractals",
             const cf::Color & startColor = cf::Color::BLACK )  [inline]
```

WindowVectorized Constructor.

Parameters

width	Image width in pixel (hight will be determind automatically)
range_x	Interval in x direction
range_y	Interval in y direction

7.14.3.2 WindowVectorized() [2/2]

```
cf::WindowVectorized::WindowVectorized (
             const char * filePath,
             int width,
             const cf::Interval & range_x,
             const cf::Interval & range_y )  [inline]
```

WindowVectorized Image reading constructoor.

Parameters

filePath	Path to image file

Parameters

width	Image width, Note: height will be calculated based on ranges and width
range↩ _x	Interval in x direction
range↩ _y	Interval in y direction

7.14.3.3 ∼WindowVectorized()

```
virtual cf::WindowVectorized::∼WindowVectorized ( )  [virtual], [default]
```

7.14.4 Member Function Documentation

7.14.4.1 convert_intervalLength_to_pixelLength()

```
float cf::WindowVectorized::convert_intervalLength_to_pixelLength (
             float intervalLength ) const  [inline]
```

convert_intervalLength_to_pixelLength Converts length from interval to pixel

Parameters

intervalLength	Length to be converted to pixel length

Returns

7.14.4.2 convert_pixelLength_to_intervalLength()

```
float cf::WindowVectorized::convert_pixelLength_to_intervalLength (
             float pixelLength ) const  [inline]
```

convert_pixelLength_to_intervalLength Converts length from pixel to interval

Parameters

pixelLength	Length to be converted to the intervall length

Returns

7.14.4.3 getColor_imageSpace()

```
cf::Color cf::WindowVectorized::getColor_imageSpace (
             int i,
             int j ) const  [inline]
```

getColor_imageSpace Get color from image i/j position

Parameters

i	I position
j	J position

Returns

7.14.4.4 setColor_imageSpace()

```
void cf::WindowVectorized::setColor_imageSpace (
             int i,
             int j,
             const cf::Color & color )  [inline]
```

setColor_imageSpace Set color from image i/j position

Parameters

i	I position
j	J position
color	Color to be set

7.14.4.5 setInterval()

```
void cf::WindowVectorized::setInterval (
             const cf::Interval & range_x,
             const cf::Interval & range_y,
             int width )  [inline]
```

setInterval Set new interval

Parameters

range←_x	Interval in x direction
range←_y	Interval in y direction
width	Image width in pixel (hight will be determind automatically)

7.14.4.6 transformPoint_fromImage_toInterval()

```
cf::Point cf::WindowVectorized::transformPoint_fromImage_toInterval (
            cf::Point point )  [inline]
```

transformPoint_fromImage_toInterval Transform point from pixel position to interval position

Parameters

point	Point to be transformed

Returns

Transformed point

7.14.4.7 transformPoint_fromInterval_toImage()

```
cf::Point cf::WindowVectorized::transformPoint_fromInterval_toImage (
            cf::Point point )  [inline]
```

transformPoint_fromInterval_toImage Transform point from interval position to pixel position

Parameters

point	Point to be transformed

Returns

Transformed point

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

- include/windowVercorized.hpp

Chapter 8

File Documentation

8.1 include/computerGeometry.hpp File Reference

```
#include "windowCoordinateSystem.hpp"
#include "utils.h"
#include <sstream>
#include <fstream>
#include <string>
```

Classes

- struct cf::Vec3< POINTVECTOR >

  The Vec3 struct General class for vector operations.
- struct cf::Vec3< POINTVECTOR >

  The Vec3 struct General class for vector operations.

Namespaces

- cf

Typedefs

- typedef Vec3< true > cf::PointVector

  PointVector Specialiaztion of general Vec3.
- typedef Vec3< false > cf::DirectionVector

  DirectionVector Specialiaztion of general Vec3, where component 'w' may not be written to.

Functions

- template<bool b>
  std::ostream & operator<< (std::ostream &os, const cf::Vec3< b > &rhs)

  operator<< Simple shift operator for output

8.1.1 Function Documentation

8.1.1.1 operator<<()

```
template<bool b>
std::ostream & operator<< (
            std::ostream & os,
            const cf::Vec3< b > & rhs )
```

operator<< Simple shift operator for output

Parameters

os	Outputstream, e.g. std::cout
rhs	cf::PointVector or cf::DirectionVector

Returns

8.2 include/IFS.h File Reference

```
#include "utils.h"
```

Classes

- struct cf::IteratedFunctionSystem

  The IteratedFunctionSystem class lazy people (like myself) may use the IFS tyepdef.

Namespaces

- cf

Typedefs

- typedef IteratedFunctionSystem cf::IFS

8.3 include/LSystem.h File Reference

```
#include <string>
#include <vector>
#include <memory>
#include <glm/glm.hpp>
#include "utils.h"
```

Classes

- struct cf::LindenmayerSystem

    The LindenmayerSystem class lazy people (like myself) may use the IFS tyepdef.

Namespaces

- cf

Typedefs

- typedef LindenmayerSystem cf::LSystem

8.4 include/ORB.h File Reference

```
#include "utils.h"
```

Classes

- struct cf::Orbit

    The Orbit class lazy people (like myself) may use the ORB tyepdef.

Namespaces

- cf

Typedefs

- typedef Orbit cf::ORB

8.5 include/utils.h File Reference

```
#include <string>
#include <vector>
#include <fstream>
#include <sstream>
#include <iostream>
#include <inttypes.h>
#include <glm/glm.hpp>
#include <glm/gtx/transform.hpp>
#include <glm/gtx/vector_angle.hpp>
#include <glm/gtx/rotate_vector.hpp>
```

Classes

- struct cf::Direction

  The Direction struct for getting absolute directions from a current direction and a relative direction.
- struct cf::Interval

  The Interval struct provides functionallity to translate values from one interval into another.
- struct cf::Color

  The Color struct offers a class for rgb access.
- struct cf::Console

  The Console struct offers utility functions for 'console'.

Namespaces

- cf

Functions

- std::ostream & operator<< (std::ostream &of, const glm::vec2 &vec)
- std::ostream & operator<< (std::ostream &of, const glm::vec3 &vec)
- std::ostream & operator<< (std::ostream &of, const glm::vec4 &vec)
- std::ostream & operator<< (std::ostream &of, const glm::mat3x3 &mat)
- std::ostream & operator<< (std::ostream &of, const glm::mat4x4 &mat)
- void cf::_removeWindowsSpecificCarriageReturn (std::string &str)

  _removeWindowsSpecificCarriageReturn Removes 'carriage return' characters in strings ('carriage return' may be read from unix system by providing windows files)
- std::vector< Color > cf::readPaletteFromFile (const std::string &filePath)

  readPaletteFromFile
- std::string cf::readAntString (const std::string &filePath)

  readAntString
- template<typename _VectorType = glm::vec3>
  std::vector< _VectorType > cf::readDATFile (const std::string &filePath)

  readDATFile Reads a ∗.dat file
- float cf::radian2degree (float radianValue)

  radian2degree Converts a radian value to a degree value
- float cf::degree2radian (float degreeValue)

  degree2radian Converts a degree value to a radian value

8.5.1 Function Documentation

8.5.1.1 operator<<() [1/5]

```
std::ostream& operator<< (
            std::ostream & of,
            const glm::vec2 & vec )
```

8.5.1.2 operator<<() [2/5]

```
std::ostream& operator<< (
            std::ostream & of,
            const glm::vec3 & vec )
```

8.5.1.3 operator<<() [3/5]

```
std::ostream& operator<< (
            std::ostream & of,
            const glm::vec4 & vec )
```

8.5.1.4 operator<<() [4/5]

```
std::ostream& operator<< (
            std::ostream & of,
            const glm::mat3x3 & mat )
```

8.5.1.5 operator<<() [5/5]

```
std::ostream& operator<< (
            std::ostream & of,
            const glm::mat4x4 & mat )
```

8.6 include/window2D.h File Reference

```
#include <opencv2/opencv.hpp>
#include "utils.h"
```

Classes

- class cf::Window2D

  The Window2D struct offers advanced features used by WindowRasterized/WindowVertorized.
- struct cf::Point

  The Point struct is a simple class for positon access on 2D images (imilar to cv::Point, but uses floats instead of integer)

Namespaces

- cf

8.7 include/window3D.h File Reference

```
#include <GL/freeglut.h>
#include <functional>
#include <vector>
#include <string>
#include "utils.h"
```

Classes

- struct cf::Window3D

  The Window3D struct is the default class for accessing 3D content, creating more than 1 instance results in undefined behavior.

Namespaces

- cf

8.8 include/windowCoordinateSystem.hpp File Reference

```
#include "window2D.h"
```

Classes

- struct cf::WindowCoordinateSystem

  The WindowCoordinateSystem struct Default class for images and raster operations.

Namespaces

- cf

8.9 include/windowRasterized.hpp File Reference

```
#include "window2D.h"
```

Classes

- struct cf::WindowRasterized

  The WindowRasterized struct Default struct for verctorized operations within a custom interval.

Namespaces

- cf

8.10 include/windowVercorized.hpp File Reference

```
#include "window2D.h"
```

Classes

- struct cf::WindowVectorized

  The WindowVectorized struct Default class for images and raster operations.

Namespaces

- cf

8.11 README.md File Reference

Index