

CF_CG-Lib

1.0

Generated by Doxygen 1.8.12

Contents

1	CF_CG-Lib	1
2	Namespace Index	3
2.1	Namespace List	3
3	Hierarchical Index	5
3.1	Class Hierarchy	5
4	Class Index	7
4.1	Class List	7
5	File Index	9
5.1	File List	9
6	Namespace Documentation	11
6.1	cf Namespace Reference	11
6.1.1	Typedef Documentation	12
6.1.1.1	DirectionVector	12
6.1.1.2	IFS	12
6.1.1.3	LSystem	13
6.1.1.4	ORB	13
6.1.1.5	PointVector	13
6.1.2	Function Documentation	13
6.1.2.1	_removeWindowsSpecificCarriageReturn()	13
6.1.2.2	degree2radian()	13
6.1.2.3	radian2degree()	14
6.1.2.4	readAntString()	14
6.1.2.5	readDATFile()	14
6.1.2.6	readPaletteFromFile()	15

7	Class Documentation	17
7.1	cf::Circle Struct Reference	17
7.1.1	Detailed Description	17
7.1.2	Constructor & Destructor Documentation	17
7.1.2.1	Circle()	17
7.1.3	Member Data Documentation	18
7.1.3.1	center	18
7.1.3.2	color	18
7.1.3.3	lineWidth	18
7.1.3.4	radius	18
7.2	cf::CirclePartition Struct Reference	18
7.2.1	Detailed Description	18
7.2.2	Constructor & Destructor Documentation	19
7.2.2.1	CirclePartition()	19
7.2.3	Member Data Documentation	19
7.2.3.1	center	19
7.2.3.2	color	19
7.2.3.3	endAngle	19
7.2.3.4	lineWidth	19
7.2.3.5	radius	19
7.2.3.6	startAngle	19
7.3	cf::Color Struct Reference	20
7.3.1	Detailed Description	21
7.3.2	Constructor & Destructor Documentation	21
7.3.2.1	Color()	21
7.3.3	Member Function Documentation	21
7.3.3.1	invert()	21
7.3.3.2	operator!=()	21
7.3.3.3	operator*()	21
7.3.3.4	operator*=()	21

7.3.3.5	operator+()	22
7.3.3.6	operator+=()	22
7.3.3.7	operator-()	22
7.3.3.8	operator-=()	22
7.3.3.9	operator/()	22
7.3.3.10	operator/=()	22
7.3.3.11	operator<()	22
7.3.3.12	operator<=()	22
7.3.3.13	operator==()	22
7.3.3.14	operator>()	23
7.3.3.15	operator>=()	23
7.3.3.16	RandomColor()	23
7.3.4	Friends And Related Function Documentation	23
7.3.4.1	operator*	23
7.3.4.2	operator/	23
7.3.4.3	operator<<	23
7.3.5	Member Data Documentation	23
7.3.5.1	b	23
7.3.5.2	BLACK	24
7.3.5.3	BLUE	24
7.3.5.4	CYAN	24
7.3.5.5	g	24
7.3.5.6	GREEN	24
7.3.5.7	GREY	24
7.3.5.8	MAGENTA	24
7.3.5.9	ORANGE	24
7.3.5.10	PINK	24
7.3.5.11	r	24
7.3.5.12	RED	25
7.3.5.13	WHITE	25

7.3.5.14	YELLOW	25
7.4	cf::Console Struct Reference	25
7.4.1	Detailed Description	25
7.4.2	Member Function Documentation	25
7.4.2.1	clearConsole()	25
7.4.2.2	readFloat()	26
7.4.2.3	readInt()	26
7.4.2.4	readString()	26
7.4.2.5	waitKey()	26
7.5	cf::Direction Struct Reference	26
7.5.1	Detailed Description	27
7.5.2	Member Enumeration Documentation	27
7.5.2.1	AbsoluteDirection	27
7.5.2.2	RelativeDirection	27
7.5.3	Member Function Documentation	28
7.5.3.1	getNextDirection()	28
7.5.3.2	toString() [1/2]	28
7.5.3.3	toString() [2/2]	28
7.6	cf::Interval Struct Reference	28
7.6.1	Detailed Description	29
7.6.2	Constructor & Destructor Documentation	29
7.6.2.1	Interval()	29
7.6.3	Member Function Documentation	29
7.6.3.1	translateIntervalPostion() [1/2]	29
7.6.3.2	translateIntervalPostion() [2/2]	29
7.6.4	Friends And Related Function Documentation	29
7.6.4.1	operator<<	29
7.6.5	Member Data Documentation	30
7.6.5.1	max	30
7.6.5.2	min	30

7.7	cf::IteratedFunctionSystem Struct Reference	30
7.7.1	Detailed Description	30
7.7.2	Member Function Documentation	30
7.7.2.1	getAllTransformation()	30
7.7.2.2	getName()	30
7.7.2.3	getNumTransformations()	31
7.7.2.4	getRangeX()	31
7.7.2.5	getRangeY()	31
7.7.2.6	getTransformation()	31
7.7.2.7	read()	31
7.8	cf::LSystem_Controller::iterator Struct Reference	31
7.8.1	Member Function Documentation	32
7.8.1.1	operator"!="()	32
7.8.1.2	operator*()	32
7.8.1.3	operator++()	32
7.8.2	Friends And Related Function Documentation	32
7.8.2.1	LSystem_Controller	32
7.9	cf::LindenmayerSystem Struct Reference	32
7.9.1	Detailed Description	33
7.9.2	Member Function Documentation	33
7.9.2.1	clearWindowEachTime()	33
7.9.2.2	getAdjustmentAngle()	33
7.9.2.3	getAllProductions()	33
7.9.2.4	getAxiom()	33
7.9.2.5	getName()	33
7.9.2.6	getNumProductions()	33
7.9.2.7	getProduction()	33
7.9.2.8	getRangeX()	33
7.9.2.9	getRangeY()	33
7.9.2.10	getScale()	34

7.9.2.11	getStartAngle()	34
7.9.2.12	read()	34
7.10	cf::Line Struct Reference	34
7.10.1	Detailed Description	34
7.10.2	Constructor & Destructor Documentation	35
7.10.2.1	Line()	35
7.10.3	Member Data Documentation	35
7.10.3.1	color	35
7.10.3.2	lineType	35
7.10.3.3	lineWidth	35
7.10.3.4	point1	35
7.10.3.5	point2	35
7.11	cf::LSystem_Controller Struct Reference	35
7.11.1	Detailed Description	36
7.11.2	Constructor & Destructor Documentation	36
7.11.2.1	LSystem_Controller()	36
7.11.3	Member Function Documentation	36
7.11.3.1	begin()	36
7.11.3.2	end()	36
7.12	cf::Orbit Struct Reference	37
7.12.1	Detailed Description	37
7.12.2	Member Function Documentation	37
7.12.2.1	getAllFactors()	37
7.12.2.2	getAllStartingPoints()	37
7.12.2.3	getName()	37
7.12.2.4	getNumFactors()	37
7.12.2.5	getNumStartingPoints()	37
7.12.2.6	getRangeX()	38
7.12.2.7	getRangeY()	38
7.12.2.8	read()	38

7.13	cf::Point Struct Reference	38
7.13.1	Detailed Description	39
7.13.2	Constructor & Destructor Documentation	39
7.13.2.1	Point()	39
7.13.3	Member Function Documentation	39
7.13.3.1	operator cv::Point()	39
7.13.3.2	operator!=()	39
7.13.3.3	operator*()	39
7.13.3.4	operator*=()	39
7.13.3.5	operator+()	40
7.13.3.6	operator+=()	40
7.13.3.7	operator-()	40
7.13.3.8	operator-=()	40
7.13.3.9	operator/()	40
7.13.3.10	operator/=()	40
7.13.3.11	operator==()	40
7.13.4	Friends And Related Function Documentation	40
7.13.4.1	operator*	40
7.13.4.2	operator/	41
7.13.5	Member Data Documentation	41
7.13.5.1	x	41
7.13.5.2	y	41
7.14	cf::Rect Struct Reference	41
7.14.1	Detailed Description	41
7.14.2	Constructor & Destructor Documentation	42
7.14.2.1	Rect()	42
7.14.3	Member Data Documentation	42
7.14.3.1	color	42
7.14.3.2	lineWidth	42
7.14.3.3	point1	42

7.14.3.4	point2	42
7.15	cf::Vec3< POINTVECTOR > Struct Template Reference	42
7.15.1	Detailed Description	44
7.15.2	Constructor & Destructor Documentation	44
7.15.2.1	Vec3() [1/3]	44
7.15.2.2	Vec3() [2/3]	44
7.15.2.3	Vec3() [3/3]	44
7.15.3	Member Function Documentation	45
7.15.3.1	getW()	45
7.15.3.2	getX()	45
7.15.3.3	getY()	45
7.15.3.4	isPointVector()	45
7.15.3.5	normalize()	46
7.15.3.6	operator cf::Point()	46
7.15.3.7	operator cf::Vec3< false >()	46
7.15.3.8	operator const glm::vec3 &()	46
7.15.3.9	operator glm::vec3()	46
7.15.3.10	operator%()	46
7.15.3.11	operator%=()	47
7.15.3.12	operator*() [1/2]	47
7.15.3.13	operator*() [2/2]	47
7.15.3.14	operator*=()	47
7.15.3.15	operator+()	48
7.15.3.16	operator+=()	48
7.15.3.17	operator-()	48
7.15.3.18	operator-=()	48
7.15.3.19	operator=() [1/2]	48
7.15.3.20	operator=() [2/2]	48
7.15.3.21	operator[]() [1/2]	48
7.15.3.22	operator[]() [2/2]	49

7.15.3.23 setW()	49
7.15.3.24 setX()	49
7.15.3.25 setY()	50
7.15.4 Friends And Related Function Documentation	50
7.15.4.1 operator*	50
7.15.4.2 operator<<()	50
7.15.4.3 Vec3<"!POINTVECTOR >	50
7.16 cf::Window2D Class Reference	50
7.16.1 Detailed Description	53
7.16.2 Member Enumeration Documentation	53
7.16.2.1 LineType	53
7.16.3 Constructor & Destructor Documentation	53
7.16.3.1 Window2D() [1/2]	53
7.16.3.2 Window2D() [2/2]	53
7.16.3.3 ~Window2D()	53
7.16.4 Member Function Documentation	54
7.16.4.1 _convertFromNewInterval()	54
7.16.4.2 _convertToNewInterval()	54
7.16.4.3 _correctYValue()	54
7.16.4.4 _window2foreground()	54
7.16.4.5 clear()	54
7.16.4.6 drawAxis()	54
7.16.4.7 drawCircle() [1/2]	55
7.16.4.8 drawCircle() [2/2]	55
7.16.4.9 drawCirclePart() [1/2]	55
7.16.4.10 drawCirclePart() [2/2]	56
7.16.4.11 drawLine() [1/2]	56
7.16.4.12 drawLine() [2/2]	56
7.16.4.13 drawRectangle() [1/2]	56
7.16.4.14 drawRectangle() [2/2]	58

7.16.4.15 drawSpecializedLine()	58
7.16.4.16 flippHorizontal()	58
7.16.4.17 flippVertical()	58
7.16.4.18 floodFill()	59
7.16.4.19 getColor()	59
7.16.4.20 getHeight()	59
7.16.4.21 getImage()	59
7.16.4.22 getIntervalX()	59
7.16.4.23 getIntervalY()	60
7.16.4.24 getInvertYAxis()	60
7.16.4.25 getWidth()	60
7.16.4.26 getWindowDisplayScale()	60
7.16.4.27 operator=()	60
7.16.4.28 resetInterval()	60
7.16.4.29 resize()	61
7.16.4.30 saveImage()	61
7.16.4.31 setColor()	61
7.16.4.32 setInvertYAxis()	61
7.16.4.33 setNewInterval()	62
7.16.4.34 setWindowDisplayScale()	62
7.16.4.35 show()	62
7.16.4.36 waitKey()	62
7.16.4.37 waitMouseInput()	63
7.16.5 Member Data Documentation	63
7.16.5.1 m_FristShowCall	63
7.16.5.2 m_Image	63
7.16.5.3 m_IntervalChanged	63
7.16.5.4 m_IntervalX	63
7.16.5.5 m_IntervalY	63
7.16.5.6 m_InvertYAxis	63

7.16.5.7	m_MouseCallBackStorage	63
7.16.5.8	m_WindowName	64
7.16.5.9	m_WindowScale	64
7.17	cf::Window3D Struct Reference	64
7.17.1	Detailed Description	66
7.17.2	Member Enumeration Documentation	66
7.17.2.1	CameraType	66
7.17.3	Constructor & Destructor Documentation	66
7.17.3.1	Window3D()	66
7.17.3.2	~Window3D()	66
7.17.4	Member Function Documentation	66
7.17.4.1	_AdjustCamera()	66
7.17.4.2	clear()	66
7.17.4.3	disableLighting()	67
7.17.4.4	draw()	67
7.17.4.5	drawAxis()	67
7.17.4.6	drawCube()	67
7.17.4.7	drawCylinder() [1/4]	67
7.17.4.8	drawCylinder() [2/4]	68
7.17.4.9	drawCylinder() [3/4]	68
7.17.4.10	drawCylinder() [4/4]	68
7.17.4.11	drawSphere()	68
7.17.4.12	enableLighting()	69
7.17.4.13	forceDisplay()	69
7.17.4.14	getWindowHeight()	69
7.17.4.15	getWindowWidth()	69
7.17.4.16	handleKeyboardInput()	69
7.17.4.17	printWindowUsage()	69
7.17.4.18	setCamera()	69
7.17.4.19	setMaxFPS()	70

7.17.4.20 startDrawing()	70
7.17.5 Friends And Related Function Documentation	70
7.17.5.1 _DrawingFunction	70
7.17.5.2 _KeyboardCallbackFunction	70
7.17.6 Member Data Documentation	71
7.17.6.1 m_AngleAdjustment	71
7.17.6.2 m_CameraAdjustment	71
7.17.6.3 m_CameraType	71
7.17.6.4 m_DistAdjustment	71
7.17.6.5 m_FreeCamera_LookDirection	71
7.17.6.6 m_FreeCamera_position	71
7.17.6.7 m_FreeCamera_UpVector	71
7.17.6.8 m_LookAt	71
7.17.6.9 m_LookAtDistance	71
7.17.6.10 m_RotationAngle_X	72
7.17.6.11 m_RotationAngle_Y	72
7.18 cf::WindowCoordinateSystem Struct Reference	72
7.18.1 Detailed Description	73
7.18.2 Member Enumeration Documentation	73
7.18.2.1 LineType	73
7.18.3 Constructor & Destructor Documentation	73
7.18.3.1 WindowCoordinateSystem()	73
7.18.3.2 ~WindowCoordinateSystem()	74
7.18.4 Member Function Documentation	74
7.18.4.1 convert_intervalLength_to_pixelLength()	74
7.18.4.2 convert_pixelLength_to_intervalLength()	74
7.18.4.3 drawCircle()	75
7.18.4.4 drawCirclePart()	75
7.18.4.5 drawLine()	75
7.18.4.6 drawLinearEquation() [1/4]	76

7.18.4.7	drawLinearEquation() [2/4]	76
7.18.4.8	drawLinearEquation() [3/4]	77
7.18.4.9	drawLinearEquation() [4/4]	77
7.18.4.10	drawPoint()	77
7.18.4.11	setInterval()	78
7.19	cf::WindowRasterized Struct Reference	78
7.19.1	Detailed Description	79
7.19.2	Member Enumeration Documentation	79
7.19.2.1	LineType	79
7.19.3	Constructor & Destructor Documentation	79
7.19.3.1	WindowRasterized() [1/2]	79
7.19.3.2	WindowRasterized() [2/2]	79
7.19.3.3	~WindowRasterized()	80
7.20	cf::WindowVectorized Struct Reference	80
7.20.1	Detailed Description	81
7.20.2	Member Enumeration Documentation	81
7.20.2.1	LineType	81
7.20.3	Constructor & Destructor Documentation	81
7.20.3.1	WindowVectorized() [1/2]	81
7.20.3.2	WindowVectorized() [2/2]	81
7.20.3.3	~WindowVectorized()	82
7.20.4	Member Function Documentation	82
7.20.4.1	convert_intervalLength_to_pixelLength()	82
7.20.4.2	convert_pixelLength_to_intervalLength()	82
7.20.4.3	getColor_imageSpace()	83
7.20.4.4	setColor_imageSpace()	83
7.20.4.5	setInterval()	83
7.20.4.6	transformPoint_fromImage_toInterval()	84
7.20.4.7	transformPoint_fromInterval_toImage()	84

8 File Documentation	85
8.1 include/computerGeometry.hpp File Reference	85
8.1.1 Function Documentation	86
8.1.1.1 operator<<()	86
8.2 include/IFS.h File Reference	86
8.3 include/LSystem.h File Reference	86
8.4 include/ORB.h File Reference	87
8.5 include/Utils.h File Reference	87
8.5.1 Function Documentation	88
8.5.1.1 operator<<() [1/5]	88
8.5.1.2 operator<<() [2/5]	88
8.5.1.3 operator<<() [3/5]	89
8.5.1.4 operator<<() [4/5]	89
8.5.1.5 operator<<() [5/5]	89
8.6 include/window2D.h File Reference	89
8.7 include/window3D.h File Reference	90
8.8 include/windowCoordinateSystem.hpp File Reference	90
8.9 include/windowRasterized.hpp File Reference	90
8.10 include/windowVectorized.hpp File Reference	91
8.11 README.md File Reference	91
Index	93

Chapter 1

CF_CG-Lib

This library is intended to be used in 'Chaos und Fraktale' and 'Computer Geometry', lessons from 'Hochschule Darmstadt'. If you want to use it in a different 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)

Copyright © 2016 Sascha Wombacher

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

3rd party licenses:

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

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

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

cf	11
------------------------------	--------------------

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

cf::Circle	17
cf::CirclePartition	18
cf::Color	20
cf::Console	25
cf::Direction	26
cf::Interval	28
cf::IteratedFunctionSystem	30
cf::LSystem_Controller::iterator	31
cf::LindenmayerSystem	32
cf::Line	34
cf::LSystem_Controller	35
cf::Orbit	37
cf::Point	38
cf::Rect	41
cf::Vec3< POINTVECTOR >	42
cf::Window2D	50
cf::WindowCoordinateSystem	72
cf::WindowRasterized	78
cf::WindowVectorized	80
cf::Window3D	64

Chapter 4

Class Index

4.1 Class List

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

cf::Circle	The Circle struct Simple parameter wrapper struct	17
cf::CirclePartition	The CirclePartition struct Simple parameter wrapper struct	18
cf::Color	The Color struct offers a class for rgb access	20
cf::Console	The Console struct offers utility functions for 'console'	25
cf::Direction	The Direction struct for getting absolute directions from a current direction and a relative direction	26
cf::Interval	The Interval struct provides functionality to translate values from one interval into another	28
cf::IteratedFunctionSystem	The IteratedFunctionSystem class lazy people (like myself) may use the IFS typedef	30
cf::LSystem_Controller::iterator	31
cf::LindenmayerSystem	The LindenmayerSystem class lazy people (like myself) may use the IFS typedef	32
cf::Line	The Line struct Simple parameter wrapper struct	34
cf::LSystem_Controller	The LSystem_Controller struct This class enables easy iterating above a given iteration depth 35	
cf::Orbit	The Orbit class lazy people (like myself) may use the ORB typedef	37
cf::Point	The Point struct is a simple class for position access on 2D images (similar to cv::Point, but uses floats instead of integer)	38
cf::Rect	The Rect struct Simple parameter wrapper struct	41
cf::Vec3< POINTVECTOR >	The Vec3 struct General class for vector operations	42
cf::Window2D	The Window2D struct offers advanced features used by WindowRasterized/WindowVectorized	50
cf::Window3D	The Window3D struct is the default class for accessing 3D content, creating more than 1 instance results in undefined behavior	64

cf::WindowCoordinateSystem	
The WindowCoordinateSystem struct Default class for images and raster operations	72
cf::WindowRasterized	
The WindowRasterized struct Default struct for verctorized operations within a custom interval	78
cf::WindowVectorized	
The WindowVectorized struct Default class for images and raster operations	80

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

include/computerGeometry.hpp	85
include/IFS.h	86
include/LSystem.h	86
include/ORB.h	87
include/utils.h	87
include/window2D.h	89
include/window3D.h	90
include/windowCoordinateSystem.hpp	90
include/windowRasterized.hpp	90
include/windowVectorized.hpp	91

Chapter 6

Namespace Documentation

6.1 cf Namespace Reference

Classes

- struct [Circle](#)
The [Circle](#) struct Simple parameter wrapper struct.
- struct [CirclePartition](#)
The [CirclePartition](#) struct Simple parameter wrapper struct.
- 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 functionality to translate values from one interval into another.
- struct [IteratedFunctionSystem](#)
The [IteratedFunctionSystem](#) class lazy people (like myself) may use the IFS typedef.
- struct [LindenmayerSystem](#)
The [LindenmayerSystem](#) class lazy people (like myself) may use the IFS typedef.
- struct [Line](#)
The [Line](#) struct Simple parameter wrapper struct.
- struct [LSystem_Controller](#)
The [LSystem_Controller](#) struct This class enables easy iterating above a given iteration depth
.
- struct [Orbit](#)
The [Orbit](#) class lazy people (like myself) may use the ORB typedef.
- struct [Point](#)
The [Point](#) struct is a simple class for position access on 2D images (similar to cv::Point, but uses floats instead of integer)
- struct [Rect](#)
The [Rect](#) struct Simple parameter wrapper struct.
- struct [Vec3](#)
The [Vec3](#) struct General class for vector operations.
- class [Window2D](#)

The [Window2D](#) struct offers advanced features used by [WindowRasterized](#)/[WindowVectorized](#).

- 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 vectorized operations within a custom interval.

- struct [WindowVectorized](#)

The [WindowVectorized](#) struct Default class for images and raster operations.

Typedefs

- typedef [Vec3](#)< true > [PointVector](#)

[PointVector](#) Specialiazion of general [Vec3](#).

- typedef [Vec3](#)< false > [DirectionVector](#)

[DirectionVector](#) Specialiazion 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](#) Specialiazion 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

<i>str</i>	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

<i>degreeValue</i>	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

<i>radianValue</i>	Radian value to be converted
--------------------	------------------------------

Returns

Converted degree value

6.1.2.4 readAntString()

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

readAntString

Parameters

<i>filePath</i>	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

<i>filePath</i>	Read *.dat file from path
-----------------	---------------------------

Returns

6.1.2.6 readPaletteFromFile()

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

readPaletteFromFile

Parameters

<i>filePath</i>	Read *.pal file from path
-----------------	---------------------------

Returns

Chapter 7

Class Documentation

7.1 cf::Circle Struct Reference

The [Circle](#) struct Simple parameter wrapper struct.

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

Public Member Functions

- [Circle](#) (const [cf::Point](#) &Center, int Radius, int LineWidth, const [cf::Color](#) &Color)

Public Attributes

- [cf::Point](#) center
- int [radius](#)
- int [lineWidth](#)
- [cf::Color](#) color

7.1.1 Detailed Description

The [Circle](#) struct Simple parameter wrapper struct.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 Circle()

```
cf::Circle::Circle (  
    const cf::Point & Center,  
    int Radius,  
    int LineWidth,  
    const cf::Color & Color ) [inline]
```

7.1.3 Member Data Documentation

7.1.3.1 center

```
cf::Point cf::Circle::center
```

7.1.3.2 color

```
cf::Color cf::Circle::color
```

7.1.3.3 lineWidth

```
int cf::Circle::lineWidth
```

7.1.3.4 radius

```
int cf::Circle::radius
```

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

- include/window2D.h

7.2 cf::CirclePartition Struct Reference

The [CirclePartition](#) struct Simple parameter wrapper struct.

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

Public Member Functions

- [CirclePartition](#) ([cf::Point](#) Center, int Radius, float StartAngle, float EndAngle, int LineWidth, const [cf::Color](#) &Color)

Public Attributes

- [cf::Point](#) center
- int radius
- float startAngle
- float endAngle
- int lineWidth
- [cf::Color](#) color

7.2.1 Detailed Description

The [CirclePartition](#) struct Simple parameter wrapper struct.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 CirclePartition()

```
cf::CirclePartition::CirclePartition (
    cf::Point Center,
    int Radius,
    float StartAngle,
    float EndAngle,
    int LineWidth,
    const cf::Color & Color ) [inline]
```

7.2.3 Member Data Documentation

7.2.3.1 center

```
cf::Point cf::CirclePartition::center
```

7.2.3.2 color

```
cf::Color cf::CirclePartition::color
```

7.2.3.3 endAngle

```
float cf::CirclePartition::endAngle
```

7.2.3.4 lineWidth

```
int cf::CirclePartition::lineWidth
```

7.2.3.5 radius

```
int cf::CirclePartition::radius
```

7.2.3.6 startAngle

```
float cf::CirclePartition::startAngle
```

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

- [include/window2D.h](#)

7.3 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) const
- `Color operator/` (float value) const
- `Color & operator*=` (float value)
- `Color & operator/=` (float value)
- `Color operator+` (const `Color` &c) const
- `Color operator-` (const `Color` &c) const
- `Color & operator+=` (const `Color` &c)
- `Color & operator-=` (const `Color` &c)
- bool `operator==` (const `Color` &c) const
- bool `operator!=` (const `Color` &c) const
- bool `operator<` (const `Color` &c) const
- bool `operator>` (const `Color` &c) const
- bool `operator<=` (const `Color` &c) const
- bool `operator>=` (const `Color` &c) const
- `Color invert` () const

invert Invert a color, for example `cf::Color::BLACK` will be changed to `cf::Color::WHITE`

Static Public Member Functions

- static `cf::Color RandomColor` ()

RandomColor Produces a color with random red, green and blue channel.

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)

7.3.1 Detailed Description

The [Color](#) struct offers a class for rgb access.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 Color()

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

7.3.3 Member Function Documentation

7.3.3.1 invert()

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

invert Invert a color, for example [cf::Color::BLACK](#) will be changed to [cf::Color::WHITE](#)

Returns

Inverted [cf::Color](#)

7.3.3.2 operator!=(())

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

7.3.3.3 operator*()

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

7.3.3.4 operator*=(())

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

7.3.3.5 operator+()

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

7.3.3.6 operator+=()

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

7.3.3.7 operator-()

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

7.3.3.8 operator-=()

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

7.3.3.9 operator/()

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

7.3.3.10 operator/=()

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

7.3.3.11 operator<()

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

7.3.3.12 operator<=()

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

7.3.3.13 operator==()

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

7.3.3.14 operator>()

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

7.3.3.15 operator>=()

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

7.3.3.16 RandomColor()

```
static Color cf::Color::RandomColor ( ) [static]
```

RandomColor Produces a color with random red, green and blue channel.

Returns

Random [cf::Color](#)

7.3.4 Friends And Related Function Documentation

7.3.4.1 operator*

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

7.3.4.2 operator/

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

7.3.4.3 operator<<

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

7.3.5 Member Data Documentation

7.3.5.1 b

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

7.3.5.2 BLACK

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

7.3.5.3 BLUE

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

7.3.5.4 CYAN

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

7.3.5.5 g

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

7.3.5.6 GREEN

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

7.3.5.7 GREY

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

7.3.5.8 MAGENTA

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

7.3.5.9 ORANGE

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

7.3.5.10 PINK

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

7.3.5.11 r

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


7.3.5.12 RED

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

7.3.5.13 WHITE

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

7.3.5.14 YELLOW

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

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

- include/[utils.h](#)

7.4 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

7.4.1 Detailed Description

The [Console](#) struct offers utility functions for 'console'.

7.4.2 Member Function Documentation

7.4.2.1 clearConsole()

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

[clearConsole](#) Clears the console

7.4.2.2 readFloat()

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

readFloat Reads a floatingpoint value

Returns

Read value

7.4.2.3 readInt()

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

readInt Reads a integer value

Returns

Read value

7.4.2.4 readString()

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

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

Returns

Read line

7.4.2.5 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](#)

7.5 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](#) getNextDirection ([AbsoluteDirection](#) currentDirection, [RelativeDirection](#) relativeMovement)
getNextDirection receive absolute direction by providing a relative direction
- static std::string toString ([AbsoluteDirection](#) absDir)
- static std::string toString ([RelativeDirection](#) relDir)

7.5.1 Detailed Description

The [Direction](#) struct for getting absolute directions from a current direction and a relative direction.

7.5.2 Member Enumeration Documentation

7.5.2.1 AbsoluteDirection

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

Enumerator

NORTH	
EAST	
SOUTH	
WEST	
NUM_ABS_DIRS	

7.5.2.2 RelativeDirection

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

Enumerator

LEFT	
FORWARD	
RIGHT	
NUM_REL_DIRS	

7.5.3 Member Function Documentation

7.5.3.1 getNextiDirection()

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

getNextiDirection receive absolute direction by providing a relative directon

Parameters

<i>currentDirection</i>	current absolute direction
<i>relativeMovement</i>	relative movement

Returns

7.5.3.2 toString() [1/2]

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

7.5.3.3 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](#)

7.6 cf::Interval Struct Reference

The [Interval](#) struct provides functionality to translate values from one interval into another.

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

Public Member Functions

- [Interval](#) (float *_min*=0, float *_max*=0)
- float [translateIntervalPostion](#) (const [Interval](#) &newInterval, float originalPosition) const

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)

7.6.1 Detailed Description

The [Interval](#) struct provides functionality to translate values from one interval into another.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 Interval()

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

7.6.3 Member Function Documentation

7.6.3.1 translateIntervalPostion() [1/2]

```
float cf::Interval::translateIntervalPostion (
    const Interval & newInterval,
    float originalPosition ) const
```

7.6.3.2 translateIntervalPostion() [2/2]

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

7.6.4 Friends And Related Function Documentation

7.6.4.1 operator<<

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

7.6.5 Member Data Documentation

7.6.5.1 max

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

7.6.5.2 min

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

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

- [include/utils.h](#)

7.7 cf::IteratedFunctionSystem Struct Reference

The [IteratedFunctionSystem](#) class lazy people (like myself) may use the IFS typedef.

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

Public Member Functions

- void [read](#) (const std::string &filePath)
*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

7.7.1 Detailed Description

The [IteratedFunctionSystem](#) class lazy people (like myself) may use the IFS typedef.

7.7.2 Member Function Documentation

7.7.2.1 getAllTransformation()

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

7.7.2.2 getName()

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

7.7.2.3 getNumTransformations()

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

7.7.2.4 getRangeX()

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

7.7.2.5 getRangeY()

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

7.7.2.6 getTransformation()

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

7.7.2.7 read()

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

read a *.ifs file from path

Parameters

<i>fiilePath</i>	Path to a *.ifs file
------------------	----------------------

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

- include/[IFS.h](#)

7.8 cf::LSystem_Controller::iterator Struct Reference

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

Public Member Functions

- const char & [operator*](#) ()
- [iterator](#) & [operator++](#) ()
- bool [operator!=](#) (const [iterator](#) &rhs)

Friends

- struct [LSystem_Controller](#)

7.8.1 Member Function Documentation

7.8.1.1 operator!=(())

```
bool cf::LSystem_Controller::iterator::operator!=(
    const iterator & rhs )
```

7.8.1.2 operator*()

```
const char& cf::LSystem_Controller::iterator::operator* ( )
```

7.8.1.3 operator++()

```
iterator& cf::LSystem_Controller::iterator::operator++ ( )
```

7.8.2 Friends And Related Function Documentation

7.8.2.1 LSystem_Controller

```
friend struct LSystem\_Controller [friend]
```

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

- include/[LSystem.h](#)

7.9 cf::LindenmayerSystem Struct Reference

The [LindenmayerSystem](#) class lazy people (like myself) may use the IFS typedef.

```
#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 [getAdjustmentAngle](#) () const
- const std::map< char, const std::string > & [getAllProductions](#) () const

7.9.1 Detailed Description

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

7.9.2 Member Function Documentation

7.9.2.1 clearWindowEachTime()

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

7.9.2.2 getAdjustmentAngle()

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

7.9.2.3 getAllProductions()

```
const std::map<char, const std::string>& cf::LindenmayerSystem::getAllProductions ( ) const
```

7.9.2.4 getAxiom()

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

7.9.2.5 getName()

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

7.9.2.6 getNumProductions()

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

7.9.2.7 getProduction()

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

7.9.2.8 getRangeX()

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

7.9.2.9 getRangeY()

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

7.9.2.10 `getScale()`

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

7.9.2.11 `getStartAngle()`

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

7.9.2.12 `read()`

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

read a *.lin file from path

Parameters

<i>filePath</i>	Path to a *.lin file
-----------------	----------------------

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

- [include/LSystem.h](#)

7.10 `cf::Line` Struct Reference

The [Line](#) struct Simple parameter wrapper struct.

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

Public Member Functions

- [Line](#) ([cf::Point](#) Point1, [cf::Point](#) Point2, int LineWidth, const [cf::Color](#) &Color, [cf::Window2D::LineType](#) Line↔
Type=[cf::Window2D::LineType::DEFAULT](#))

Public Attributes

- [cf::Point](#) point1
- [cf::Point](#) point2
- int lineWidth
- [cf::Color](#) color
- [cf::Window2D::LineType](#) lineType

7.10.1 Detailed Description

The [Line](#) struct Simple parameter wrapper struct.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 Line()

```
cf::Line::Line (
    cf::Point Point1,
    cf::Point Point2,
    int LineWidth,
    const cf::Color & Color,
    cf::Window2D::LineType LineType = cf::Window2D::LineType::DEFAULT ) [inline]
```

7.10.3 Member Data Documentation

7.10.3.1 color

```
cf::Color cf::Line::color
```

7.10.3.2 lineType

```
cf::Window2D::LineType cf::Line::lineType
```

7.10.3.3 lineWidth

```
int cf::Line::lineWidth
```

7.10.3.4 point1

```
cf::Point cf::Line::point1
```

7.10.3.5 point2

```
cf::Point cf::Line::point2
```

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

- include/[window2D.h](#)

7.11 cf::LSystem_Controller Struct Reference

The [LSystem_Controller](#) struct This class enables easy iterating above a given iteration depth

.

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

Classes

- struct [iterator](#)

Public Member Functions

- [LSystem_Controller](#) (size_t depth, const [LSystem](#) &[LSystem](#))
- [iterator](#) begin ()
- [iterator](#) end ()

7.11.1 Detailed Description

The [LSystem_Controller](#) struct This class enables easy iterating above a given iteration depth
.

usage:

```
LSystem_Controller myController(<depth>, <lsystem>);  
for (char c : myController)  
    std::cout << c;
```

7.11.2 Constructor & Destructor Documentation

7.11.2.1 LSystem_Controller()

```
cf::LSystem_Controller::LSystem_Controller (  
    size_t depth,  
    const LSystem & LSystem )
```

7.11.3 Member Function Documentation

7.11.3.1 begin()

```
iterator cf::LSystem_Controller::begin ( )
```

7.11.3.2 end()

```
iterator cf::LSystem_Controller::end ( )
```

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

- include/[LSystem.h](#)

7.12 cf::Orbit Struct Reference

The [Orbit](#) class lazy people (like myself) may use the ORB typedef.

```
#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

7.12.1 Detailed Description

The [Orbit](#) class lazy people (like myself) may use the ORB typedef.

7.12.2 Member Function Documentation

7.12.2.1 [getAllFactors\(\)](#)

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

7.12.2.2 [getAllStartingPoints\(\)](#)

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

7.12.2.3 [getName\(\)](#)

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

7.12.2.4 [getNumFactors\(\)](#)

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

7.12.2.5 [getNumStartingPoints\(\)](#)

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

7.12.2.6 `getRangeX()`

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

7.12.2.7 `getRangeY()`

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

7.12.2.8 `read()`

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

read a *.orb file from path

Parameters

<i>filePath</i>	Path to a *.orb file
-----------------	----------------------

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

- include/[ORB.h](#)

7.13 `cf::Point` Struct Reference

The [Point](#) struct is a simple class for position 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)

7.13.1 Detailed Description

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

7.13.2 Constructor & Destructor Documentation

7.13.2.1 Point()

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

7.13.3 Member Function Documentation

7.13.3.1 operator cv::Point()

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

7.13.3.2 operator!=(())

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

7.13.3.3 operator*()

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

7.13.3.4 operator*=(())

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

7.13.3.5 operator+()

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

7.13.3.6 operator+=()

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

7.13.3.7 operator-()

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

7.13.3.8 operator-=()

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

7.13.3.9 operator/()

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

7.13.3.10 operator/=()

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

7.13.3.11 operator==()

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

7.13.4 Friends And Related Function Documentation

7.13.4.1 operator*

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


7.13.4.2 operator/

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

7.13.5 Member Data Documentation

7.13.5.1 x

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

7.13.5.2 y

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

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

- include/window2D.h

7.14 cf::Rect Struct Reference

The [Rect](#) struct Simple parameter wrapper struct.

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

Public Member Functions

- [Rect](#) ([cf::Point](#) Point1, [cf::Point](#) Point2, int LineWidth, const [cf::Color](#) &[Color](#))

Public Attributes

- [cf::Point](#) point1
- [cf::Point](#) point2
- int lineWidth
- [cf::Color](#) color

7.14.1 Detailed Description

The [Rect](#) struct Simple parameter wrapper struct.

7.14.2 Constructor & Destructor Documentation

7.14.2.1 Rect()

```
cf::Rect::Rect (
    cf::Point Point1,
    cf::Point Point2,
    int LineWidth,
    const cf::Color & Color ) [inline]
```

7.14.3 Member Data Documentation

7.14.3.1 color

```
cf::Color cf::Rect::color
```

7.14.3.2 lineWidth

```
int cf::Rect::lineWidth
```

7.14.3.3 point1

```
cf::Point cf::Rect::point1
```

7.14.3.4 point2

```
cf::Point cf::Rect::point2
```

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

- [include/window2D.h](#)

7.15 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 Multiplies 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 >](#) &)

7.15.1 Detailed Description

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

The [Vec3](#) struct General class for vector operations.

it provides:

- 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')

7.15.2 Constructor & Destructor Documentation

7.15.2.1 [Vec3\(\)](#) [1/3]

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

7.15.2.2 [Vec3\(\)](#) [2/3]

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

7.15.2.3 [Vec3\(\)](#) [3/3]

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

7.15.3 Member Function Documentation

7.15.3.1 getW()

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

getW Read access to component 'w'

Returns

7.15.3.2 getX()

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

getX Read access to component 'x'

Returns

7.15.3.3 getY()

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

getY Read access to component 'y'

Returns

7.15.3.4 isPointVector()

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

isPointVector Checks wether a Vector is a PointVector or DirectionVector

Returns

7.15.3.5 normalize()

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

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

7.15.3.6 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

7.15.3.7 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

7.15.3.8 operator const glm::vec3 &()

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

7.15.3.9 operator glm::vec3()

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

7.15.3.10 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

<i>rhs</i>	Second operand for cross product
------------	----------------------------------

Returns

7.15.3.11 operator%=()

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

7.15.3.12 operator*() [1/2]

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

operator* Multiplies each component of the vector with a factor

Parameters

<i>rhs</i>	Factor for the multiplication
------------	-------------------------------

Returns

Multiplied vector

7.15.3.13 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

<i>rhs</i>	Second operand for dot product
------------	--------------------------------

Returns

7.15.3.14 operator*=()

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

7.15.3.15 operator+()

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

7.15.3.16 operator+=()

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

7.15.3.17 operator-()

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

7.15.3.18 operator-=()

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

7.15.3.19 operator=() [1/2]

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

7.15.3.20 operator=() [2/2]

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

7.15.3.21 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

<i>idx</i>	Access index
------------	--------------

Returns

7.15.3.22 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

<i>idx</i>	Access index, idx = 0 -> x, idx = 1 -> y, idx = 2 -> w
------------	--

Returns

7.15.3.23 setW()

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

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

Parameters

<i>value</i>	
--------------	--

7.15.3.24 setX()

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

setX Write to component 'x'

Parameters

<i>value</i>	
--------------	--

7.15.3.25 setY()

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

setY Write to component 'y'

Parameters

<i>value</i>	
--------------	--

7.15.4 Friends And Related Function Documentation

7.15.4.1 operator*

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

7.15.4.2 operator<<)

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

7.15.4.3 Vec3<"!POINTVECTOR >

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

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

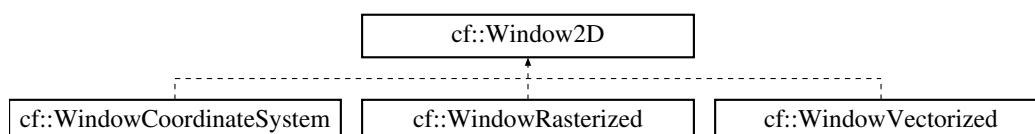
- include/[computerGeometry.hpp](#)

7.16 cf::Window2D Class Reference

The [Window2D](#) struct offers advanced features used by WindowRasterized/WindowVectorized.

```
#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 waitKey(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 width - 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 Access to underlying image width
- int [getHeight](#) () const
getHeight Access 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 [drawCirclePart](#) ([cf::Point](#) center, int radius, float startAngle, float endAngle, int lineWidth, const [cf::Color](#) &color)
drawCirclePart Draws a part of a circle
- void [floodFill](#) ([cf::Point](#) startingPoint, const [cf::Color](#) &color)
floodFill Fills an area
- void [drawLine](#) (const [cf::Line](#) &line)
drawLine Draws a line from line class
- void [drawRectangle](#) (const [cf::Rect](#) &rect)
drawRectangle Draws a rect from rect class
- void [drawCircle](#) (const [cf::Circle](#) &circle)
drawCircle Draws a circle from circle class
- void [drawCirclePart](#) (const [cf::CirclePartition](#) &circlePartition)
drawCirclePart Draws a circlePartition from circlePartition class
- [Window2D](#) & [operator=](#) (const [Window2D](#) &rhs)
operator= Copy assignment operator

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

7.16.1 Detailed Description

The [Window2D](#) struct offers advanced features used by WindowRasterized/WindowVertorized.

7.16.2 Member Enumeration Documentation

7.16.2.1 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↔ _2	

7.16.3 Constructor & Destructor Documentation

7.16.3.1 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} )
```

7.16.3.2 Window2D() [2/2]

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

7.16.3.3 ~Window2D()

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

7.16.4 Member Function Documentation

7.16.4.1 `_convertFromNewInterval()`

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

7.16.4.2 `_convertToNewInterval()`

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

7.16.4.3 `_correctYValue()`

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

7.16.4.4 `_window2foreground()`

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

7.16.4.5 `clear()`

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

7.16.4.6 `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

<i>color</i>	Axis color, default is white
<i>stepSize</i> ↔ <i>_x</i>	Dynamially set step size (x-axis), negative numbers indicate 10 steps for interval x
<i>stepSize</i> ↔ <i>_y</i>	Dynamially set step size (y-axis), negative numbers indicate 10 steps for interval y

7.16.4.7 drawCircle() [1/2]

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

drawCircle Draws a circle around the center

Parameters

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

7.16.4.8 drawCircle() [2/2]

```
void cf::Window2D::drawCircle (
    const cf::Circle & circle )
```

drawCircle Draws a circle from circle class

Parameters

<i>circle</i>	
---------------	--

7.16.4.9 drawCirclePart() [1/2]

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

drawCirclePart Draws a part of a circle

Parameters

<i>center</i>	Center point of the circle
<i>radius</i>	Radius of the circle
<i>startAngle</i>	Start position (in degrees)
<i>endAngle</i>	End position (in degrees)
<i>color</i>	Color of the drawn line

7.16.4.10 drawCirclePart() [2/2]

```
void cf::Window2D::drawCirclePart (
    const cf::CirclePartition & circlePartition )
```

drawCirclePart Draws a circlePartition from circlePartition class

Parameters

<i>circlePartition</i>	
------------------------	--

7.16.4.11 drawLine() [1/2]

```
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

<i>point1</i>	Point within interval_x and interval_y
<i>point2</i>	Point within interval_x and interval_y
<i>lineWidth</i>	Line width in pixel size
<i>color</i>	Line color

7.16.4.12 drawLine() [2/2]

```
void cf::Window2D::drawLine (
    const cf::Line & line )
```

drawLine Draws a line from line class

Parameters

<i>line</i>	
-------------	--

7.16.4.13 drawRectangle() [1/2]

```
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

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

7.16.4.14 drawRectangle() [2/2]

```
void cf::Window2D::drawRectangle (
    const cf::Rect & rect )
```

drawRectangle Draws a rect from rect class

Parameters

<i>rect</i>	
-------------	--

7.16.4.15 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

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

7.16.4.16 flippHorizontal()

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

flippHorizontal Flipp image horizontally

7.16.4.17 flippVertical()

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

flippHorizontal Flipp image vertically

7.16.4.18 floodFill()

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

floodFill Fills an area

Parameters

<i>startingPoint</i>	First point to be colored
<i>color</i>	Fill color

7.16.4.19 getColor()

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

7.16.4.20 getHeight()

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

getHeight Access to underlying image height

Returns

Height

7.16.4.21 getImage()

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

getImage Direct access to the underlying image

Returns

Image handle

7.16.4.22 getIntervalX()

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

getIntervalX Const access to interval in x direction

Returns

7.16.4.23 getIntervalY()

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

getIntervalY Const access to interval in y direction

Returns**7.16.4.24 getInvertYAxis()**

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

7.16.4.25 getWidth()

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

getWidth Access to underlying image width

Returns

Width

7.16.4.26 getWindowDisplayScale()

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

7.16.4.27 operator=()

```
Window2D& cf::Window2D::operator= (
    const Window2D & rhs )
```

operator= Copy assignment operator

Parameters

<i>rhs</i>	Element to be copied
------------	----------------------

Returns**7.16.4.28 resetInterval()**

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

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

7.16.4.29 resize()

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

resize Resize underlying image

Parameters

<i>pixelWidth</i>	New width
<i>pixelHeight</i>	New height

7.16.4.30 saveImage()

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

saveImage Saves current image to harddrive

Parameters

<i>filePath</i>	File path and name, format will be determined based on file ending (*.png, *.jpeg, ...)
-----------------	---

7.16.4.31 setColor()

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

7.16.4.32 setInvertYAxis()

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

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

Parameters

<i>invert</i>	
---------------	--

7.16.4.33 setNewInterval()

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

setNewInterval Set new interval

Parameters

<i>intervalX</i>	Interval in x direction
<i>intervalY</i>	Interval in y direction

7.16.4.34 setWindowDisplayScale()

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

setWindowDisplayScale Scales the image before displaying

Parameters

<i>scale</i>	Window scale size
--------------	-------------------

7.16.4.35 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 wait↵Key(1000))

7.16.4.36 waitKey()

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

waitKey Block access until key input on window

Parameters

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

Returns

7.16.4.37 waitMouseInput()

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

waitMouseInput Blocks until mouse input has been given

Parameters

<i>x</i>	X-Window position
<i>y</i>	Y-Window position

7.16.5 Member Data Documentation**7.16.5.1 m_FristShowCall**

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

7.16.5.2 m_Image

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

7.16.5.3 m_IntervalChanged

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

7.16.5.4 m_IntervalX

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

7.16.5.5 m_IntervalY

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

7.16.5.6 m_InvertYAxis

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

7.16.5.7 m_MouseCallBackStorage

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

7.16.5.8 m_WindowName

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

7.16.5.9 m_WindowScale

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

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

- include/[window2D.h](#)

7.17 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::STATIC_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::WHITE](#))
- 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 cylinder*
- 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 [drawSphere](#) (const glm::vec3 &position, float diameter=1.f, const [Color](#) &color=[Color::WHITE](#)) const
drawSphere Draws a solid Sphere
- void [drawCube](#) (const glm::vec3 &position, float size=1.f, const [Color](#) &color=[Color::WHITE](#)) const
drawCube Draws a solid Cube
- 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](#) ()

7.17.1 Detailed Description

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

7.17.2 Member Enumeration Documentation

7.17.2.1 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	

7.17.3 Constructor & Destructor Documentation

7.17.3.1 Window3D()

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

7.17.3.2 ~Window3D()

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

7.17.4 Member Function Documentation

7.17.4.1 _AdjustCamera()

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

7.17.4.2 clear()

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

7.17.4.3 disableLighting()

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

disableLighting Disable lightning (Default: lightning is enabled)

7.17.4.4 draw()

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

draw Draw function, this has to be implemented

7.17.4.5 drawAxis()

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

drawAxis Draw x-,y- and z-axis

Parameters

<i>length</i>	Axis length
---------------	-------------

7.17.4.6 drawCube()

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

drawCube Draws a solid Cube

Parameters

<i>position</i>	Midpoint position
<i>size</i>	Cube size
<i>color</i>	Cube color

7.17.4.7 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 cylinder

Parameters

<i>drawingDirection</i>	Cylinder direction
<i>position</i>	Start position
<i>diameter</i>	Cylinder diamenter
<i>color</i>	Cylinder color

7.17.4.8 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](#).

7.17.4.9 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](#).

7.17.4.10 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](#).

7.17.4.11 drawSphere()

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

drawSphere Draws a solid Sphere

Parameters

<i>position</i>	Midpoint position
<i>diameter</i>	Sphere diamenter
<i>color</i>	Sphere color

7.17.4.12 enableLighting()

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

enableLighting Enable lightning (Default: lightning is enabled)

7.17.4.13 forceDisplay()

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

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

7.17.4.14 getWindowHeight()

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

7.17.4.15 getWindowWidth()

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

7.17.4.16 handleKeyboardInput()

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

handleKeyboardInput Access key input by simple override this function

Parameters

<i>key</i>	Key pressed
<i>x</i>	Mouse-x-position of the key press event
<i>y</i>	Mouse-y-position of the key press event

7.17.4.17 printWindowUsage()

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

printWindowUsage Print camera usage to console

7.17.4.18 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

<i>type</i>	Camera type
<i>lookAt</i>	
<i>distance</i>	

7.17.4.19 setMaxFPS()

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

setMaxFPS Set maximum frames per second

Parameters

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

7.17.4.20 startDrawing()

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

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

fistClearColor Fist clear color (clear in 'draw' function might be ignored the first time)

Returns

7.17.5 Friends And Related Function Documentation

7.17.5.1 _DrawingFunction

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

7.17.5.2 _KeyboardCallbackFunction

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

7.17.6 Member Data Documentation

7.17.6.1 m_AngleAdjustment

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

7.17.6.2 m_CameraAdjustment

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

7.17.6.3 m_CameraType

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

7.17.6.4 m_DistAdjustment

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

7.17.6.5 m_FreeCamera_LookDirection

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

7.17.6.6 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.

7.17.6.7 m_FreeCamera_UpVector

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

7.17.6.8 m_LookAt

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

7.17.6.9 m_LookAtDistance

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

7.17.6.10 m_RotationAngle_X

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

7.17.6.11 m_RotationAngle_Y

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

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

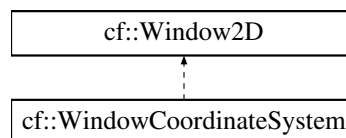
- include/[window3D.h](#)

7.18 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 coefficient 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 $y = m \cdot x + t$
- void [drawCircle](#) (const [cf::Point](#) ¢er, 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 [drawCirclePart](#) (const [cf::Point](#) ¢er, float radius, float startAngle, float endAngle, const [cf::Color](#) &color=[cf::Color::BLACK](#), int lineWidth=1)
drawCirclePart Draw a partition of a circle

Additional Inherited Members

7.18.1 Detailed Description

The [WindowCoordinateSystem](#) struct Default class for images and raster operations.

7.18.2 Member Enumeration Documentation

7.18.2.1 LineType

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

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

7.18.3 Constructor & Destructor Documentation

7.18.3.1 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

<i>range</i> ↔ _x	Interval in x direction
<i>range</i> ↔ _y	Interval in y direction
<i>width</i>	Image width in pixel (hight will be determind automatically)

7.18.3.2 ~WindowCoordinateSystem()

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

7.18.4 Member Function Documentation

7.18.4.1 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

<i>intervalLength</i>	
-----------------------	--

Returns

7.18.4.2 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

<i>pixelLength</i>	
--------------------	--

Returns

7.18.4.3 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

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

7.18.4.4 drawCirclePart()

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

drawCirclePart Draw a partition of a circle

Parameters

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

7.18.4.5 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

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

7.18.4.6 `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

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

7.18.4.7 `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

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

7.18.4.8 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 coefficient vector

Parameters

<i>vec</i>	Vector of coefficients a b and c
<i>color</i>	Line color
<i>type</i>	Change line type to dot/dash/dot-dash
<i>lineWidth</i>	Width of the line, Note: only available on default line type

7.18.4.9 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 $y = m \cdot x + t$

Parameters

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

7.18.4.10 drawPoint()

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

drawPoint Draws a cross-shaped point

Parameters

<i>pos</i>	Cross position
<i>color</i>	Cross color

7.18.4.11 setInterval()

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

setInterval Set new interval

Parameters

<i>range</i> ↔ _x	Interval in x direction
<i>range</i> ↔ _y	Interval in y direction
<i>width</i>	Image width in pixel (hight will be determind automatically)

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

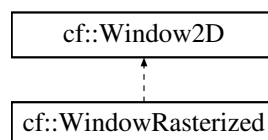
- include/[windowCoordinateSystem.hpp](#)

7.19 cf::WindowRasterized Struct Reference

The [WindowRasterized](#) struct Default struct for vectorized 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

7.19.1 Detailed Description

The [WindowRasterized](#) struct Default struct for vectorized operations within a custom interval.

7.19.2 Member Enumeration Documentation

7.19.2.1 LineType

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

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

7.19.3 Constructor & Destructor Documentation

7.19.3.1 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

<i>width</i>	Pixel width of the image
<i>height</i>	Pixel height of the image
<i>windowName</i>	Name of the window
<i>startColor</i>	Background color

7.19.3.2 WindowRasterized() [2/2]

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

[WindowRasterized](#) Load image from file path.

Parameters

<i>filePath</i>	Path to file
-----------------	--------------

7.19.3.3 ~WindowRasterized()

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

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

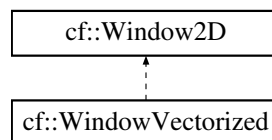
- include/[windowRasterized.hpp](#)

7.20 cf::WindowVectorized Struct Reference

The [WindowVectorized](#) struct Default class for images and raster operations.

```
#include <windowVectorized.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 *windowName="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 constructor.
- 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 x, int y) const
getColor_imageSpace Get color from image x/y position
- void [setColor_imageSpace](#) (int x, int y, const [cf::Color](#) &color)
setColor_imageSpace Set color from image x/y position

Additional Inherited Members

7.20.1 Detailed Description

The [WindowVectorized](#) struct Default class for images and raster operations.

7.20.2 Member Enumeration Documentation

7.20.2.1 LineType

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

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

7.20.3 Constructor & Destructor Documentation

7.20.3.1 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

<i>width</i>	Image width in pixel (hight will be determind automatically)
<i>range</i> _↔ <i>_x</i>	Interval in x direction
<i>range</i> _↔ <i>_y</i>	Interval in y direction

7.20.3.2 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

<i>filePath</i>	Path to image file
-----------------	--------------------

Parameters

<i>width</i>	Image width, Note: height will be calculated based on ranges and width
<i>range</i> ↔ <i>_x</i>	Interval in x direction
<i>range</i> ↔ <i>_y</i>	Interval in y direction

7.20.3.3 ~WindowVectorized()

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

7.20.4 Member Function Documentation

7.20.4.1 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

<i>intervalLength</i>	Length to be converted to pixel length
-----------------------	--

Returns

7.20.4.2 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

<i>pixelLength</i>	Length to be converted to the intervall length
--------------------	--

Returns

7.20.4.3 getColor_imageSpace()

```
cf::Color cf::WindowVectorized::getColor_imageSpace (
    int x,
    int y ) const [inline]
```

getColor_imageSpace Get color from image x/y position

Parameters

<i>x</i>	X position
<i>y</i>	Y position

Returns

7.20.4.4 setColor_imageSpace()

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

setColor_imageSpace Set color from image x/y position

Parameters

<i>x</i>	X position
<i>y</i>	Y position
<i>color</i>	Color to be set

7.20.4.5 setInterval()

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

setInterval Set new interval

Parameters

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

7.20.4.6 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

<i>point</i>	Point to be transformed
--------------	---

Returns

Transformed point

7.20.4.7 transformPoint_fromInterval_tolmage()

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

transformPoint_fromInterval_tolmage Transform point from interval position to pixel position

Parameters

<i>point</i>	Point to be transformed
--------------	---

Returns

Transformed point

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

- [include/windowVectorized.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

<i>os</i>	Outputstream, e.g. std::cout
<i>rhs</i>	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 typedef.

Namespaces

- [cf](#)

Typedefs

- typedef IteratedFunctionSystem [cf::IFS](#)

8.3 include/LSystem.h File Reference

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

Classes

- struct [cf::LindenmayerSystem](#)
The *LindenmayerSystem* class lazy people (like myself) may use the IFS tyepdef.
- struct [cf::LSystem_Controller](#)
The *LSystem_Controller* struct This class enables easy iterating above a given iteration depth
- struct [cf::LSystem_Controller::iterator](#)

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 functionality 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/WindowVectorized.
- struct [cf::Point](#)
The [Point](#) struct is a simple class for position access on 2D images (similar to cv::Point, but uses floats instead of integer)
- struct [cf::Line](#)
The [Line](#) struct Simple parameter wrapper struct.
- struct [cf::Rect](#)
The [Rect](#) struct Simple parameter wrapper struct.
- struct [cf::Circle](#)
The [Circle](#) struct Simple parameter wrapper struct.
- struct [cf::CirclePartition](#)
The [CirclePartition](#) struct Simple parameter wrapper struct.

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 vectorized operations within a custom interval.

Namespaces

- [cf](#)

8.10 include/windowVectorized.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

- _AdjustCamera
 - cf::Window3D, 66
 - _DrawingFunction
 - cf::Window3D, 70
 - _KeyboardCallbackFunction
 - cf::Window3D, 70
 - _convertFromNewInterval
 - cf::Window2D, 54
 - _convertToNewInterval
 - cf::Window2D, 54
 - _correctYValue
 - cf::Window2D, 54
 - _removeWindowsSpecificCarriageReturn
 - cf, 13
 - _window2foreground
 - cf::Window2D, 54
 - ~Window2D
 - cf::Window2D, 53
 - ~Window3D
 - cf::Window3D, 66
 - ~WindowCoordinateSystem
 - cf::WindowCoordinateSystem, 74
 - ~WindowRasterized
 - cf::WindowRasterized, 79
 - ~WindowVectorized
 - cf::WindowVectorized, 82
- AbsoluteDirection
 - cf::Direction, 27
- b
 - cf::Color, 23
- BLACK
 - cf::Color, 23
- BLUE
 - cf::Color, 24
- begin
 - cf::LSystem_Controller, 36
- CYAN
 - cf::Color, 24
- CameraType
 - cf::Window3D, 66
- center
 - cf::Circle, 18
 - cf::CirclePartition, 19
- cf, 11
 - _removeWindowsSpecificCarriageReturn, 13
 - degree2radian, 13
 - DirectionVector, 12
 - IFS, 12
 - LSystem, 12
 - ORB, 13
 - PointVector, 13
 - radian2degree, 13
 - readAntString, 14
 - readDATFile, 14
 - readPaletteFromFile, 14
- cf::Circle, 17
 - center, 18
 - Circle, 17
 - color, 18
 - lineWidth, 18
 - radius, 18
- cf::CirclePartition, 18
 - center, 19
 - CirclePartition, 19
 - color, 19
 - endAngle, 19
 - lineWidth, 19
 - radius, 19
 - startAngle, 19
- cf::Color, 20
 - b, 23
 - BLACK, 23
 - BLUE, 24
 - CYAN, 24
 - Color, 21
 - g, 24
 - GREEN, 24
 - GREY, 24
 - invert, 21
 - MAGENTA, 24
 - ORANGE, 24
 - operator!=, 21
 - operator<, 22
 - operator<<, 23
 - operator<=, 22
 - operator>, 22
 - operator>=, 23
 - operator*, 21, 23
 - operator*==, 21
 - operator+, 21
 - operator+==, 22
 - operator-, 22
 - operator-=, 22
 - operator/, 22, 23
 - operator/=, 22
 - operator==, 22

- PINK, 24
- r, 24
- RED, 24
- RandomColor, 23
- WHITE, 25
- YELLOW, 25
- cf::Console, 25
 - clearConsole, 25
 - readFloat, 25
 - readInt, 26
 - readString, 26
 - waitKey, 26
- cf::Direction, 26
 - AbsoluteDirection, 27
 - getNextiDirection, 28
 - RelativeDirection, 27
 - toString, 28
- cf::Interval, 28
 - Interval, 29
 - max, 30
 - min, 30
 - operator<<, 29
 - translateIntervalPostion, 29
- cf::IteratedFunctionSystem, 30
 - getAllTransformation, 30
 - getName, 30
 - getNumTransformations, 30
 - getRangeX, 31
 - getRangeY, 31
 - getTransformation, 31
 - read, 31
- cf::LSystem_Controller, 35
 - begin, 36
 - end, 36
 - LSystem_Controller, 36
- cf::LSystem_Controller::iterator, 31
 - LSystem_Controller, 32
 - operator!=, 32
 - operator*, 32
 - operator++, 32
- cf::LindenmayerSystem, 32
 - clearWindowEachTime, 33
 - getAdjustmentAngle, 33
 - getAllProductions, 33
 - getAxiom, 33
 - getName, 33
 - getNumProductions, 33
 - getProduction, 33
 - getRangeX, 33
 - getRangeY, 33
 - getScale, 33
 - getStartAngle, 34
 - read, 34
- cf::Line, 34
 - color, 35
 - Line, 35
 - lineType, 35
 - lineWidth, 35
 - point1, 35
 - point2, 35
- cf::Orbit, 37
 - getAllFactors, 37
 - getAllStartingPoints, 37
 - getName, 37
 - getNumFactors, 37
 - getNumStartingPoints, 37
 - getRangeX, 37
 - getRangeY, 38
 - read, 38
- cf::Point, 38
 - operator cv::Point, 39
 - operator!=, 39
 - operator*, 39, 40
 - operator*==, 39
 - operator+, 39
 - operator+==, 40
 - operator-, 40
 - operator-=, 40
 - operator/, 40
 - operator/=, 40
 - operator==, 40
 - Point, 39
 - x, 41
 - y, 41
- cf::Rect, 41
 - color, 42
 - lineWidth, 42
 - point1, 42
 - point2, 42
 - Rect, 42
- cf::Vec3
 - getW, 45
 - getX, 45
 - getY, 45
 - isPointVector, 45
 - normalize, 45
 - operator cf::Point, 46
 - operator cf::Vec3< false >, 46
 - operator const glm::vec3 &, 46
 - operator glm::vec3, 46
 - operator<<(), 50
 - operator*, 47, 50
 - operator*==, 47
 - operator+, 47
 - operator+==, 48
 - operator-, 48
 - operator-=, 48
 - operator=, 48
 - operator%, 46
 - operator%==, 47
 - operator[], 48, 49
 - setW, 49
 - setX, 49
 - setY, 50
 - Vec3, 44
 - Vec3<!POINTVECTOR >, 50

- cf::Vec3< POINTVECTOR >, 42
- cf::Window2D, 50
 - _convertFromNewInterval, 54
 - _convertToNewInterval, 54
 - _correctYValue, 54
 - _window2foreground, 54
 - ~Window2D, 53
 - clear, 54
 - drawAxis, 54
 - drawCircle, 54, 55
 - drawCirclePart, 55
 - drawLine, 56
 - drawRectangle, 56, 58
 - drawSpecializedLine, 58
 - flippHorizontal, 58
 - flippVertical, 58
 - floodFill, 58
 - getColor, 59
 - getHeight, 59
 - getImage, 59
 - getIntervalX, 59
 - getIntervalY, 59
 - getInvertYAxis, 60
 - getWidth, 60
 - getWindowDisplayScale, 60
 - LineType, 53
 - m_FristShowCall, 63
 - m_Image, 63
 - m_IntervalChanged, 63
 - m_IntervalX, 63
 - m_IntervalY, 63
 - m_InvertYAxis, 63
 - m_MouseCallbackStorage, 63
 - m_WindowName, 63
 - m_WindowScale, 64
 - operator=, 60
 - resetInterval, 60
 - resize, 61
 - saveImage, 61
 - setColor, 61
 - setInvertYAxis, 61
 - setNewInterval, 61
 - setWindowDisplayScale, 62
 - show, 62
 - waitKey, 62
 - waitMouseInput, 62
 - Window2D, 53
- cf::Window3D, 64
 - _AdjustCamera, 66
 - _DrawingFunction, 70
 - _KeyboardCallbackFunction, 70
 - ~Window3D, 66
 - CameraType, 66
 - clear, 66
 - disableLighting, 66
 - draw, 67
 - drawAxis, 67
 - drawCube, 67
 - drawCylinder, 67, 68
 - drawSphere, 68
 - enableLighting, 69
 - forceDisplay, 69
 - getWindowHeight, 69
 - getWindowWidth, 69
 - handleKeyboardInput, 69
 - m_AngleAdjustment, 71
 - m_CameraAdjustment, 71
 - m_CameraType, 71
 - m_DistAdjustment, 71
 - m_FreeCamera_LookDirection, 71
 - m_FreeCamera_UpVector, 71
 - m_FreeCamera_position, 71
 - m_LookAt, 71
 - m_LookAtDistance, 71
 - m_RotationAngle_X, 71
 - m_RotationAngle_Y, 72
 - printWindowUsage, 69
 - setCamera, 69
 - setMaxFPS, 70
 - startDrawing, 70
 - Window3D, 66
- cf::WindowCoordinateSystem, 72
 - ~WindowCoordinateSystem, 74
 - convert_intervalLength_to_pixelLength, 74
 - convert_pixelLength_to_intervalLength, 74
 - drawCircle, 74
 - drawCirclePart, 75
 - drawLine, 75
 - drawLinearEquation, 76, 77
 - drawPoint, 77
 - LineType, 73
 - setInterval, 77
 - WindowCoordinateSystem, 73
- cf::WindowRasterized, 78
 - ~WindowRasterized, 79
 - LineType, 79
 - WindowRasterized, 79
- cf::WindowVectorized, 80
 - ~WindowVectorized, 82
 - convert_intervalLength_to_pixelLength, 82
 - convert_pixelLength_to_intervalLength, 82
 - getColor_imageSpace, 82
 - LineType, 81
 - setColor_imageSpace, 83
 - setInterval, 83
 - transformPoint_fromImage_toInterval, 83
 - transformPoint_fromInterval_toImage, 84
 - WindowVectorized, 81
- Circle
 - cf::Circle, 17
- CirclePartition
 - cf::CirclePartition, 19
- clear
 - cf::Window2D, 54
 - cf::Window3D, 66
- clearConsole

- cf::Console, 25
- clearWindowEachTime
 - cf::LindenmayerSystem, 33
- Color
 - cf::Color, 21
- color
 - cf::Circle, 18
 - cf::CirclePartition, 19
 - cf::Line, 35
 - cf::Rect, 42
- computerGeometry.hpp
 - operator<, 86
- convert_intervalLength_to_pixelLength
 - cf::WindowCoordinateSystem, 74
 - cf::WindowVectorized, 82
- convert_pixelLength_to_intervalLength
 - cf::WindowCoordinateSystem, 74
 - cf::WindowVectorized, 82
- degree2radian
 - cf, 13
- DirectionVector
 - cf, 12
- disableLighting
 - cf::Window3D, 66
- draw
 - cf::Window3D, 67
- drawAxis
 - cf::Window2D, 54
 - cf::Window3D, 67
- drawCircle
 - cf::Window2D, 54, 55
 - cf::WindowCoordinateSystem, 74
- drawCirclePart
 - cf::Window2D, 55
 - cf::WindowCoordinateSystem, 75
- drawCube
 - cf::Window3D, 67
- drawCylinder
 - cf::Window3D, 67, 68
- drawLine
 - cf::Window2D, 56
 - cf::WindowCoordinateSystem, 75
- drawLinearEquation
 - cf::WindowCoordinateSystem, 76, 77
- drawPoint
 - cf::WindowCoordinateSystem, 77
- drawRectangle
 - cf::Window2D, 56, 58
- drawSpecializedLine
 - cf::Window2D, 58
- drawSphere
 - cf::Window3D, 68
- enableLighting
 - cf::Window3D, 69
- end
 - cf::LSystem_Controller, 36
- endAngle
 - cf::CirclePartition, 19
- flippHorizontal
 - cf::Window2D, 58
- flippVertical
 - cf::Window2D, 58
- floodFill
 - cf::Window2D, 58
- forceDisplay
 - cf::Window3D, 69
- g
 - cf::Color, 24
- GREEN
 - cf::Color, 24
- GREY
 - cf::Color, 24
- getAdjustmentAngle
 - cf::LindenmayerSystem, 33
- getAllFactors
 - cf::Orbit, 37
- getAllProductions
 - cf::LindenmayerSystem, 33
- getAllStartingPoints
 - cf::Orbit, 37
- getAllTransformation
 - cf::IteratedFunctionSystem, 30
- getAxiom
 - cf::LindenmayerSystem, 33
- getColor
 - cf::Window2D, 59
- getColor_imageSpace
 - cf::WindowVectorized, 82
- getHeight
 - cf::Window2D, 59
- getImage
 - cf::Window2D, 59
- getIntervalX
 - cf::Window2D, 59
- getIntervalY
 - cf::Window2D, 59
- getInvertYAxis
 - cf::Window2D, 60
- getName
 - cf::IteratedFunctionSystem, 30
 - cf::LindenmayerSystem, 33
 - cf::Orbit, 37
- getNextDirection
 - cf::Direction, 28
- getNumFactors
 - cf::Orbit, 37
- getNumProductions
 - cf::LindenmayerSystem, 33
- getNumStartingPoints
 - cf::Orbit, 37
- getNumTransformations
 - cf::IteratedFunctionSystem, 30
- getProduction
 - cf::LindenmayerSystem, 33

- getRangeX
 - cf::IteratedFunctionSystem, 31
 - cf::LindenmayerSystem, 33
 - cf::Orbit, 37
- getRangeY
 - cf::IteratedFunctionSystem, 31
 - cf::LindenmayerSystem, 33
 - cf::Orbit, 38
- getScale
 - cf::LindenmayerSystem, 33
- getStartAngle
 - cf::LindenmayerSystem, 34
- getTransformation
 - cf::IteratedFunctionSystem, 31
- getWidth
 - cf::Window2D, 60
- getWindowDisplayScale
 - cf::Window2D, 60
- getWindowHeight
 - cf::Window3D, 69
- getWindowWidth
 - cf::Window3D, 69
- getW
 - cf::Vec3, 45
- getX
 - cf::Vec3, 45
- getY
 - cf::Vec3, 45
- handleKeyboardInput
 - cf::Window3D, 69
- IFS
 - cf, 12
- include/IFS.h, 86
- include/LSystem.h, 86
- include/ORB.h, 87
- include/computerGeometry.hpp, 85
- include/Utils.h, 87
- include/window2D.h, 89
- include/window3D.h, 90
- include/windowCoordinateSystem.hpp, 90
- include/windowRasterized.hpp, 90
- include/windowVectorized.hpp, 91
- Interval
 - cf::Interval, 29
- invert
 - cf::Color, 21
- isPointVector
 - cf::Vec3, 45
- LSystem
 - cf, 12
- LSystem_Controller
 - cf::LSystem_Controller, 36
 - cf::LSystem_Controller::iterator, 32
- Line
 - cf::Line, 35
- LineType
 - cf::Window2D, 53
 - cf::WindowCoordinateSystem, 73
 - cf::WindowRasterized, 79
 - cf::WindowVectorized, 81
- lineType
 - cf::Line, 35
- lineWidth
 - cf::Circle, 18
 - cf::CirclePartition, 19
 - cf::Line, 35
 - cf::Rect, 42
- m_AngleAdjustment
 - cf::Window3D, 71
- m_CameraAdjustment
 - cf::Window3D, 71
- m_CameraType
 - cf::Window3D, 71
- m_DistAdjustment
 - cf::Window3D, 71
- m_FreeCamera_LookDirection
 - cf::Window3D, 71
- m_FreeCamera_UpVector
 - cf::Window3D, 71
- m_FreeCamera_position
 - cf::Window3D, 71
- m_FristShowCall
 - cf::Window2D, 63
- m_Image
 - cf::Window2D, 63
- m_IntervalChanged
 - cf::Window2D, 63
- m_IntervalX
 - cf::Window2D, 63
- m_IntervalY
 - cf::Window2D, 63
- m_InvertYAxis
 - cf::Window2D, 63
- m_LookAt
 - cf::Window3D, 71
- m_LookAtDistance
 - cf::Window3D, 71
- m_MouseCallBackStorage
 - cf::Window2D, 63
- m_RotationAngle_X
 - cf::Window3D, 71
- m_RotationAngle_Y
 - cf::Window3D, 72
- m_WindowName
 - cf::Window2D, 63
- m_WindowScale
 - cf::Window2D, 64
- MAGENTA
 - cf::Color, 24
- max
 - cf::Interval, 30
- min
 - cf::Interval, 30

normalize
 cf::Vec3, 45
 ORANGE
 cf::Color, 24
 ORB
 cf, 13
 operator cf::Point
 cf::Vec3, 46
 operator cf::Vec3< false >
 cf::Vec3, 46
 operator const glm::vec3 &
 cf::Vec3, 46
 operator cv::Point
 cf::Point, 39
 operator glm::vec3
 cf::Vec3, 46
 operator!=
 cf::Color, 21
 cf::LSystem_Controller::iterator, 32
 cf::Point, 39
 operator<
 cf::Color, 22
 operator<<
 cf::Color, 23
 cf::Interval, 29
 computerGeometry.hpp, 86
 utils.h, 88, 89
 operator<<(>
 cf::Vec3, 50
 operator<=
 cf::Color, 22
 operator>
 cf::Color, 22
 operator>=
 cf::Color, 23
 operator*
 cf::Color, 21, 23
 cf::LSystem_Controller::iterator, 32
 cf::Point, 39, 40
 cf::Vec3, 47, 50
 operator*=
 cf::Color, 21
 cf::Point, 39
 cf::Vec3, 47
 operator+
 cf::Color, 21
 cf::Point, 39
 cf::Vec3, 47
 operator++
 cf::LSystem_Controller::iterator, 32
 operator+=
 cf::Color, 22
 cf::Point, 40
 cf::Vec3, 48
 operator-
 cf::Color, 22
 cf::Point, 40
 cf::Vec3, 48
 operator-=
 cf::Color, 22
 cf::Point, 40
 cf::Vec3, 48
 operator/
 cf::Color, 22, 23
 cf::Point, 40
 operator/=
 cf::Color, 22
 cf::Point, 40
 operator=
 cf::Vec3, 48
 cf::Window2D, 60
 operator==
 cf::Color, 22
 cf::Point, 40
 operator%
 cf::Vec3, 46
 operator%=
 cf::Vec3, 47
 operator[]
 cf::Vec3, 48, 49

 PINK
 cf::Color, 24
 Point
 cf::Point, 39
 point1
 cf::Line, 35
 cf::Rect, 42
 point2
 cf::Line, 35
 cf::Rect, 42
 PointVector
 cf, 13
 printWindowUsage
 cf::Window3D, 69

 r
 cf::Color, 24
 README.md, 91
 RED
 cf::Color, 24
 radian2degree
 cf, 13
 radius
 cf::Circle, 18
 cf::CirclePartition, 19
 RandomColor
 cf::Color, 23
 read
 cf::IteratedFunctionSystem, 31
 cf::LindenmayerSystem, 34
 cf::Orbit, 38
 readAntString
 cf, 14
 readDATFile
 cf, 14
 readFloat

- cf::Console, 25
- readInt
 - cf::Console, 26
- readPaletteFromFile
 - cf, 14
- readString
 - cf::Console, 26
- Rect
 - cf::Rect, 42
- RelativeDirection
 - cf::Direction, 27
- resetInterval
 - cf::Window2D, 60
- resize
 - cf::Window2D, 61
- saveImage
 - cf::Window2D, 61
- setCamera
 - cf::Window3D, 69
- setColor
 - cf::Window2D, 61
- setColor_imageSpace
 - cf::WindowVectorized, 83
- setInterval
 - cf::WindowCoordinateSystem, 77
 - cf::WindowVectorized, 83
- setInvertYAxis
 - cf::Window2D, 61
- setMaxFPS
 - cf::Window3D, 70
- setNewInterval
 - cf::Window2D, 61
- setWindowDisplayScale
 - cf::Window2D, 62
- setW
 - cf::Vec3, 49
- setX
 - cf::Vec3, 49
- setY
 - cf::Vec3, 50
- show
 - cf::Window2D, 62
- startAngle
 - cf::CirclePartition, 19
- startDrawing
 - cf::Window3D, 70
- toString
 - cf::Direction, 28
- transformPoint_fromImage_toInterval
 - cf::WindowVectorized, 83
- transformPoint_fromInterval_toImage
 - cf::WindowVectorized, 84
- translateIntervalPostion
 - cf::Interval, 29
- utils.h
 - operator<<, 88, 89
- Vec3
 - cf::Vec3, 44
- Vec3<!POINTVECTOR >
 - cf::Vec3, 50
- WHITE
 - cf::Color, 25
- waitKey
 - cf::Console, 26
 - cf::Window2D, 62
- waitMouseInput
 - cf::Window2D, 62
- Window2D
 - cf::Window2D, 53
- Window3D
 - cf::Window3D, 66
- WindowCoordinateSystem
 - cf::WindowCoordinateSystem, 73
- WindowRasterized
 - cf::WindowRasterized, 79
- WindowVectorized
 - cf::WindowVectorized, 81
- x
 - cf::Point, 41
- y
 - cf::Point, 41
- YELLOW
 - cf::Color, 25