CF_CG-Lib

2.718

Generated by Doxygen 1.8.14

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 13

6.1.1.1 DirectionVector 13

6.1.1.2 DirectionVector_d 13

6.1.1.3 DirectionVector_f 13

6.1.1.4 DirectionVector_ld 13

6.1.1.5 dMultiVector 13

6.1.1.6 fMultiVector 14

6.1.1.7 IFS 14

6.1.1.8 ldMultiVector 14

6.1.1.9 LSystem 14

6.1.1.10 ORB 14
6.1.1.11 PointVector 14
6.1.1.12 PointVector_d 14
6.1.1.13 PointVector_f 15
6.1.1.14 PointVector_ld 15
6.1.1.15 Vec 15
6.1.2 Function Documentation 15
6.1.2.1 _removeWindowsSpecificCarriageReturn() 15
6.1.2.2 abs() 15
6.1.2.3 degree2radian() 15
6.1.2.4 radian2degree() 16
6.1.2.5 readAntString() 16
6.1.2.6 readDATFile() 16
6.1.2.7 readPaletteFromFile() 17
6.2 cf::internal Namespace Reference 17
6.3 cf::literals Namespace Reference 17

7 Class Documentation 19

7.1 cf::internal::_ProtectedFunction< _ReturnType, _Args > Struct Template Reference 19
7.2 cf::internal::_ProtectedFunction< _ReturnType(_Args...)> Struct Template Reference 19
7.2.1 Member Function Documentation 19
7.2.1.1 operator()() 19
7.2.1.2 set() 20
7.3 cf::MultiVector< _ValueType >::Blade Struct Reference 20
7.3.1 Member Enumeration Documentation 21
7.3.1.1 TYPE 21
7.3.2 Constructor & Destructor Documentation 21
7.3.2.1 Blade() [1/3] 21
7.3.2.2 Blade() [2/3] 21
7.3.2.3 Blade() [3/3] 21
7.3.3 Member Function Documentation 22

7.3.3.1 getCompleteType() . . . 22
7.3.3.2 operator=() . . . 22
7.3.3.3 operator==() . . . 22
7.3.3.4 sameType() . . . 22
7.3.3.5 sortBladeTypes() . . . 22
7.3.3.6 type2int() . . . 22
7.3.3.7 TYPE_TO_STRING() . . . 23
7.3.4 Friends And Related Function Documentation . . . 23
7.3.4.1 operator<< . . . 23
7.3.5 Member Data Documentation . . . 23
7.3.5.1 factor . . . 23
7.3.5.2 outerProduct . . . 23
7.3.5.3 type . . . 23
7.4 cf::Circle Struct Reference . . . 24
7.4.1 Detailed Description . . . 24
7.4.2 Constructor & Destructor Documentation . . . 24
7.4.2.1 Circle() . . . 24
7.4.3 Member Data Documentation . . . 24
7.4.3.1 center . . . 24
7.4.3.2 color . . . 25
7.4.3.3 lineWidth . . . 25
7.4.3.4 radius . . . 25
7.5 cf::CirclePartition Struct Reference . . . 25
7.5.1 Detailed Description . . . 25
7.5.2 Constructor & Destructor Documentation . . . 26
7.5.2.1 CirclePartition() . . . 26
7.5.3 Member Data Documentation . . . 26
7.5.3.1 center . . . 26
7.5.3.2 color . . . 26
7.5.3.3 endAngle . . . 26

7.5.3.4 lineWidth 26
7.5.3.5 radius 26
7.5.3.6 startAngle 27
7.6 cf::Color Struct Reference 27
7.6.1 Detailed Description 28
7.6.2 Constructor & Destructor Documentation 28
7.6.2.1 Color() 28
7.6.3 Member Function Documentation 28
7.6.3.1 CreateEndlessColorIterator() 28
7.6.3.2 invert() 29
7.6.3.3 operator"!=() 29
7.6.3.4 operator∗() 29
7.6.3.5 operator∗=() 29
7.6.3.6 operator+() 29
7.6.3.7 operator+=() 30
7.6.3.8 operator-() 30
7.6.3.9 operator-=() 30
7.6.3.10 operator/() 30
7.6.3.11 operator/=() 30
7.6.3.12 operator<() 30
7.6.3.13 operator<=() 30
7.6.3.14 operator==() 31
7.6.3.15 operator>() 31
7.6.3.16 operator>=() 31
7.6.3.17 RandomColor() 31
7.6.4 Friends And Related Function Documentation 31
7.6.4.1 operator∗ 31
7.6.4.2 operator/ 31
7.6.4.3 operator<< 32
7.6.5 Member Data Documentation 32

7.6.5.1 b 32
7.6.5.2 BLACK 32
7.6.5.3 BLUE 32
7.6.5.4 CYAN 32
7.6.5.5 g 32
7.6.5.6 GREEN 32
7.6.5.7 GREY 33
7.6.5.8 MAGENTA 33
7.6.5.9 ORANGE 33
7.6.5.10 PINK 33
7.6.5.11 r 33
7.6.5.12 RED 33
7.6.5.13 WHITE 33
7.6.5.14 YELLOW 34
7.7 cf::Console Struct Reference 34
7.7.1 Detailed Description 34
7.7.2 Member Function Documentation 34
7.7.2.1 clearConsole() 34
7.7.2.2 printError() 35
7.7.2.3 printWarning() 35
7.7.2.4 readFloat() 35
7.7.2.5 readInt() 35
7.7.2.6 readString() 36
7.7.2.7 waitKey() 36
7.8 cf::Direction Struct Reference 36
7.8.1 Detailed Description 36
7.8.2 Member Enumeration Documentation 36
7.8.2.1 AbsoluteDirection 36
7.8.2.2 RelativeDirection 37
7.8.3 Member Function Documentation 37

7.8.3.1 getNextiDirection() 37
7.8.3.2 toString() [1/2] 37
7.8.3.3 toString() [2/2] 38
7.9 cf::Interval Struct Reference 38
7.9.1 Detailed Description 38
7.9.2 Constructor & Destructor Documentation 38
7.9.2.1 Interval() 39
7.9.3 Member Function Documentation 39
7.9.3.1 translateIntervalPostion() [1/2] 39
7.9.3.2 translateIntervalPostion() [2/2] 39
7.9.4 Friends And Related Function Documentation 39
7.9.4.1 operator<< 39
7.9.5 Member Data Documentation 39
7.9.5.1 max 39
7.9.5.2 min 40
7.10 cf::IteratedFunctionSystem Struct Reference 40
7.10.1 Detailed Description 40
7.10.2 Member Function Documentation 40
7.10.2.1 getAllTransformation() 40
7.10.2.2 getName() 40
7.10.2.3 getNumTransformations() 41
7.10.2.4 getRangeX() 41
7.10.2.5 getRangeY() 41
7.10.2.6 getTransformation() 41
7.10.2.7 read() 41
7.11 cf::LSystem_Controller::iterator Struct Reference 41
7.11.1 Constructor & Destructor Documentation 42
7.11.1.1 iterator() 42
7.11.2 Member Function Documentation 42
7.11.2.1 operator"!=() 42

7.11.2.2 operator∗() 42
7.11.2.3 operator++() 42
7.11.3 Friends And Related Function Documentation 42
7.11.3.1 LSystem_Controller 43
7.12 cf::LindenmayerSystem Struct Reference 43
7.12.1 Detailed Description 43
7.12.2 Member Function Documentation 43
7.12.2.1 clearWindowEachTime() 43
7.12.2.2 getAdjustmentAngle() 44
7.12.2.3 getAllProductions() 44
7.12.2.4 getAxiom() 44
7.12.2.5 getName() 44
7.12.2.6 getNumProductions() 44
7.12.2.7 getProduction() 44
7.12.2.8 getRangeX() 44
7.12.2.9 getRangeY() 45
7.12.2.10 getScale() 45
7.12.2.11 getStartAngle() 45
7.12.2.12 read() 45
7.13 cf::Line Struct Reference 45
7.13.1 Detailed Description 46
7.13.2 Constructor & Destructor Documentation 46
7.13.2.1 Line() 46
7.13.3 Member Data Documentation 46
7.13.3.1 color 46
7.13.3.2 lineType 46
7.13.3.3 lineWidth 46
7.13.3.4 point1 47
7.13.3.5 point2 47
7.14 cf::LSystem_Controller Struct Reference 47

7.14.1 Detailed Description 47
7.14.2 Constructor & Destructor Documentation 47
7.14.2.1 LSystem_Controller() 48
7.14.3 Member Function Documentation 48
7.14.3.1 begin() 48
7.14.3.2 end() 48
7.15 cf::MultiVector< _ValueType > Struct Template Reference 48
7.15.1 Detailed Description 50
7.15.2 Constructor & Destructor Documentation 50
7.15.2.1 MultiVector() [1/3] 50
7.15.2.2 MultiVector() [2/3] 50
7.15.2.3 MultiVector() [3/3] 50
7.15.3 Member Function Documentation 50
7.15.3.1 getData() 51
7.15.3.2 operator &() 51
7.15.3.3 operator &=() 51
7.15.3.4 operator _VType() 51
7.15.3.5 operator%() 51
7.15.3.6 operator%=() 51
7.15.3.7 operator*() [1/3] 52
7.15.3.8 operator*() [2/3] 52
7.15.3.9 operator*() [3/3] 52
7.15.3.10 operator*=() [1/2] 52
7.15.3.11 operator*=() [2/2] 52
7.15.3.12 operator+() [1/3] 52
7.15.3.13 operator+() [2/3] 53
7.15.3.14 operator+() [3/3] 53
7.15.3.15 operator+=() [1/2] 53
7.15.3.16 operator+=() [2/2] 53
7.15.3.17 operator-() [1/3] 53

7.15.3.18 operator-() [2/3] 53
7.15.3.19 operator-() [3/3] 54
7.15.3.20 operator-=() [1/2] 54
7.15.3.21 operator-=() [2/2] 54
7.15.3.22 operator/() 54
7.15.3.23 operator/=() 54
7.15.3.24 operator=() 54
7.15.3.25 operator==() 55
7.15.3.26 operator^() 55
7.15.3.27 operator^=() 55
7.15.3.28 operator∼() 55
7.15.3.29 setData() 55
7.15.4 Friends And Related Function Documentation 55
7.15.4.1 MultiVector 55
7.15.4.2 operator<< 56
7.16 cf::Orbit Struct Reference 56
7.16.1 Detailed Description 56
7.16.2 Member Function Documentation 56
7.16.2.1 getAllFactors() 56
7.16.2.2 getAllStartingPoints() 57
7.16.2.3 getName() 57
7.16.2.4 getNumFactors() 57
7.16.2.5 getNumStartingPoints() 57
7.16.2.6 getRangeX() 57
7.16.2.7 getRangeY() 57
7.16.2.8 read() 57
7.17 cf::Point Struct Reference 58
7.17.1 Detailed Description 58
7.17.2 Constructor & Destructor Documentation 58
7.17.2.1 Point() 59

7.17.3 Member Function Documentation 59
7.17.3.1 operator cv::Point() 59
7.17.3.2 operator"!=() 59
7.17.3.3 operator∗() 59
7.17.3.4 operator∗=() 59
7.17.3.5 operator+() 59
7.17.3.6 operator+=() 60
7.17.3.7 operator-() 60
7.17.3.8 operator-=() 60
7.17.3.9 operator/() 60
7.17.3.10 operator/=() 60
7.17.3.11 operator==() 60
7.17.4 Friends And Related Function Documentation 60
7.17.4.1 operator∗ 61
7.17.4.2 operator/ 61
7.17.5 Member Data Documentation 61
7.17.5.1 x 61
7.17.5.2 y 61
7.18 cf::Rect Struct Reference 61
7.18.1 Detailed Description 62
7.18.2 Constructor & Destructor Documentation 62
7.18.2.1 Rect() 62
7.18.3 Member Data Documentation 62
7.18.3.1 color 62
7.18.3.2 lineWidth 62
7.18.3.3 point1 62
7.18.3.4 point2 63
7.19 cf::Color::SimpleEndlessIterator< _Size > Struct Template Reference 63
7.19.1 Constructor & Destructor Documentation 63
7.19.1.1 SimpleEndlessIterator() 63

7.19.2 Member Function Documentation 63
7.19.2.1 operator∗() 63
7.19.2.2 operator++() [1/2] 64
7.19.2.3 operator++() [2/2] 64
7.19.2.4 operator->() 64
7.20 cf::SimpleSignal Struct Reference 64
7.20.1 Member Function Documentation 64
7.20.1.1 fireSignal() 64
7.20.1.2 waitSignal() 65
7.21 cf::Vec3< IS_POINTVECTOR, _ValueType > Class Template Reference 65
7.21.1 Detailed Description 67
7.21.2 Member Typedef Documentation 67
7.21.2.1 self_type 67
7.21.2.2 value_type 67
7.21.3 Constructor & Destructor Documentation 67
7.21.3.1 Vec3() [1/3] 67
7.21.3.2 Vec3() [2/3] 68
7.21.3.3 Vec3() [3/3] 68
7.21.4 Member Function Documentation 68
7.21.4.1 getVector90Degree() 68
7.21.4.2 getW() 68
7.21.4.3 getX() 69
7.21.4.4 getY() 69
7.21.4.5 isPointVector() 69
7.21.4.6 length() 69
7.21.4.7 normalize() 70
7.21.4.8 operator cf::Point() 70
7.21.4.9 operator cf::Vec3< PV_RHS, _VType >() 70
7.21.4.10 operator const glmVec3 &() 70
7.21.4.11 operator glm::vec3() 70

7.21.4.12 operator"!=() 70
7.21.4.13 operator%() 71
7.21.4.14 operator%=() 71
7.21.4.15 operator∗() [1/2] 71
7.21.4.16 operator∗() [2/2] 72
7.21.4.17 operator∗=() 72
7.21.4.18 operator+() 72
7.21.4.19 operator+=() 73
7.21.4.20 operator-() 73
7.21.4.21 operator-=() 73
7.21.4.22 operator=() [1/2] 73
7.21.4.23 operator=() [2/2] 73
7.21.4.24 operator==() 73
7.21.4.25 operator[]() [1/2] 74
7.21.4.26 operator[]() [2/2] 74
7.21.4.27 setW() 75
7.21.4.28 setX() 75
7.21.4.29 setY() 75
7.21.5 Friends And Related Function Documentation 75
7.21.5.1 operator∗ 75
7.21.5.2 operator<<) 76
7.21.5.3 Vec3 76
7.22 cf::Window2D Class Reference 76
7.22.1 Detailed Description 78
7.22.2 Member Enumeration Documentation 79
7.22.2.1 LineType 79
7.22.3 Constructor & Destructor Documentation 79
7.22.3.1 Window2D() [1/2] 79
7.22.3.2 Window2D() [2/2] 79
7.22.3.3 ∼Window2D() 80

7.22.4 Member Function Documentation 80

7.22.4.1 _convertFromNewInterval() 80

7.22.4.2 _convertToNewInterval() 80

7.22.4.3 _correctYValue() 80

7.22.4.4 _CreateUniqueWindowName() 80

7.22.4.5 _window2foreground() 80

7.22.4.6 clear() 81

7.22.4.7 drawAxis() 81

7.22.4.8 drawCircle() [1/2] 81

7.22.4.9 drawCircle() [2/2] 81

7.22.4.10 drawCirclePart() [1/2] 82

7.22.4.11 drawCirclePart() [2/2] 82

7.22.4.12 drawLine() [1/2] 82

7.22.4.13 drawLine() [2/2] 83

7.22.4.14 drawRectangle() [1/2] 83

7.22.4.15 drawRectangle() [2/2] 84

7.22.4.16 drawSpecializedLine() 84

7.22.4.17 flippHorizontal() 84

7.22.4.18 flippVertical() 84

7.22.4.19 floodFill() 85

7.22.4.20 getColor() 85

7.22.4.21 getHeight() 85

7.22.4.22 getImage() 85

7.22.4.23 getIntervalX() 86

7.22.4.24 getIntervalY() 86

7.22.4.25 getInvertYAxis() 86

7.22.4.26 getWidth() 86

7.22.4.27 getWindowDisplayScale() 86

7.22.4.28 operator=() 86

7.22.4.29 resetInterval() 87

7.22.4.30 resize() 87
7.22.4.31 saveImage() 87
7.22.4.32 setColor() 88
7.22.4.33 setInvertYAxis() 88
7.22.4.34 setNewInterval() 88
7.22.4.35 setWindowDisplayScale() 88
7.22.4.36 show() 89
7.22.4.37 waitKey() 89
7.22.4.38 waitMouseInput() [1/2] 89
7.22.4.39 waitMouseInput() [2/2] 89
7.22.5 Member Data Documentation 90
7.22.5.1 m_FristShowCall 90
7.22.5.2 m_Image 90
7.22.5.3 m_IntervalChanged 90
7.22.5.4 m_IntervalX 90
7.22.5.5 m_IntervalY 90
7.22.5.6 m_InvertYAxis 90
7.22.5.7 m_MouseCallBackStorage 91
7.22.5.8 m_WindowName 91
7.22.5.9 m_WindowScale 91
7.23 cf::Window3D Struct Reference 91
7.23.1 Detailed Description 93
7.23.2 Member Enumeration Documentation 93
7.23.2.1 CameraType 93
7.23.2.2 MouseButton 94
7.23.2.3 MouseButtonEvent 94
7.23.3 Constructor & Destructor Documentation 94
7.23.3.1 Window3D() 94
7.23.3.2 ∼Window3D() 95
7.23.4 Member Function Documentation 95

7.23.4.1 _AdjustCamera() 95

7.23.4.2 _AdjustColorOpenGL() 95

7.23.4.3 _ZoomCamera() 95

7.23.4.4 clear() 95

7.23.4.5 disableLighting() 95

7.23.4.6 draw() 96

7.23.4.7 drawAxis() 96

7.23.4.8 drawCube() 96

7.23.4.9 drawCylinder() [1/4] 96

7.23.4.10 drawCylinder() [2/4] 97

7.23.4.11 drawCylinder() [3/4] 97

7.23.4.12 drawCylinder() [4/4] 97

7.23.4.13 drawSphere() 97

7.23.4.14 enableLighting() 98

7.23.4.15 forceDisplay() 98

7.23.4.16 getWindowHeight() 98

7.23.4.17 getWindowWidth() 98

7.23.4.18 handleKeyboardInput() 98

7.23.4.19 handleMousePressedMovement() 99

7.23.4.20 handleMousePressEvent() 99

7.23.4.21 printWindowUsage() 100

7.23.4.22 setCamera() 100

7.23.4.23 setMaxFPS() 100

7.23.4.24 startDrawing() 100

7.23.5 Friends And Related Function Documentation 101

7.23.5.1 _DrawingFunction 101

7.23.5.2 _KeyboardCallbackFunction 101

7.23.5.3 _MouseCtlClickCallbackFunction 101

7.23.5.4 _MouseCtlMotionCallbackFunction 101

7.23.5.5 _WindowResizeEvent 101

7.23.6 Member Data Documentation 101
7.23.6.1 m_AngleAdjustment 102
7.23.6.2 m_CameraAdjustment 102
7.23.6.3 m_CameraPositionCorrection 102
7.23.6.4 m_CameraType 102
7.23.6.5 m_DistAdjustment 102
7.23.6.6 m_FreeCamera_LookDirection 102
7.23.6.7 m_FreeCamera_position 102
7.23.6.8 m_FreeCamera_UpVector 103
7.23.6.9 m_LookAt 103
7.23.6.10 m_LookAtDistance 103
7.23.6.11 m_RotationAngle_X 103
7.23.6.12 m_RotationAngle_Y 103
7.24 cf::Window3DObject Struct Reference 103
7.24.1 Member Enumeration Documentation 105
7.24.1.1 CameraType 105
7.24.1.2 MouseButton 105
7.24.1.3 MouseButtonEvent 105
7.24.2 Constructor & Destructor Documentation 105
7.24.2.1 ∼Window3DObject() 105
7.24.3 Member Function Documentation 106
7.24.3.1 clear() 106
7.24.3.2 createWindow3DObject() 106
7.24.3.3 disableLighting() 106
7.24.3.4 drawAxis() 106
7.24.3.5 drawCube() 106
7.24.3.6 drawCylinder() [1/4] 107
7.24.3.7 drawCylinder() [2/4] 107
7.24.3.8 drawCylinder() [3/4] 107
7.24.3.9 drawCylinder() [4/4] 107

7.24.3.10 drawSphere() 107
7.24.3.11 enableLighting() 108
7.24.3.12 exit() 108
7.24.3.13 forceDisplay() 108
7.24.3.14 getWindowHeight() 108
7.24.3.15 getWindowWidth() 108
7.24.3.16 printWindowUsage() 108
7.24.3.17 setCamera() 108
7.24.3.18 setDrawingFunction() 109
7.24.3.19 setKeyboardInputFunction() 109
7.24.3.20 setMaxFPS() 109
7.24.3.21 setMousePressedMovementFunction() 109
7.24.3.22 setMousePressEvent() 109
7.24.3.23 waitKeyPressed() 110
7.25 cf::WindowCoordinateSystem Struct Reference 110
7.25.1 Detailed Description 112
7.25.2 Member Enumeration Documentation 112
7.25.2.1 LineType 112
7.25.3 Constructor & Destructor Documentation 112
7.25.3.1 WindowCoordinateSystem() 112
7.25.3.2 ∼WindowCoordinateSystem() 112
7.25.4 Member Function Documentation 113
7.25.4.1 clear() 113
7.25.4.2 convert_intervalLength_to_pixelLength() 113
7.25.4.3 convert_pixelLength_to_intervalLength() 113
7.25.4.4 drawAxis() 113
7.25.4.5 drawCircle() 114
7.25.4.6 drawCirclePart() [1/2] 114
7.25.4.7 drawCirclePart() [2/2] 115
7.25.4.8 drawLine() 115

7.25.4.9 drawLinearEquation() [1/4] 115
7.25.4.10 drawLinearEquation() [2/4] 116
7.25.4.11 drawLinearEquation() [3/4] 116
7.25.4.12 drawLinearEquation() [4/4] 117
7.25.4.13 drawPoint() 117
7.25.4.14 floodFill() 117
7.25.4.15 getColor() 118
7.25.4.16 getHeight() 118
7.25.4.17 getIntervalX() 118
7.25.4.18 getIntervalY() 118
7.25.4.19 getWidth() 119
7.25.4.20 getWindowDisplayScale() 119
7.25.4.21 saveImage() 119
7.25.4.22 setColor() 119
7.25.4.23 setInterval() 119
7.25.4.24 setWindowDisplayScale() 120
7.25.4.25 show() 120
7.25.4.26 waitKey() 120
7.25.4.27 waitMouseInput() [1/2] 121
7.25.4.28 waitMouseInput() [2/2] 121
7.26 cf::WindowCoordinateSystem3D Struct Reference 121
7.26.1 Member Enumeration Documentation 122
7.26.1.1 MULTI_VECTOR_TYPE 122
7.26.1.2 SPACE_TYPE 122
7.26.2 Constructor & Destructor Documentation 123
7.26.2.1 WindowCoordinateSystem3D() 123
7.26.3 Member Function Documentation 123
7.26.3.1 beginDrawing() 123
7.26.3.2 clearWindow() 123
7.26.3.3 drawCircle() 123

7.26.3.4 drawLine() 124
7.26.3.5 drawMultiVector() 124
7.26.3.6 drawPlane() [1/2] 124
7.26.3.7 drawPlane() [2/2] 124
7.26.3.8 drawPoint() 124
7.26.3.9 drawSphere() 125
7.26.3.10 getMultiVectorType() 125
7.26.3.11 waitKey() 125
7.27 cf::WindowRasterized Struct Reference 125
7.27.1 Detailed Description 127
7.27.2 Member Enumeration Documentation 127
7.27.2.1 LineType 127
7.27.3 Constructor & Destructor Documentation 127
7.27.3.1 WindowRasterized() [1/2] 127
7.27.3.2 WindowRasterized() [2/2] 127
7.27.3.3 ∼WindowRasterized() 128
7.27.4 Member Function Documentation 128
7.27.4.1 clear() 128
7.27.4.2 drawCircle() [1/2] 128
7.27.4.3 drawCircle() [2/2] 128
7.27.4.4 drawLine() [1/2] 129
7.27.4.5 drawLine() [2/2] 129
7.27.4.6 drawRectangle() [1/2] 129
7.27.4.7 drawRectangle() [2/2] 130
7.27.4.8 drawSpecializedLine() 130
7.27.4.9 flippHorizontal() 130
7.27.4.10 flippVertical() 130
7.27.4.11 floodFill() 130
7.27.4.12 getColor() 131
7.27.4.13 getHeight() 131

7.27.4.14 getImage() 131
7.27.4.15 getWidth() 131
7.27.4.16 getWindowDisplayScale() 131
7.27.4.17 resize() 131
7.27.4.18 saveImage() 132
7.27.4.19 setColor() 132
7.27.4.20 setWindowDisplayScale() 132
7.27.4.21 show() 132
7.27.4.22 waitKey() 133
7.27.4.23 waitMouseInput() [1/2] 133
7.27.4.24 waitMouseInput() [2/2] 133
7.28 cf::WindowVectorized Struct Reference 134
7.28.1 Detailed Description 135
7.28.2 Member Enumeration Documentation 136
7.28.2.1 LineType 136
7.28.3 Constructor & Destructor Documentation 136
7.28.3.1 WindowVectorized() [1/2] 136
7.28.3.2 WindowVectorized() [2/2] 136
7.28.3.3 ∼WindowVectorized() 137
7.28.4 Member Function Documentation 137
7.28.4.1 clear() 137
7.28.4.2 convert_intervalLength_to_pixelLength() 137
7.28.4.3 convert_pixelLength_to_intervalLength() 137
7.28.4.4 drawAxis() 138
7.28.4.5 drawCircle() [1/2] 138
7.28.4.6 drawCircle() [2/2] 138
7.28.4.7 drawCirclePart() [1/2] 139
7.28.4.8 drawCirclePart() [2/2] 139
7.28.4.9 drawLine() [1/2] 139
7.28.4.10 drawLine() [2/2] 140

7.28.4.11 drawRectangle() [1/2] 140
7.28.4.12 drawRectangle() [2/2] 140
7.28.4.13 drawSpecializedLine() 140
7.28.4.14 floodFill() 141
7.28.4.15 getColor() 141
7.28.4.16 getColor_imageSpace() 141
7.28.4.17 getHeight() 142
7.28.4.18 getImage() 142
7.28.4.19 getIntervalX() 142
7.28.4.20 getIntervalY() 142
7.28.4.21 getWidth() 143
7.28.4.22 getWindowDisplayScale() 143
7.28.4.23 saveImage() 143
7.28.4.24 setColor() 143
7.28.4.25 setColor_imageSpace() 143
7.28.4.26 setInterval() 144
7.28.4.27 setWindowDisplayScale() 144
7.28.4.28 show() 144
7.28.4.29 transformPoint_fromImage_toInterval() 144
7.28.4.30 transformPoint_fromInterval_toImage() 145
7.28.4.31 waitKey() 145
7.28.4.32 waitMouseInput() [1/2] 145
7.28.4.33 waitMouseInput() [2/2] 146

8 File Documentation 147

8.1 include/computerGeometry.hpp File Reference 147

8.1.1 Macro Definition Documentation 148

8.1.1.1 MSG `[1/5]` 148

8.1.1.2 MSG `[2/5]` 148

8.1.1.3 MSG `[3/5]` 148

8.1.1.4 MSG `[4/5]` 148

8.1.1.5 MSG `[5/5]` 149

8.1.2 Function Documentation 149

8.1.2.1 operator<<() 149

8.2 include/computerGeometry3D.hpp File Reference 149

8.3 include/IFS.h File Reference 150

8.4 include/internal.hpp File Reference 150

8.5 include/LSystem.h File Reference 151

8.6 include/ORB.h File Reference 151

8.7 include/utils.h File Reference 152

8.7.1 Function Documentation 153

8.7.1.1 operator<<() `[1/5]` 153

8.7.1.2 operator<<() `[2/5]` 153

8.7.1.3 operator<<() `[3/5]` 153

8.7.1.4 operator<<() `[4/5]` 153

8.7.1.5 operator<<() `[5/5]` 153

8.8 include/window2D.h File Reference 153

8.9 include/window3D.h File Reference 154

8.10 include/window3DObjectbased.h File Reference 154

8.11 include/windowCoordinateSystem.h File Reference 155

8.12 include/windowCoordinateSystem3D.h File Reference 155

8.13 include/windowRasterized.h File Reference 155

8.14 include/windowVectorized.h File Reference 156

8.15 README.md File Reference 156

Index 157

Chapter 1

CF_CG-Lib

This library is inteted to be used in 'Chaos und Fraktale' and 'Computer Geometry', lessons from 'Hochschule Darmstadt'.

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

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

cf 11
cf::internal 17
cf::literals 17

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

cf::internal::_ProtectedFunction< _ReturnType, _Args > 19
cf::internal::_ProtectedFunction< _ReturnType(_Args...)> 19
cf::internal::_ProtectedFunction< bool(MouseButton, int, int)> 19
cf::internal::_ProtectedFunction< void(cf::Window3DObject &)> 19
cf::internal::_ProtectedFunction< void(MouseButton, MouseButtonEvent, int, int)> 19
cf::internal::_ProtectedFunction< void(unsigned char, int, int)> 19
cf::MultiVector< _ValueType >::Blade 20
cf::Circle 24
cf::CirclePartition 25
cf::Color 27
cf::Console 34
cf::Direction 36
cf::Interval 38
cf::IteratedFunctionSystem 40
cf::LSystem_Controller::iterator 41
cf::LindenmayerSystem 43
cf::Line 45
cf::LSystem_Controller 47
cf::MultiVector< _ValueType > 48
cf::Orbit 56
cf::Point 58
cf::Rect 61
cf::Color::SimpleEndlessIterator< _Size > 63
cf::SimpleSignal 64
cf::Vec3< IS_POINTVECTOR, _ValueType > 65
cf::Window2D 76
- cf::WindowCoordinateSystem 110
- cf::WindowRasterized 125
- cf::WindowVectorized 134

cf::Window3D 91
- cf::Window3DObject 103
- cf::WindowCoordinateSystem3D 121

Chapter 4

Class Index

4.1 Class List

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

cf::internal::_ProtectedFunction< _ReturnType, _Args > 19
cf::internal::_ProtectedFunction< _ReturnType(_Args...)> 19
cf::MultiVector< _ValueType >::Blade 20
cf::Circle
The Circle struct Simple parameter wrapper struct 24
cf::CirclePartition
The CirclePartition struct Simple parameter wrapper struct 25
cf::Color
The Color struct offers a class for rgb access 27
cf::Console
The Console struct offers utility functions for 'console' 34
cf::Direction
The Direction struct for getting absolute directions from a current direction and a relative direction 36
cf::Interval
The Interval struct provides functionallity to translate values from one interval into another . . . 38
cf::IteratedFunctionSystem
The IteratedFunctionSystem class lazy people (like myself) may use the IFS tyepdef 40
cf::LSystem_Controller::iterator 41
cf::LindenmayerSystem
The LindenmayerSystem class lazy people (like myself) may use the IFS tyepdef 43
cf::Line
The Line struct Simple parameter wrapper struct 45
cf::LSystem_Controller
The LSystem_Controller struct
This class enables easy iterating above a given iteration depth
47
cf::MultiVector< _ValueType > 48
cf::Orbit
The Orbit class lazy people (like myself) may use the ORB tyepdef 56
cf::Point
The Point struct is a simple class for positon access on 2D images (imilar to cv::Point, but uses floats instead of integer) 58
cf::Rect
The Rect struct Simple parameter wrapper struct 61
cf::Color::SimpleEndlessIterator< _Size > 63

cf::SimpleSignal 64
cf::Vec3< IS_POINTVECTOR, _ValueType >
The Vec3 struct General class for vector operations 65
cf::Window2D
The Window2D struct offers advanced features used by WindowRasterized/WindowVertorized 76
cf::Window3D
The Window3D struct is the default class for accessing 3D content, creating more than 1 instance results in undefined behavior 91
cf::Window3DObject 103
cf::WindowCoordinateSystem
The WindowCoordinateSystem struct Default class for images and raster operations 110
cf::WindowCoordinateSystem3D 121
cf::WindowRasterized
The WindowRasterized struct Default struct for verctorized operations within a custom interval 125
cf::WindowVectorized
The WindowVectorized struct Default class for images and raster operations 134

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

include/computerGeometry.hpp 147
include/computerGeometry3D.hpp 149
include/IFS.h 150
include/internal.hpp 150
include/LSystem.h 151
include/ORB.h 151
include/utils.h 152
include/window2D.h 153
include/window3D.h 154
include/window3DObjectbased.h 154
include/windowCoordinateSystem.h 155
include/windowCoordinateSystem3D.h 155
include/windowRasterized.h 155
include/windowVectorized.h 156

Chapter 6

Namespace Documentation

6.1 cf Namespace Reference

Namespaces

- internal
- literals

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

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

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

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

- struct Point

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

  The Rect struct Simple parameter wrapper struct.
- struct SimpleSignal
- class Vec3

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

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

  The Window3D struct is the default class for accessing 3D content, creating more than 1 instance results in undefined behavior.
- struct Window3DObject
- struct WindowCoordinateSystem

  The WindowCoordinateSystem struct Default class for images and raster operations.
- struct WindowCoordinateSystem3D
- struct WindowRasterized

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

  The WindowVectorized struct Default class for images and raster operations.

Typedefs

- typedef Vec3< true, double > PointVector_d
- typedef Vec3< false, double > DirectionVector_d
- typedef Vec3< true, float > PointVector_f
- typedef Vec3< false, float > DirectionVector_f
- typedef Vec3< true, long double > PointVector_ld
- typedef Vec3< false, long double > DirectionVector_ld
- typedef PointVector_d PointVector

  PointVector Specialiaztion of general Vec3.
- typedef DirectionVector_d DirectionVector

  DirectionVector Specialiaztion of general Vec3, where component 'w' may not be written to.
- typedef MultiVector< long double > ldMultiVector
- typedef MultiVector< double > dMultiVector
- typedef MultiVector< float > fMultiVector
- typedef MultiVector< double > Vec
- typedef IteratedFunctionSystem IFS
- typedef LindenmayerSystem LSystem
- typedef Orbit ORB

Functions

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

  readDATFile Reads a ∗.dat file

6.1.1 Typedef Documentation

6.1.1.1 DirectionVector

```
typedef DirectionVector_d cf::DirectionVector
```

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

6.1.1.2 DirectionVector_d

```
typedef Vec3<false, double> cf::DirectionVector_d
```

6.1.1.3 DirectionVector_f

```
typedef Vec3<false, float> cf::DirectionVector_f
```

6.1.1.4 DirectionVector_ld

```
typedef Vec3<false, long double> cf::DirectionVector_ld
```

6.1.1.5 dMultiVector

```
typedef MultiVector<double> cf::dMultiVector
```

6.1.1.6 fMultiVector

```
typedef MultiVector<float> cf::fMultiVector
```

6.1.1.7 IFS

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

6.1.1.8 ldMultiVector

```
typedef MultiVector<long double> cf::ldMultiVector
```

6.1.1.9 LSystem

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

6.1.1.10 ORB

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

6.1.1.11 PointVector

```
typedef PointVector_d cf::PointVector
```

PointVector Specialiaztion of general Vec3.

6.1.1.12 PointVector_d

```
typedef Vec3<true, double> cf::PointVector_d
```

6.1.1.13 PointVector_f

```
typedef Vec3<true, float> cf::PointVector_f
```

6.1.1.14 PointVector_ld

```
typedef Vec3<true, long double> cf::PointVector_ld
```

6.1.1.15 Vec

```
typedef MultiVector<double> cf::Vec
```

6.1.2 Function Documentation

6.1.2.1 _removeWindowsSpecificCarriageReturn()

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

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

Parameters

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

6.1.2.2 abs()

```
template<typename _ValueType >
_ValueType cf::abs (
             const cf::MultiVector< _ValueType > & multiVector )
```

6.1.2.3 degree2radian()

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

degree2radian Converts a degree value to a radian value

Parameters

degreeValue	Degree value to be converted

Returns

Converted radian value

6.1.2.4 radian2degree()

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

radian2degree Converts a radian value to a degree value

Parameters

radianValue	Radian value to be converted

Returns

Converted degree value

6.1.2.5 readAntString()

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

readAntString

Parameters

filePath	Read ∗.ant file from path

Returns

6.1.2.6 readDATFile()

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

readDATFile Reads a ∗.dat file

Parameters

filePath	Read ∗.dat file from path

Returns

6.1.2.7 readPaletteFromFile()

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

readPaletteFromFile

Parameters

filePath	Read ∗.pal file from path

Returns

6.2 cf::internal Namespace Reference

Classes

- struct _ProtectedFunction
- struct _ProtectedFunction< _ReturnType(_Args...)>

6.3 cf::literals Namespace Reference

Chapter 7

Class Documentation

7.1 cf::internal::_ProtectedFunction< _ReturnType, _Args > Struct Template Reference

```
#include <internal.hpp>
```

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

- include/internal.hpp

7.2 cf::internal::_ProtectedFunction< _ReturnType(_Args...)> Struct Template Reference

```
#include <internal.hpp>
```

Public Member Functions

- template<typename _PT >
  void set (_PT &&forwardRef)
- template<typename... _FunctionArgs>
  _ReturnType operator() (_FunctionArgs &&... args)

7.2.1 Member Function Documentation

7.2.1.1 operator()()

```
template<typename _ReturnType , typename... _Args>
template<typename... _FunctionArgs>
_ReturnType cf::internal::_ProtectedFunction< _ReturnType(_Args...)>::operator() (
             _FunctionArgs &&... args ) [inline]
```

7.2.1.2 set()

```
template<typename _ReturnType , typename...  _Args>
template<typename _PT >
void cf::internal::_ProtectedFunction< _ReturnType(_Args...)>::set (
            _PT && forwardRef )  [inline]
```

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

- include/internal.hpp

7.3 cf::MultiVector< _ValueType >::Blade Struct Reference

```
#include <computerGeometry3D.hpp>
```

Public Types

- enum TYPE {
  TYPE::E1 = 1, TYPE::E2, TYPE::E3, TYPE::EINF,
  TYPE::E0, TYPE::VALUE = std::numeric_limits<int16_t>::max() }

Public Member Functions

- Blade ()=default
- template<typename _RHS_Blade >
  Blade (const _RHS_Blade &rhs)
- Blade (TYPE t, const _ValueType &f)
- bool operator== (const Blade &rhs) const
- bool sameType (const Blade &rhs) const
- int type2int () const
- void sortBladeTypes ()
- std::string getCompleteType () const
- template<typename _RHS_Blade >
  Blade & operator= (const _RHS_Blade &rhs)

Static Public Member Functions

- static std::string TYPE_TO_STRING (const TYPE &type)

Public Attributes

- TYPE type
- _ValueType factor
- std::vector< TYPE > outerProduct

Friends

- std::ostream & operator<< (std::ostream &os, const Blade &blade)

7.3.1 Member Enumeration Documentation

7.3.1.1 TYPE

```
template<typename _ValueType>
enum cf::MultiVector::Blade::TYPE  [strong]
```

Enumerator

E1	
E2	
E3	
EINF	
E0	
VALUE	

7.3.2 Constructor & Destructor Documentation

7.3.2.1 Blade() [1/3]

```
template<typename _ValueType>
cf::MultiVector< _ValueType >::Blade::Blade ( )  [default]
```

7.3.2.2 Blade() [2/3]

```
template<typename _ValueType>
template<typename _RHS_Blade >
cf::MultiVector< _ValueType >::Blade::Blade (
             const _RHS_Blade & rhs )  [inline]
```

7.3.2.3 Blade() [3/3]

```
template<typename _ValueType>
cf::MultiVector< _ValueType >::Blade::Blade (
             TYPE t,
             const _ValueType & f )  [inline]
```

7.3.3 Member Function Documentation

7.3.3.1 getCompleteType()

```
template<typename _ValueType>
std::string cf::MultiVector< _ValueType >::Blade::getCompleteType ( ) const  [inline]
```

7.3.3.2 operator=()

```
template<typename _ValueType>
template<typename _RHS_Blade >
Blade& cf::MultiVector< _ValueType >::Blade::operator= (
             const _RHS_Blade & rhs )  [inline]
```

7.3.3.3 operator==()

```
template<typename _ValueType>
bool cf::MultiVector< _ValueType >::Blade::operator== (
             const Blade & rhs ) const  [inline]
```

7.3.3.4 sameType()

```
template<typename _ValueType>
bool cf::MultiVector< _ValueType >::Blade::sameType (
             const Blade & rhs ) const  [inline]
```

7.3.3.5 sortBladeTypes()

```
template<typename _ValueType>
void cf::MultiVector< _ValueType >::Blade::sortBladeTypes ( )  [inline]
```

7.3.3.6 type2int()

```
template<typename _ValueType>
int cf::MultiVector< _ValueType >::Blade::type2int ( ) const  [inline]
```

7.3.3.7 TYPE_TO_STRING()

```
template<typename _ValueType>
static std::string cf::MultiVector< _ValueType >::Blade::TYPE_TO_STRING (
             const TYPE & type )  [inline], [static]
```

7.3.4 Friends And Related Function Documentation

7.3.4.1 operator<<

```
template<typename _ValueType>
std::ostream& operator<< (
             std::ostream & os,
             const Blade & blade )  [friend]
```

7.3.5 Member Data Documentation

7.3.5.1 factor

```
template<typename _ValueType>
_ValueType cf::MultiVector< _ValueType >::Blade::factor
```

7.3.5.2 outerProduct

```
template<typename _ValueType>
std::vector<TYPE> cf::MultiVector< _ValueType >::Blade::outerProduct
```

7.3.5.3 type

```
template<typename _ValueType>
TYPE cf::MultiVector< _ValueType >::Blade::type
```

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

- include/computerGeometry3D.hpp

7.4 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

7.4.1 Detailed Description

The Circle struct Simple parameter wrapper struct.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 Circle()

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

7.4.3 Member Data Documentation

7.4.3.1 center

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

7.4.3.2 color

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

7.4.3.3 lineWidth

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

7.4.3.4 radius

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

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

- include/window2D.h

7.5 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

7.5.1 Detailed Description

The CirclePartition struct Simple parameter wrapper struct.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 CirclePartition()

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

7.5.3 Member Data Documentation

7.5.3.1 center

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

7.5.3.2 color

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

7.5.3.3 endAngle

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

7.5.3.4 lineWidth

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

7.5.3.5 radius

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

7.5.3.6 startAngle

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

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

- include/window2D.h

7.6 cf::Color Struct Reference

The Color struct offers a class for rgb access.

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

Classes

- struct SimpleEndlessIterator

Public Member Functions

- Color (uint8_t red=0, uint8_t green=0, uint8_t blue=0)
- cf::Color operator∗ (float value) const
- cf::Color operator/ (float value) const
- cf::Color & operator∗= (float value)
- cf::Color & operator/= (float value)
- cf::Color operator+ (const Color &c) const
- cf::Color operator- (const Color &c) const
- cf::Color & operator+= (const Color &c)
- cf::Color & operator-= (const Color &c)
- bool operator== (const cf::Color &c) const
- bool operator!= (const cf::Color &c) const
- bool operator< (const cf::Color &c) const
- bool operator> (const cf::Color &c) const
- bool operator<= (const cf::Color &c) const
- bool operator>= (const cf::Color &c) const
- cf::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.

- template<typename... _Colors>
  static SimpleEndlessIterator< sizeof...(_Colors)> CreateEndlessColorIterator (_Colors &&... colors)

  CreateEndlessColorIterator creates an iterator, which cycles through alls provided colors.

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 cf::Color &c)

7.6.1 Detailed Description

The Color struct offers a class for rgb access.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 Color()

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

7.6.3 Member Function Documentation

7.6.3.1 CreateEndlessColorIterator()

```
template<typename...  _Colors>
static SimpleEndlessIterator<sizeof...(_Colors)> cf::Color::CreateEndlessColorIterator (
             _Colors &&...  colors )  [inline], [static]
```

CreateEndlessColorIterator creates an iterator, which cycles through alls provided colors.

Parameters

colors	All colors

Returns

Iterator

7.6.3.2 invert()

```
cf::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

7.6.3.3 operator"!=()

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

7.6.3.4 operator*()

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

7.6.3.5 operator*=()

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

7.6.3.6 operator+()

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

7.6.3.7 operator+=()

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

7.6.3.8 operator-()

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

7.6.3.9 operator-=()

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

7.6.3.10 operator/()

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

7.6.3.11 operator/=()

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

7.6.3.12 operator<()

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

7.6.3.13 operator<=()

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

7.6.3.14 operator==()

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

7.6.3.15 operator>()

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

7.6.3.16 operator>=()

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

7.6.3.17 RandomColor()

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

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

Returns

Random cf::Color

7.6.4 Friends And Related Function Documentation

7.6.4.1 operator*

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

7.6.4.2 operator/

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

7.6.4.3 operator<<

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

7.6.5 Member Data Documentation

7.6.5.1 b

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

7.6.5.2 BLACK

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

7.6.5.3 BLUE

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

7.6.5.4 CYAN

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

7.6.5.5 g

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

7.6.5.6 GREEN

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

7.6.5.7 GREY

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

7.6.5.8 MAGENTA

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

7.6.5.9 ORANGE

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

7.6.5.10 PINK

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

7.6.5.11 r

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

7.6.5.12 RED

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

7.6.5.13 WHITE

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

7.6.5.14 YELLOW

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

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

- include/utils.h

7.7 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
- template<typename... Args>
  static void printWarning (const Args &... args)

    Simple function for console warnings.
- template<typename... Args>
  static void printError (const Args &... args)

    Simple function for console error messages.

7.7.1 Detailed Description

The Console struct offers utility functions for 'console'.

7.7.2 Member Function Documentation

7.7.2.1 clearConsole()

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

clearConsole Clears the console

7.7.2.2 printError()

```
template<typename... Args>
static void cf::Console::printError (
             const Args &... args ) [inline], [static]
```

Simple function for console error messages.

7.7.2.3 printWarning()

```
template<typename... Args>
static void cf::Console::printWarning (
             const Args &... args ) [inline], [static]
```

Simple function for console warnings.

7.7.2.4 readFloat()

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

readFloat Reads a floatingpoint value

Returns

Read value

7.7.2.5 readInt()

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

readInt Reads a integer value

Returns

Read value

7.7.2.6 readString()

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

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

Returns

Read line

7.7.2.7 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

7.8 cf::Direction Struct Reference

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

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

Public Types

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

Static Public Member Functions

- static AbsoluteDirection getNextiDirection (AbsoluteDirection currentDirection, RelativeDirection relative↩ Movement)

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

7.8.1 Detailed Description

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

7.8.2 Member Enumeration Documentation

7.8.2.1 AbsoluteDirection

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

Enumerator

NORTH	
EAST	
SOUTH	
WEST	
NUM_ABS_DIRS	

7.8.2.2 RelativeDirection

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

Enumerator

LEFT	
FORWARD	
RIGHT	
NUM_REL_DIRS	

7.8.3 Member Function Documentation

7.8.3.1 getNextiDirection()

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

getNextiDirection receive absolute direction by providing a relative directon

Parameters

currentDirection	current absolute direction
relativeMovement	relative movement

Returns

7.8.3.2 toString() [1/2]

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

7.8.3.3 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

7.9 cf::Interval Struct Reference

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

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

Public Member Functions

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

7.9.1 Detailed Description

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

7.9.2 Constructor & Destructor Documentation

7.9.2.1 Interval()

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

7.9.3 Member Function Documentation

7.9.3.1 translateIntervalPostion() [1/2]

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

7.9.3.2 translateIntervalPostion() [2/2]

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

7.9.4 Friends And Related Function Documentation

7.9.4.1 operator<<

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

7.9.5 Member Data Documentation

7.9.5.1 max

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

7.9.5.2 min

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

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

- include/utils.h

7.10 cf::IteratedFunctionSystem Struct Reference

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

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

Public Member Functions

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

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

7.10.1 Detailed Description

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

7.10.2 Member Function Documentation

7.10.2.1 getAllTransformation()

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

7.10.2.2 getName()

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

7.10.2.3 getNumTransformations()

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

7.10.2.4 getRangeX()

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

7.10.2.5 getRangeY()

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

7.10.2.6 getTransformation()

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

7.10.2.7 read()

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

read a ∗.ifs file from path

Parameters

fiilePath	Path to a ∗.ifs file

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

- include/IFS.h

7.11 cf::LSystem_Controller::iterator Struct Reference

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

Public Member Functions

- iterator ()=default
- const char & operator∗ ()
- iterator & operator++ ()
- bool operator!= (const iterator &rhs)

Friends

- struct LSystem_Controller

7.11.1 Constructor & Destructor Documentation

7.11.1.1 iterator()

```
cf::LSystem_Controller::iterator::iterator ( )  [default]
```

7.11.2 Member Function Documentation

7.11.2.1 operator"!=()

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

7.11.2.2 operator∗()

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

7.11.2.3 operator++()

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

7.11.3 Friends And Related Function Documentation

7.11.3.1 LSystem_Controller

```
friend struct LSystem_Controller  [friend]
```

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

- include/LSystem.h

7.12 cf::LindenmayerSystem Struct Reference

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

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

Public Member Functions

- void read (const std::string &filePath)
  read a ∗.lin file from path
- const std::string & getName () const
- const std::string & 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

7.12.1 Detailed Description

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

7.12.2 Member Function Documentation

7.12.2.1 clearWindowEachTime()

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

7.12.2.2 getAdjustmentAngle()

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

7.12.2.3 getAllProductions()

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

7.12.2.4 getAxiom()

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

7.12.2.5 getName()

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

7.12.2.6 getNumProductions()

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

7.12.2.7 getProduction()

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

7.12.2.8 getRangeX()

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

7.12.2.9 getRangeY()

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

7.12.2.10 getScale()

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

7.12.2.11 getStartAngle()

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

7.12.2.12 read()

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

read a ∗.lin file from path

Parameters

filePath	Path to a ∗.lin file

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

- include/LSystem.h

7.13 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

7.13.1 Detailed Description

The Line struct Simple parameter wrapper struct.

7.13.2 Constructor & Destructor Documentation

7.13.2.1 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]
```

7.13.3 Member Data Documentation

7.13.3.1 color

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

7.13.3.2 lineType

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

7.13.3.3 lineWidth

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

7.13.3.4 point1

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

7.13.3.5 point2

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

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

- include/window2D.h

7.14 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 () const
- iterator end () const

7.14.1 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;
```

7.14.2 Constructor & Destructor Documentation

7.14.2.1 LSystem_Controller()

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

7.14.3 Member Function Documentation

7.14.3.1 begin()

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

7.14.3.2 end()

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

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

- include/LSystem.h

7.15 cf::MultiVector< _ValueType > Struct Template Reference

```
#include <computerGeometry3D.hpp>
```

Classes

- struct Blade

Public Member Functions

- MultiVector ()=default
- template<typename _VType >
  MultiVector (const MultiVector< _VType > &vec)
- template<typename... _Blades>
  MultiVector (const _Blades &... blades)
- void setData (const std::vector< Blade > &data)
- const std::vector< Blade > & getData () const
- template<typename _VType >
  MultiVector< _ValueType > & operator= (const MultiVector< _VType > &rhs)
- MultiVector< _ValueType > & operator+ () const
- MultiVector< _ValueType > & operator- ()
- MultiVector< _ValueType > operator* () const
- MultiVector< _ValueType > operator~ () const
- template<typename _VType >
  operator _VType () const
- template<typename _VType >
  MultiVector< decltype(_ValueType(1)/_VType(1))> operator/ (const _VType &value) const
- template<typename _VType >
  MultiVector< decltype(_ValueType(1)/_VType(1))> & operator/= (const _VType &value)
- template<typename _VType >
  MultiVector< _ValueType > operator+ (const _VType &value) const
- template<typename _VType >
  MultiVector< _ValueType > & operator+= (const _VType &value)
- template<typename _VType >
  MultiVector< _ValueType > operator- (const _VType &value) const
- template<typename _VType >
  MultiVector< _ValueType > & operator-= (const _VType &value)
- template<typename _VType >
  MultiVector< decltype(_ValueType(1)+_VType(1))> operator+ (const MultiVector< _VType > &rhs) const
- template<typename _VType >
  MultiVector< _ValueType > & operator+= (const MultiVector< _VType > &rhs)
- template<typename _VType >
  MultiVector< decltype(_ValueType(1) *_VType(1))> operator- (const MultiVector< _VType > &rhs) const
- template<typename _VType >
  MultiVector< _ValueType > & operator-= (const MultiVector< _VType > &rhs)
- MultiVector< _ValueType > & operator*= (const _ValueType &rhs)
- MultiVector< _ValueType > operator* (const _ValueType &rhs) const
- template<typename _VType >
  MultiVector< decltype(_ValueType(1) *_VType(1))> operator* (const MultiVector< _VType > &rhs) const
- template<typename _VType >
  MultiVector< _ValueType > & operator*= (const MultiVector< _VType > &rhs)
- template<typename _VType >
  MultiVector< decltype(_ValueType(1) *_VType(1))> operator% (const MultiVector< _VType > &rhs) const
- template<typename _VType >
  MultiVector< _ValueType > & operator%= (const MultiVector< _VType > &rhs)
- template<typename _VType >
  MultiVector< decltype(_ValueType(1) *_VType(1))> operator^ (const MultiVector< _VType > &rhs) const
- template<typename _VType >
  MultiVector< _ValueType > & operator^= (const MultiVector< _VType > &rhs)
- template<typename _VType >
  MultiVector< decltype(_ValueType(1) *_VType(1))> operator & (const MultiVector< _VType > &rhs) const
- template<typename _VType >
  MultiVector< decltype(_ValueType(1) *_VType(1))> operator &= (const MultiVector< _VType > &rhs)
- bool operator== (const MultiVector< _ValueType > &rhs) const

Friends

- template<typename _VType >
  struct MultiVector
- std::ostream & operator<< (std::ostream &os, const MultiVector< _ValueType > &vec)

7.15.1 Detailed Description

template<typename _ValueType>
struct cf::MultiVector< _ValueType >

TODO operators and value in front

7.15.2 Constructor & Destructor Documentation

7.15.2.1 MultiVector() [1/3]

```
template<typename _ValueType>
cf::MultiVector< _ValueType >::MultiVector ( )  [default]
```

7.15.2.2 MultiVector() [2/3]

```
template<typename _ValueType>
template<typename _VType >
cf::MultiVector< _ValueType >::MultiVector (
             const MultiVector< _VType > & vec )  [inline]
```

7.15.2.3 MultiVector() [3/3]

```
template<typename _ValueType>
template<typename...  _Blades>
cf::MultiVector< _ValueType >::MultiVector (
             const _Blades &...  blades )  [inline]
```

7.15.3 Member Function Documentation

7.15.3.1 getData()

```
template<typename _ValueType>
const std::vector<Blade>& cf::MultiVector< _ValueType >::getData ( ) const  [inline]
```

7.15.3.2 operator &()

```
template<typename _ValueType>
template<typename _VType >
MultiVector<decltype(_ValueType(1) * _VType(1))> cf::MultiVector< _ValueType >::operator& (
             const MultiVector< _VType > & rhs ) const  [inline]
```

7.15.3.3 operator &=()

```
template<typename _ValueType>
template<typename _VType >
MultiVector<decltype(_ValueType(1) * _VType(1))> cf::MultiVector< _ValueType >::operator&= (
             const MultiVector< _VType > & rhs )  [inline]
```

7.15.3.4 operator _VType()

```
template<typename _ValueType>
template<typename _VType >
cf::MultiVector< _ValueType >::operator _VType ( ) const  [inline], [explicit]
```

7.15.3.5 operator%()

```
template<typename _ValueType>
template<typename _VType >
MultiVector<decltype(_ValueType(1) * _VType(1))> cf::MultiVector< _ValueType >::operator% (
             const MultiVector< _VType > & rhs ) const  [inline]
```

7.15.3.6 operator%=()

```
template<typename _ValueType>
template<typename _VType >
MultiVector<_ValueType>& cf::MultiVector< _ValueType >::operator%= (
             const MultiVector< _VType > & rhs )  [inline]
```

7.15.3.7 operator*() [1/3]

```
template<typename _ValueType>
MultiVector<_ValueType> cf::MultiVector< _ValueType >::operator* ( ) const  [inline]
```

7.15.3.8 operator*() [2/3]

```
template<typename _ValueType>
MultiVector<_ValueType> cf::MultiVector< _ValueType >::operator* (
            const _ValueType & rhs ) const  [inline]
```

7.15.3.9 operator*() [3/3]

```
template<typename _ValueType>
template<typename _VType >
MultiVector<decltype(_ValueType(1) * _VType(1))> cf::MultiVector< _ValueType >::operator* (
            const MultiVector< _VType > & rhs ) const  [inline]
```

7.15.3.10 operator*=() [1/2]

```
template<typename _ValueType>
MultiVector<_ValueType>& cf::MultiVector< _ValueType >::operator*= (
            const _ValueType & rhs )  [inline]
```

7.15.3.11 operator*=() [2/2]

```
template<typename _ValueType>
template<typename _VType >
MultiVector<_ValueType>& cf::MultiVector< _ValueType >::operator*= (
            const MultiVector< _VType > & rhs )  [inline]
```

7.15.3.12 operator+() [1/3]

```
template<typename _ValueType>
MultiVector<_ValueType>& cf::MultiVector< _ValueType >::operator+ ( ) const  [inline]
```

7.15.3.13 operator+() [2/3]

```
template<typename _ValueType>
template<typename _VType >
MultiVector<_ValueType> cf::MultiVector< _ValueType >::operator+ (
             const _VType & value ) const  [inline]
```

7.15.3.14 operator+() [3/3]

```
template<typename _ValueType>
template<typename _VType >
MultiVector<decltype(_ValueType(1) + _VType(1))> cf::MultiVector< _ValueType >::operator+ (
             const MultiVector< _VType > & rhs ) const  [inline]
```

7.15.3.15 operator+=() [1/2]

```
template<typename _ValueType>
template<typename _VType >
MultiVector<_ValueType>& cf::MultiVector< _ValueType >::operator+= (
             const _VType & value )  [inline]
```

7.15.3.16 operator+=() [2/2]

```
template<typename _ValueType>
template<typename _VType >
MultiVector<_ValueType>& cf::MultiVector< _ValueType >::operator+= (
             const MultiVector< _VType > & rhs )  [inline]
```

7.15.3.17 operator-() [1/3]

```
template<typename _ValueType>
MultiVector<_ValueType>& cf::MultiVector< _ValueType >::operator- ( )  [inline]
```

7.15.3.18 operator-() [2/3]

```
template<typename _ValueType>
template<typename _VType >
MultiVector<_ValueType> cf::MultiVector< _ValueType >::operator- (
             const _VType & value ) const  [inline]
```

7.15.3.19 operator-() [3/3]

```
template<typename _ValueType>
template<typename _VType >
MultiVector<decltype(_ValueType(1) * _VType(1))> cf::MultiVector< _ValueType >::operator- (
             const MultiVector< _VType > & rhs ) const  [inline]
```

7.15.3.20 operator-=() [1/2]

```
template<typename _ValueType>
template<typename _VType >
MultiVector<_ValueType>& cf::MultiVector< _ValueType >::operator-= (
             const _VType & value )  [inline]
```

7.15.3.21 operator-=() [2/2]

```
template<typename _ValueType>
template<typename _VType >
MultiVector<_ValueType>& cf::MultiVector< _ValueType >::operator-= (
             const MultiVector< _VType > & rhs )  [inline]
```

7.15.3.22 operator/()

```
template<typename _ValueType>
template<typename _VType >
MultiVector<decltype(_ValueType(1) / _VType(1))> cf::MultiVector< _ValueType >::operator/ (
             const _VType & value ) const  [inline]
```

7.15.3.23 operator/=()

```
template<typename _ValueType>
template<typename _VType >
MultiVector<decltype(_ValueType(1) / _VType(1))>& cf::MultiVector< _ValueType >::operator/= (
             const _VType & value )  [inline]
```

7.15.3.24 operator=()

```
template<typename _ValueType>
template<typename _VType >
MultiVector<_ValueType>& cf::MultiVector< _ValueType >::operator= (
             const MultiVector< _VType > & rhs )  [inline]
```

7.15.3.25 operator==()

```
template<typename _ValueType>
bool cf::MultiVector< _ValueType >::operator== (
             const MultiVector< _ValueType > & rhs ) const  [inline]
```

7.15.3.26 operator^()

```
template<typename _ValueType>
template<typename _VType >
MultiVector<decltype(_ValueType(1) * _VType(1))> cf::MultiVector< _ValueType >::operator^ (
             const MultiVector< _VType > & rhs ) const  [inline]
```

7.15.3.27 operator^=()

```
template<typename _ValueType>
template<typename _VType >
MultiVector<_ValueType>& cf::MultiVector< _ValueType >::operator^= (
             const MultiVector< _VType > & rhs )  [inline]
```

7.15.3.28 operator~()

```
template<typename _ValueType>
MultiVector<_ValueType> cf::MultiVector< _ValueType >::operator~ ( ) const  [inline]
```

7.15.3.29 setData()

```
template<typename _ValueType>
void cf::MultiVector< _ValueType >::setData (
             const std::vector< Blade > & data )  [inline]
```

7.15.4 Friends And Related Function Documentation

7.15.4.1 MultiVector

```
template<typename _ValueType>
template<typename _VType >
friend struct MultiVector  [friend]
```

7.15.4.2 operator<<

```
template<typename _ValueType>
std::ostream& operator<< (
            std::ostream & os,
            const MultiVector< _ValueType > & vec )  [friend]
```

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

- include/computerGeometry3D.hpp

7.16 cf::Orbit Struct Reference

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

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

Public Member Functions

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

    read a ∗.orb file from path
- const cf::Interval & getRangeX () const
- const cf::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

7.16.1 Detailed Description

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

7.16.2 Member Function Documentation

7.16.2.1 getAllFactors()

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

7.16.2.2 getAllStartingPoints()

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

7.16.2.3 getName()

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

7.16.2.4 getNumFactors()

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

7.16.2.5 getNumStartingPoints()

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

7.16.2.6 getRangeX()

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

7.16.2.7 getRangeY()

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

7.16.2.8 read()

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

read a ∗.orb file from path

Parameters

filePath	Path to a ∗.orb file

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

- include/ORB.h

7.17 cf::Point Struct Reference

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

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

Public Member Functions

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

Public Attributes

- float x
- float y

Friends

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

7.17.1 Detailed Description

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

7.17.2 Constructor & Destructor Documentation

7.17.2.1 Point()

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

7.17.3 Member Function Documentation

7.17.3.1 operator cv::Point()

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

7.17.3.2 operator"!=()

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

7.17.3.3 operator∗()

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

7.17.3.4 operator∗=()

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

7.17.3.5 operator+()

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

7.17.3.6 operator+=()

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

7.17.3.7 operator-()

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

7.17.3.8 operator-=()

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

7.17.3.9 operator/()

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

7.17.3.10 operator/=()

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

7.17.3.11 operator==()

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

7.17.4 Friends And Related Function Documentation

7.17.4.1 operator*

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

7.17.4.2 operator/

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

7.17.5 Member Data Documentation

7.17.5.1 x

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

7.17.5.2 y

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

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

- include/window2D.h

7.18 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

7.18.1 Detailed Description

The Rect struct Simple parameter wrapper struct.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 Rect()

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

7.18.3 Member Data Documentation

7.18.3.1 color

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

7.18.3.2 lineWidth

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

7.18.3.3 point1

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

7.18.3.4 point2

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

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

- include/window2D.h

7.19 cf::Color::SimpleEndlessIterator< _Size > Struct Template Reference

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

Public Member Functions

- void operator++ ()
- void operator++ (int)
- const cf::Color & operator* () const
- const cf::Color & operator-> () const
- SimpleEndlessIterator (SimpleEndlessIterator &&)=default

7.19.1 Constructor & Destructor Documentation

7.19.1.1 SimpleEndlessIterator()

```
template<int _Size>
cf::Color::SimpleEndlessIterator< _Size >::SimpleEndlessIterator (
            SimpleEndlessIterator< _Size > &&  )  [default]
```

7.19.2 Member Function Documentation

7.19.2.1 operator*()

```
template<int _Size>
const cf::Color& cf::Color::SimpleEndlessIterator< _Size >::operator* ( ) const  [inline]
```

7.19.2.2 operator++() [1/2]

```
template<int _Size>
void cf::Color::SimpleEndlessIterator< _Size >::operator++ ( )  [inline]
```

7.19.2.3 operator++() [2/2]

```
template<int _Size>
void cf::Color::SimpleEndlessIterator< _Size >::operator++ (
             int  )  [inline]
```

7.19.2.4 operator->()

```
template<int _Size>
const cf::Color& cf::Color::SimpleEndlessIterator< _Size >::operator-> ( ) const  [inline]
```

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

- include/utils.h

7.20 cf::SimpleSignal Struct Reference

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

Public Member Functions

- void waitSignal ()
- void fireSignal ()

7.20.1 Member Function Documentation

7.20.1.1 fireSignal()

```
void cf::SimpleSignal::fireSignal ( )
```

7.20.1.2 waitSignal()

```
void cf::SimpleSignal::waitSignal ( )
```

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

- include/utils.h

7.21 cf::Vec3< IS_POINTVECTOR, _ValueType > Class Template Reference

The Vec3 struct General class for vector operations.

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

Public Types

- typedef Vec3< IS_POINTVECTOR, _ValueType > self_type
- typedef _ValueType value_type

Public Member Functions

- Vec3 (const _ValueType &x=0.0, const _ValueType &y=0.0)
- Vec3 (const _ValueType &x, const _ValueType &y, const _ValueType &w)
- Vec3 (const cf::Point &p)
- template< bool PV_RHS, typename _VType >
  Vec3< PV_RHS|IS_POINTVECTOR, decltype(_ValueType(0)+_VType(0))> operator+ (const Vec3< PV_↩
  RHS, _VType > &rhs) const
- template< bool PV_RHS, typename _VType >
  self_type & operator+= (const Vec3< PV_RHS, _VType > &rhs)
- template< bool PV_RHS, typename _VType >
  Vec3< PV_RHS|IS_POINTVECTOR, decltype(_ValueType(0) - _VType(0))> operator- (const Vec3< PV_↩
  RHS, _VType > &rhs) const
- template< bool PV_RHS, typename _VType >
  self_type & operator-= (const Vec3< PV_RHS, _VType > &rhs)
- self_type operator∗ (const _ValueType &rhs) const

  operator∗ Multiplys each component of the vector with a factor
- self_type & operator∗= (const _ValueType &rhs)
- template< bool PV_RHS, typename _VType >
  Vec3< PV_RHS|IS_POINTVECTOR, decltype(_ValueType(0) ∗_ValueType(0) - _ValueType(0))> operator%
  (const Vec3< PV_RHS, _VType > &rhs) const

  operator% Performs the cross product between two vectors
- template< bool PV_RHS, typename _VType >
  self_type & operator%= (const Vec3< PV_RHS, _VType > &rhs)
- self_type & 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 PV_RHS, typename _VType >
  decltype(_VType(0) ∗_ValueType(0) ∗(_VType(0)+_ValueType(0))) operator∗ (const Vec3< PV_RHS, _VType
  > &rhs) const

operator∗ Performs the dot product between two vectors

- const _ValueType & getX () const

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

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

  getW Read access to component 'w'
- void setX (const _ValueType &value)

  setX Write to component 'x'
- void setY (const _ValueType &value)

  setY Write to component 'y'
- void setW (const _ValueType &value)

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

  operator[] Access to each component of the Vector, Note: read access is granted to all components (including index 2)
- _ValueType & 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 glmVec3 & () const
- operator cf::Point () const

  operator cf::Point Conversion operator to cf::Point, compile error on DirectionVectors
- self_type & operator= (const cf::Point &p)
- template< typename _VType , glm::precision precision>
  self_type & operator= (const glm::tvec3< _VType, precision > &rhs)
- template< bool PV_RHS, typename _VType >
  operator cf::Vec3< PV_RHS, _VType > () const

  Conversion operator from point vector to direction vector and vise versa, may throw an exception if 'w' is not 0 (point to direction vector)
- decltype(_ValueType(0) ∗_ValueType(0)+_ValueType(0)) length () const

  length Calculates the vector length for Direction type vectors
- self_type getVector90Degree () const

  getVector90Degree A vector that that has an angle of 90 degree from the original vector (only available for direction type vectors)
- bool operator== (const self_type &rhs) const

  operator== Equals operator
- bool operator!= (const self_type &rhs) const

  operator!= Not equals operator

Friends

- template< bool b, typename _VType >
  class Vec3
- self_type operator∗ (const _ValueType &lhs, const self_type &vec)
- template< bool b, typename _VType >
  std::ostream & operator<<) (std::ostream &, const Vec3< b, _VType > &)

7.21.1 Detailed Description

template<bool IS_POINTVECTOR, typename _ValueType>
class cf::Vec3< IS_POINTVECTOR, _ValueType >

The Vec3 struct General class for vector operations.

it porvides:

- 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')

7.21.2 Member Typedef Documentation

7.21.2.1 self_type

```
template<bool IS_POINTVECTOR, typename _ValueType >
typedef Vec3<IS_POINTVECTOR, _ValueType> cf::Vec3< IS_POINTVECTOR, _ValueType >::self_type
```

7.21.2.2 value_type

```
template<bool IS_POINTVECTOR, typename _ValueType >
typedef _ValueType cf::Vec3< IS_POINTVECTOR, _ValueType >::value_type
```

7.21.3 Constructor & Destructor Documentation

7.21.3.1 Vec3() [1/3]

```
template<bool IS_POINTVECTOR, typename _ValueType >
cf::Vec3< IS_POINTVECTOR, _ValueType >::Vec3 (
            const _ValueType & x = 0.0,
            const _ValueType & y = 0.0 )  [inline]
```

7.21.3.2 Vec3() [2/3]

```
template<bool IS_POINTVECTOR, typename _ValueType >
cf::Vec3< IS_POINTVECTOR, _ValueType >::Vec3 (
            const _ValueType & x,
            const _ValueType & y,
            const _ValueType & w )  [inline]
```

7.21.3.3 Vec3() [3/3]

```
template<bool IS_POINTVECTOR, typename _ValueType >
cf::Vec3< IS_POINTVECTOR, _ValueType >::Vec3 (
            const cf::Point & p )  [inline]
```

7.21.4 Member Function Documentation

7.21.4.1 getVector90Degree()

```
template<bool IS_POINTVECTOR, typename _ValueType >
self_type cf::Vec3< IS_POINTVECTOR, _ValueType >::getVector90Degree ( ) const  [inline]
```

getVector90Degree A vector that that has an angle of 90 degree from the original vector (only available for direction type vectors)

Returns

7.21.4.2 getW()

```
template<bool IS_POINTVECTOR, typename _ValueType >
const _ValueType& cf::Vec3< IS_POINTVECTOR, _ValueType >::getW ( ) const  [inline]
```

getW Read access to component 'w'

Returns

7.21.4.3 getX()

```
template<bool IS_POINTVECTOR, typename _ValueType >
const _ValueType& cf::Vec3< IS_POINTVECTOR, _ValueType >::getX ( ) const  [inline]
```

getX Read access to component 'x'

Returns

7.21.4.4 getY()

```
template<bool IS_POINTVECTOR, typename _ValueType >
const _ValueType& cf::Vec3< IS_POINTVECTOR, _ValueType >::getY ( ) const  [inline]
```

getY Read access to component 'y'

Returns

7.21.4.5 isPointVector()

```
template<bool IS_POINTVECTOR, typename _ValueType >
bool cf::Vec3< IS_POINTVECTOR, _ValueType >::isPointVector ( ) const  [inline]
```

isPointVector Checks wether a Vector is a PointVector or DirectionVector

Returns

7.21.4.6 length()

```
template<bool IS_POINTVECTOR, typename _ValueType >
decltype(_ValueType(0) * _ValueType(0) + _ValueType(0)) cf::Vec3< IS_POINTVECTOR, _ValueType
>::length ( ) const  [inline]
```

length Calculates the vector length for Direction type vectors

Returns

Length of the underlying vector

7.21.4.7 normalize()

```
template<bool IS_POINTVECTOR, typename _ValueType >
self_type& cf::Vec3< IS_POINTVECTOR, _ValueType >::normalize ( )  [inline]
```

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

Returns

Return the normalized vector

7.21.4.8 operator cf::Point()

```
template<bool IS_POINTVECTOR, typename _ValueType >
cf::Vec3< IS_POINTVECTOR, _ValueType >::operator cf::Point ( ) const  [inline]
```

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

7.21.4.9 operator cf::Vec3< PV_RHS, _VType >()

```
template<bool IS_POINTVECTOR, typename _ValueType >
template<bool PV_RHS, typename _VType >
cf::Vec3< IS_POINTVECTOR, _ValueType >::operator cf::Vec3< PV_RHS, _VType > ( ) const  [inline]
```

Conversion operator from point vector to direction vector and vise versa, may throw an exception if 'w' is not 0 (point to direction vector)

7.21.4.10 operator const glmVec3 &()

```
template<bool IS_POINTVECTOR, typename _ValueType >
cf::Vec3< IS_POINTVECTOR, _ValueType >::operator const glmVec3 & ( ) const  [inline]
```

7.21.4.11 operator glm::vec3()

```
template<bool IS_POINTVECTOR, typename _ValueType >
cf::Vec3< IS_POINTVECTOR, _ValueType >::operator glm::vec3 ( ) const  [inline]
```

7.21.4.12 operator"!=()

```
template<bool IS_POINTVECTOR, typename _ValueType >
bool cf::Vec3< IS_POINTVECTOR, _ValueType >::operator!= (
             const self_type & rhs ) const  [inline]
```

operator!= Not equals operator

Parameters

rhs	Other vector

Returns

7.21.4.13 operator%()

```
template<bool IS_POINTVECTOR, typename _ValueType >
template<bool PV_RHS, typename _VType >
Vec3<PV_RHS | IS_POINTVECTOR, decltype(_ValueType(0) * _ValueType(0) - _ValueType(0))> cf::Vec3<
IS_POINTVECTOR, _ValueType >::operator% (
             const Vec3< PV_RHS, _VType > & rhs ) const  [inline]
```

operator% Performs the cross product between two vectors

Parameters

rhs	Second operand for cross product

Returns

7.21.4.14 operator%=()

```
template<bool IS_POINTVECTOR, typename _ValueType >
template<bool PV_RHS, typename _VType >
self_type& cf::Vec3< IS_POINTVECTOR, _ValueType >::operator%= (
             const Vec3< PV_RHS, _VType > & rhs )  [inline]
```

7.21.4.15 operator∗() [1/2]

```
template<bool IS_POINTVECTOR, typename _ValueType >
self_type cf::Vec3< IS_POINTVECTOR, _ValueType >::operator* (
             const _ValueType & rhs ) const  [inline]
```

operator∗ Multiplys each component of the vector with a factor

Parameters

rhs	Factor for the multiplication

Returns

Multiplied vector

7.21.4.16 operator∗() [2/2]

```
template<bool IS_POINTVECTOR, typename _ValueType >
template<bool PV_RHS, typename _VType >
decltype(_VType(0) * _ValueType(0) * (_VType(0) + _ValueType(0))) cf::Vec3< IS_POINTVECTOR, ↩
_ValueType >::operator* (
             const Vec3< PV_RHS, _VType > & rhs ) const  [inline]
```

operator∗ Performs the dot product between two vectors

Parameters

rhs	Second operand for dot product

Returns

7.21.4.17 operator∗=()

```
template<bool IS_POINTVECTOR, typename _ValueType >
self_type& cf::Vec3< IS_POINTVECTOR, _ValueType >::operator*= (
             const _ValueType & rhs )  [inline]
```

7.21.4.18 operator+()

```
template<bool IS_POINTVECTOR, typename _ValueType >
template<bool PV_RHS, typename _VType >
Vec3<PV_RHS | IS_POINTVECTOR, decltype(_ValueType(0) + _VType(0))> cf::Vec3< IS_POINTVECTOR,
_ValueType >::operator+ (
             const Vec3< PV_RHS, _VType > & rhs ) const  [inline]
```

7.21.4.19 operator+=()

```
template<bool IS_POINTVECTOR, typename _ValueType >
template<bool PV_RHS, typename _VType >
self_type& cf::Vec3< IS_POINTVECTOR, _ValueType >::operator+= (
             const Vec3< PV_RHS, _VType > & rhs )  [inline]
```

7.21.4.20 operator-()

```
template<bool IS_POINTVECTOR, typename _ValueType >
template<bool PV_RHS, typename _VType >
Vec3<PV_RHS | IS_POINTVECTOR, decltype(_ValueType(0) - _VType(0))> cf::Vec3< IS_POINTVECTOR,
_ValueType >::operator- (
             const Vec3< PV_RHS, _VType > & rhs ) const  [inline]
```

7.21.4.21 operator-=()

```
template<bool IS_POINTVECTOR, typename _ValueType >
template<bool PV_RHS, typename _VType >
self_type& cf::Vec3< IS_POINTVECTOR, _ValueType >::operator-= (
             const Vec3< PV_RHS, _VType > & rhs )  [inline]
```

7.21.4.22 operator=() [1/2]

```
template<bool IS_POINTVECTOR, typename _ValueType >
self_type& cf::Vec3< IS_POINTVECTOR, _ValueType >::operator= (
             const cf::Point & p )  [inline]
```

7.21.4.23 operator=() [2/2]

```
template<bool IS_POINTVECTOR, typename _ValueType >
template<typename _VType , glm::precision precision>
self_type& cf::Vec3< IS_POINTVECTOR, _ValueType >::operator= (
             const glm::tvec3< _VType, precision > & rhs )  [inline]
```

7.21.4.24 operator==()

```
template<bool IS_POINTVECTOR, typename _ValueType >
bool cf::Vec3< IS_POINTVECTOR, _ValueType >::operator== (
             const self_type & rhs ) const  [inline]
```

operator== Equals operator

Parameters

rhs	Other vector

Returns

7.21.4.25 operator[]() [1/2]

```
template<bool IS_POINTVECTOR, typename _ValueType >
const _ValueType& cf::Vec3< IS_POINTVECTOR, _ValueType >::operator[] (
             int idx ) const  [inline]
```

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

Parameters

idx	Acess index

Returns

7.21.4.26 operator[]() [2/2]

```
template<bool IS_POINTVECTOR, typename _ValueType >
_ValueType& cf::Vec3< IS_POINTVECTOR, _ValueType >::operator[] (
             int idx )  [inline]
```

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

Parameters

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

Returns

7.21.4.27 setW()

```
template<bool IS_POINTVECTOR, typename _ValueType >
void cf::Vec3< IS_POINTVECTOR, _ValueType >::setW (
             const _ValueType & value )  [inline]
```

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

Parameters

value	

7.21.4.28 setX()

```
template<bool IS_POINTVECTOR, typename _ValueType >
void cf::Vec3< IS_POINTVECTOR, _ValueType >::setX (
             const _ValueType & value )  [inline]
```

setX Write to component 'x'

Parameters

value	

7.21.4.29 setY()

```
template<bool IS_POINTVECTOR, typename _ValueType >
void cf::Vec3< IS_POINTVECTOR, _ValueType >::setY (
             const _ValueType & value )  [inline]
```

setY Write to component 'y'

Parameters

value	

7.21.5 Friends And Related Function Documentation

7.21.5.1 operator*

```
template<bool IS_POINTVECTOR, typename _ValueType >
self_type operator* (
             const _ValueType & lhs,
             const self_type & vec )  [friend]
```

7.21.5.2 operator<<)

```
template<bool IS_POINTVECTOR, typename _ValueType >
template<bool b, typename _VType >
std::ostream& operator<<) (
            std::ostream & ,
            const Vec3< b, _VType > &  )  [friend]
```

7.21.5.3 Vec3

```
template<bool IS_POINTVECTOR, typename _ValueType >
template<bool b, typename _VType >
friend class Vec3  [friend]
```

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

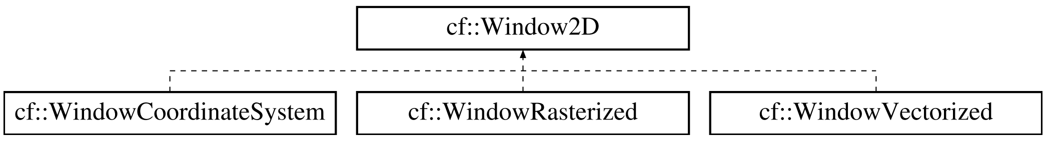
- include/computerGeometry.hpp

7.22 cf::Window2D Class Reference

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

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

Inheritance diagram for cf::Window2D:



Public Types

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

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

Public Member Functions

- Window2D (int width=800, int height=600, const std::string &windowName="Lab", const cf::Color &start↩ Color=cf::Color::BLACK)
- Window2D (const std::string &filePath)
- virtual ∼Window2D ()
- void show () const

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

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

  waitMouseInput Blocks until mouse input has been given
- cf::Point waitMouseInput ()

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

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

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

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

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

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

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

  setNewInterval Set new interval
- void resetInterval ()

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

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

  resize Resize underlying image
- void flippHorizontal ()

  flippHorizontal Flipp image horizontally
- void flippVertical ()

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

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

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

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

  getHeight Acess to underlying image height

- cv::Mat & getImage ()

  getImage Direct access to the underlying image
- void drawAxis (const cf::Color &color=cf::Color::BLACK, float stepSize_x=1.f, float stepSize_y=1.f, float interceptLength=3.f)

  drawAxis This function draws x and y axis based on Interval
- void 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 assigment 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

Static Protected Member Functions

- static std::string _CreateUniqueWindowName (const std::string &name)

Protected Attributes

- cv::Mat m_Image
- bool m_InvertYAxis
- const std::string 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

7.22.1 Detailed Description

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

7.22.2 Member Enumeration Documentation

7.22.2.1 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	

7.22.3 Constructor & Destructor Documentation

7.22.3.1 Window2D() [1/2]

```
cf::Window2D::Window2D (
            int width = 800,
            int height = 600,
            const std::string & windowName = "Lab",
            const cf::Color & startColor = cf::Color::BLACK )
```

7.22.3.2 Window2D() [2/2]

```
cf::Window2D::Window2D (
            const std::string & filePath )
```

7.22.3.3 ∼Window2D()

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

7.22.4 Member Function Documentation

7.22.4.1 _convertFromNewInterval()

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

7.22.4.2 _convertToNewInterval()

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

7.22.4.3 _correctYValue()

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

7.22.4.4 _CreateUniqueWindowName()

```
static std::string cf::Window2D::_CreateUniqueWindowName (
             const std::string & name )  [static], [protected]
```

7.22.4.5 _window2foreground()

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

7.22.4.6 clear()

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

7.22.4.7 drawAxis()

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

drawAxis This function draws x and y axis based on Interval

Parameters

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

7.22.4.8 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

point	Point within interval_x and interval_y
radius	Circle radius in pixel (not effected by intervals)
lineWidth	Pixelwidth of line (not effected by intervals), negative values fills the rectangle
color	Circle color

7.22.4.9 drawCircle() [2/2]

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

drawCircle Draws a circle from circle class

Parameters

circle	

7.22.4.10 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

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

7.22.4.11 drawCirclePart() [2/2]

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

drawCirclePart Draws a circlePartition from circlePartition class

Parameters

circlePartition	

7.22.4.12 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

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

7.22.4.13 drawLine() [2/2]

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

drawLine Draws a line from line class

Parameters

line	

7.22.4.14 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

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

7.22.4.15 drawRectangle() [2/2]

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

drawRectangle Draws a rect from rect class

Parameters

rect	

7.22.4.16 drawSpecializedLine()

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

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

Parameters

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

7.22.4.17 flippHorizontal()

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

flippHorizontal Flipp image horizontally

7.22.4.18 flippVertical()

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

flippHorizontal Flipp image vertically

7.22.4.19 floodFill()

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

floodFill Fills an area

Parameters

startingPoint	First point to be colored
color	Fill color

7.22.4.20 getColor()

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

7.22.4.21 getHeight()

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

getHeight Acess to underlying image height

Returns

Height

7.22.4.22 getImage()

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

getImage Direct access to the underlying image

Returns

Image handle

7.22.4.23 getIntervalX()

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

getIntervalX Const access to interval in x direction

Returns

7.22.4.24 getIntervalY()

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

getIntervalY Const access to interval in y direction

Returns

7.22.4.25 getInvertYAxis()

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

7.22.4.26 getWidth()

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

getWidth Acess to underlying image width

Returns

Width

7.22.4.27 getWindowDisplayScale()

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

7.22.4.28 operator=()

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

operator= Copy assigment operator

Parameters

rhs	Element to be copied

Returns

7.22.4.29 resetInterval()

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

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

7.22.4.30 resize()

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

resize Resize underlying image

Parameters

pixelWidth	New width
pixelHeight	New height

7.22.4.31 saveImage()

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

saveImage Saves current image to harddrive

Parameters

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

7.22.4.32 setColor()

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

7.22.4.33 setInvertYAxis()

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

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

Parameters

invert	

7.22.4.34 setNewInterval()

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

setNewInterval Set new interval

Parameters

intervalX	Interval in x direction
intervalY	Interval in y direction

7.22.4.35 setWindowDisplayScale()

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

setWindowDisplayScale Scales the image before displaying

Parameters

scale	Window scale size

7.22.4.36 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))

7.22.4.37 waitKey()

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

waitKey Block access until key input on window

Parameters

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

Returns

7.22.4.38 waitMouseInput() [1/2]

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

waitMouseInput Blocks until mouse input has been given

Parameters

x	X-Window position
y	Y-Window position

7.22.4.39 waitMouseInput() [2/2]

```
cf::Point cf::Window2D::waitMouseInput ( )
```

waitMouseInput Blocks until mouse input has been given

Returns

7.22.5 Member Data Documentation

7.22.5.1 m_FristShowCall

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

7.22.5.2 m_Image

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

7.22.5.3 m_IntervalChanged

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

7.22.5.4 m_IntervalX

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

7.22.5.5 m_IntervalY

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

7.22.5.6 m_InvertYAxis

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

7.22.5.7 m_MouseCallBackStorage

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

7.22.5.8 m_WindowName

```
const std::string cf::Window2D::m_WindowName  [protected]
```

7.22.5.9 m_WindowScale

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

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

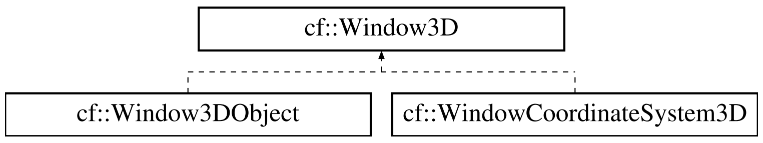
- include/window2D.h

7.23 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>
```

Inheritance diagram for cf::Window3D:



Public Types

- enum MouseButton {
  MouseButton::LEFT, MouseButton::CENTER, MouseButton::RIGHT, MouseButton::WHEEL_UP,
  MouseButton::WHEEL_DOWN }

  Friendly mousebutton mnmes.
- enum MouseButtonEvent { MouseButtonEvent::RELEASED, MouseButtonEvent::PRESSED }

  Friendly mousebutton events.
- 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 ()
- virtual bool handleMousePressedMovement (MouseButton button, int x, int y)

    handleMousePressedMovement Access mouse movement position while one mousebutton is pressed. Should return true if the default behavior (rotate Camera around object) should be dismissed.
- virtual void handleMousePressEvent (MouseButton button, MouseButtonEvent event, int x, int y)

    handleMousePress Access mousebutton presses or releases on position
- 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, glm::vec3 positionCorrection=glm::vec3(0, 0, 0))

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

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

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

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

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

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

    Type adjusted version of Window3D::drawCylinder.
- void 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 ()
- void _ZoomCamera (bool positveZoom)

Static Protected Member Functions

- static cf::Color _AdjustColorOpenGL (const cf::Color &color)

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
- glm::vec3 m_CameraPositionCorrection = glm::vec3(0.f, 0.f, 0.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 ()
- void _MouseCtlClickCallbackFunction (int button, int press, int y, int x)
- void _MouseCtlMotionCallbackFunction (int y, int x)
- void _WindowResizeEvent (int w, int h)

7.23.1 Detailed Description

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

7.23.2 Member Enumeration Documentation

7.23.2.1 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	

7.23.2.2 MouseButton

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

Friendly mousebutton mnmes.

Enumerator

LEFT	
CENTER	
RIGHT	
WHEEL_UP	
WHEEL_DOWN	

7.23.2.3 MouseButtonEvent

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

Friendly mousebutton events.

Enumerator

RELEASED	
PRESSED	

7.23.3 Constructor & Destructor Documentation

7.23.3.1 Window3D()

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

7.23.3.2 ∼Window3D()

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

7.23.4 Member Function Documentation

7.23.4.1 _AdjustCamera()

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

7.23.4.2 _AdjustColorOpenGL()

```
static cf::Color cf::Window3D::_AdjustColorOpenGL (
             const cf::Color & color )  [static], [protected]
```

7.23.4.3 _ZoomCamera()

```
void cf::Window3D::_ZoomCamera (
             bool positveZoom )  [protected]
```

7.23.4.4 clear()

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

7.23.4.5 disableLighting()

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

disableLighting Disable lightning (Default: lightning is enabled)

7.23.4.6 draw()

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

draw Draw function, this has to be implemented

7.23.4.7 drawAxis()

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

drawAxis Draw x-,y- and z-axis

Parameters

length	Axis length

7.23.4.8 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

position	Midpoint position
size	Cube size
color	Cube color

7.23.4.9 drawCylinder() [1/4]

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

drawCylinder Draws a solid clynder

Parameters

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

7.23.4.10 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.

7.23.4.11 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.

7.23.4.12 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.

7.23.4.13 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

position	Midpoint position
diameter	Sphere diamenter
color	Sphere color

7.23.4.14 enableLighting()

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

enableLighting Enable lightning (Default: lightning is enabled)

7.23.4.15 forceDisplay()

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

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

7.23.4.16 getWindowHeight()

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

7.23.4.17 getWindowWidth()

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

7.23.4.18 handleKeyboardInput()

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

handleKeyboardInput Access key input by simple override this function

Parameters

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

7.23.4.19 handleMousePressedMovement()

```
virtual bool cf::Window3D::handleMousePressedMovement (
            MouseButton button,
            int x,
            int y )  [virtual]
```

handleMousePressedMovement Access mouse movement position while one mousebutton is pressed. Should return true if the default behavior (rotate Camera around object) should be dismissed.

Parameters

button	The pressed Mousebutton
x	Mouse-x-position
y	Mouse-y-position

Returns

Should return true if the default behavior (rotate Camera around object) should be dismissed.

7.23.4.20 handleMousePressEvent()

```
virtual void cf::Window3D::handleMousePressEvent (
            MouseButton button,
            MouseButtonEvent event,
            int x,
            int y )  [virtual]
```

handleMousePress Access mousebutton presses or releases on position

Parameters

button	The pressed Mousebutton
event	The button event
x	Mouse-x-position of the event
y	Mouse-y-position of the event

7.23.4.21 printWindowUsage()

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

printWindowUsage Print camera usage to console

7.23.4.22 setCamera()

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

setCamera Set or change current camera type

Parameters

type	Camera type
lookAt	
distance	

7.23.4.23 setMaxFPS()

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

setMaxFPS Set maximum frames per second

Parameters

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

7.23.4.24 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

7.23.5 Friends And Related Function Documentation

7.23.5.1 _DrawingFunction

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

7.23.5.2 _KeyboardCallbackFunction

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

7.23.5.3 _MouseCtlClickCallbackFunction

```
void _MouseCtlClickCallbackFunction (
             int button,
             int press,
             int y,
             int x )  [friend]
```

7.23.5.4 _MouseCtlMotionCallbackFunction

```
void _MouseCtlMotionCallbackFunction (
             int y,
             int x )  [friend]
```

7.23.5.5 _WindowResizeEvent

```
void _WindowResizeEvent (
             int w,
             int h )  [friend]
```

7.23.6 Member Data Documentation

7.23.6.1 m_AngleAdjustment

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

7.23.6.2 m_CameraAdjustment

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

7.23.6.3 m_CameraPositionCorrection

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

7.23.6.4 m_CameraType

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

7.23.6.5 m_DistAdjustment

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

7.23.6.6 m_FreeCamera_LookDirection

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

7.23.6.7 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.

7.23.6.8 m_FreeCamera_UpVector

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

7.23.6.9 m_LookAt

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

7.23.6.10 m_LookAtDistance

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

7.23.6.11 m_RotationAngle_X

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

7.23.6.12 m_RotationAngle_Y

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

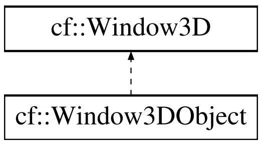
The documentation for this struct was generated from the following file:

- include/window3D.h

7.24 cf::Window3DObject Struct Reference

```
#include <window3DObjectbased.h>
```

Inheritance diagram for cf::Window3DObject:



Public Types

- enum MouseButtonEvent

    Friendly mousebutton events.
- enum MouseButton

    Friendly mousebutton mnmes.
- enum CameraType

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

Public Member Functions

- ∼Window3DObject () override=default
- void setDrawingFunction (std::function< void(Window3DObject &)> function)
- void setKeyboardInputFunction (std::function< void(unsigned char, int, int)> function)
- void setMousePressedMovementFunction (std::function< bool(MouseButton, int, int)> function)
- void setMousePressEvent (std::function< void(MouseButton, MouseButtonEvent, int, int)> function)
- void waitKeyPressed (size_t delay=0)
- void exit ()
- void disableLighting ()

    disableLighting Disable lightning (Default: lightning is enabled)
- int getWindowHeight () const
- int getWindowWidth () const
- void enableLighting ()

    enableLighting Enable lightning (Default: lightning is enabled)
- void drawCylinder (const glm::vec3 &drawingDirection, const glm::vec3 &position, float diameter=1.f, const Color &color=Color::WHITE) const

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

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

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

    Type adjusted version of Window3D::drawCylinder.
- void forceDisplay () const

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

    drawSphere Draws a solid Sphere
- void setMaxFPS (float maxFPS=0.f)

    setMaxFPS Set maximum frames per second
- void setCamera (CameraType type, glm::vec3 lookAt=glm::vec3(0, 0, 0), float distance=10.f, glm::vec3 positionCorrection=glm::vec3(0, 0, 0))

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

    drawAxis Draw x-,y- and z-axis
- void drawCube (const glm::vec3 &position, float size=1.f, const Color &color=Color::WHITE) const

    drawCube Draws a solid Cube
- void clear (const Color &color=Color::WHITE)

Static Public Member Functions

- static Window3DObject & createWindow3DObject (int ∗argc, char ∗∗argv, int width=800, int height=600, const char ∗title="chaos and fractals")
- static void printWindowUsage ()

  printWindowUsage Print camera usage to console

Additional Inherited Members

7.24.1 Member Enumeration Documentation

7.24.1.1 CameraType

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

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

7.24.1.2 MouseButton

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

Friendly mousebutton mnmes.

7.24.1.3 MouseButtonEvent

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

Friendly mousebutton events.

7.24.2 Constructor & Destructor Documentation

7.24.2.1 ∼Window3DObject()

```
cf::Window3DObject::∼Window3DObject ( ) [override], [default]
```

7.24.3 Member Function Documentation

7.24.3.1 clear()

```
void cf::Window3D::clear
```

7.24.3.2 createWindow3DObject()

```
static Window3DObject& cf::Window3DObject::createWindow3DObject (
             int * argc,
             char ** argv,
             int width = 800,
             int height = 600,
             const char * title = "chaos and fractals" )  [static]
```

7.24.3.3 disableLighting()

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

disableLighting Disable lightning (Default: lightning is enabled)

7.24.3.4 drawAxis()

```
void cf::Window3D::drawAxis
```

drawAxis Draw x-,y- and z-axis

Parameters

length	Axis length

7.24.3.5 drawCube()

```
void cf::Window3D::drawCube
```

drawCube Draws a solid Cube

Parameters

position	Midpoint position
size	Cube size
color	Cube color

7.24.3.6 drawCylinder() [1/4]

```
void cf::Window3D::drawCylinder
```

Type adjusted version of Window3D::drawCylinder.

7.24.3.7 drawCylinder() [2/4]

```
void cf::Window3D::drawCylinder
```

drawCylinder Draws a solid clynder

Parameters

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

7.24.3.8 drawCylinder() [3/4]

```
void cf::Window3D::drawCylinder
```

Type adjusted version of Window3D::drawCylinder.

7.24.3.9 drawCylinder() [4/4]

```
void cf::Window3D::drawCylinder
```

Type adjusted version of Window3D::drawCylinder.

7.24.3.10 drawSphere()

```
void cf::Window3D::drawSphere
```

drawSphere Draws a solid Sphere

Parameters

position	Midpoint position
diameter	Sphere diamenter
color	Sphere color

7.24.3.11 enableLighting()

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

enableLighting Enable lightning (Default: lightning is enabled)

7.24.3.12 exit()

```
void cf::Window3DObject::exit ( )
```

7.24.3.13 forceDisplay()

```
void cf::Window3D::forceDisplay
```

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

7.24.3.14 getWindowHeight()

```
int cf::Window3D::getWindowHeight
```

7.24.3.15 getWindowWidth()

```
int cf::Window3D::getWindowWidth
```

7.24.3.16 printWindowUsage()

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

printWindowUsage Print camera usage to console

7.24.3.17 setCamera()

```
void cf::Window3D::setCamera
```

setCamera Set or change current camera type

Parameters

type	Camera type
lookAt	
distance	

7.24.3.18 setDrawingFunction()

```
void cf::Window3DObject::setDrawingFunction (
             std::function< void(Window3DObject &)> function )
```

7.24.3.19 setKeyboardInputFunction()

```
void cf::Window3DObject::setKeyboardInputFunction (
             std::function< void(unsigned char, int, int)> function )
```

7.24.3.20 setMaxFPS()

```
void cf::Window3D::setMaxFPS
```

setMaxFPS Set maximum frames per second

Parameters

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

7.24.3.21 setMousePressedMovementFunction()

```
void cf::Window3DObject::setMousePressedMovementFunction (
             std::function< bool(MouseButton, int, int)> function )
```

7.24.3.22 setMousePressEvent()

```
void cf::Window3DObject::setMousePressEvent (
             std::function< void(MouseButton, MouseButtonEvent, int, int)> function )
```

7.24.3.23 waitKeyPressed()

```
void cf::Window3DObject::waitKeyPressed (
            size_t delay = 0 )
```

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

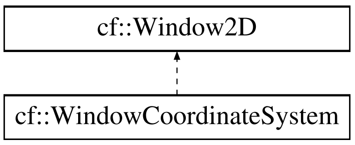
- include/window3DObjectbased.h

7.25 cf::WindowCoordinateSystem Struct Reference

The WindowCoordinateSystem struct Default class for images and raster operations.

```
#include <windowCoordinateSystem.h>
```

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 std::string &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, int lineWidth=1)

  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 line←Width=1)

  drawLinearEquation Draws a line from a point on line and direction vector
- void drawLinearEquation (float a, float b, float c, const cf::Color &color=cf::Color::BLACK, cf::Window2D::LineType type=cf::Window2D::LineType::DEFAULT, int lineWidth=1)

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

- void drawLinearEquation (const glm::vec3 &vec, const cf::Color &color=cf::Color::BLACK, cf::Window2D::LineType type=cf::Window2D::LineType::DEFAULT, int lineWidth=1)

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

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

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

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

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

    drawCirclePart Draw a partition of a circle
- void drawCirclePart (const cf::Point &center, const cf::Point &p0, const cf::Point &p1, const cf::Color &color=cf::Color::BLACK, int lineWidth=1, bool smallerAngle=true)

    drawCirclePart Draw a partition of a circle
- void clear (const cf::Color &color=cf::Color::WHITE)
- 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 floodFill (cf::Point startingPoint, const cf::Color &color)

    floodFill Fills an area
- Color getColor (float x, float y) const
- int getHeight () const

    getHeight Acess to underlying image height
- const cf::Interval & getIntervalX () const

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

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

    getWidth Acess to underlying image width
- float getWindowDisplayScale () const
- void saveImage (const char ∗filePath) const

    saveImage Saves current image to harddrive
- void setColor (float x, float y, const Color &color)
- void setWindowDisplayScale (float scale)

    setWindowDisplayScale Scales the image before displaying
- void show () const

    show Show image, on first call it may require additional time to display content correctly (in those cases use wait←Key(1000))
- 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
- cf::Point waitMouseInput ()

    waitMouseInput Blocks until mouse input has been given

Additional Inherited Members

7.25.1 Detailed Description

The WindowCoordinateSystem struct Default class for images and raster operations.

7.25.2 Member Enumeration Documentation

7.25.2.1 LineType

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

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

7.25.3 Constructor & Destructor Documentation

7.25.3.1 WindowCoordinateSystem()

```
cf::WindowCoordinateSystem::WindowCoordinateSystem (
            int width,
            const cf::Interval & range_x,
            const cf::Interval & range_y,
            const std::string & windowName = "Computer Geometry",
            const cf::Color & startColor = cf::Color::WHITE )
```

WindowCoordinateSystem Constructor.

Parameters

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

7.25.3.2 ∼WindowCoordinateSystem()

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

7.25.4 Member Function Documentation

7.25.4.1 clear()

```
void cf::Window2D::clear
```

7.25.4.2 convert_intervalLength_to_pixelLength()

```
float cf::WindowCoordinateSystem::convert_intervalLength_to_pixelLength (
             float intervalLength ) const
```

convert_intervalLength_to_pixelLength Converts length from interval to pixel

Parameters

intervalLength	

Returns

7.25.4.3 convert_pixelLength_to_intervalLength()

```
float cf::WindowCoordinateSystem::convert_pixelLength_to_intervalLength (
             float pixelLength ) const
```

convert_pixelLength_to_intervalLength Converts length from pixel to interval

Parameters

pixelLength	

Returns

7.25.4.4 drawAxis()

```
void cf::Window2D::drawAxis
```

drawAxis This function draws x and y axis based on Interval

Parameters

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

7.25.4.5 drawCircle()

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

drawCircle Draws a circle with interval radius

Parameters

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

7.25.4.6 drawCirclePart() [1/2]

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

drawCirclePart Draw a partition of a circle

Parameters

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

7.25.4.7 drawCirclePart() [2/2]

```
void cf::WindowCoordinateSystem::drawCirclePart (
             const cf::Point & center,
             const cf::Point & p0,
             const cf::Point & p1,
             const cf::Color & color = cf::Color::BLACK,
             int lineWidth = 1,
             bool smallerAngle = true )
```

drawCirclePart Draw a partition of a circle

Parameters

center	Circle center
p0	1st point on the Circle line
p1	2nd point on the Circle line
color	Circle color
lineWidth	Line width
smallerAngle	Choose wich part of the Circle should be drawn (default: smaller part of the circle)

7.25.4.8 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 )
```

drawLine Draw a simple line of width 1

Parameters

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

7.25.4.9 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 )
```

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

Parameters

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

7.25.4.10 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 )
```

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

Parameters

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

7.25.4.11 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 )
```

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

Parameters

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

7.25.4.12 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 )
```

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

Parameters

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

7.25.4.13 drawPoint()

```
void cf::WindowCoordinateSystem::drawPoint (
             const cf::Point & pos,
             const cf::Color & color = cf::Color::BLACK,
             int lineWidth = 1 )
```

drawPoint Draws a cross-shaped point

Parameters

pos	Cross position
color	Cross color

7.25.4.14 floodFill()

```
void cf::Window2D::floodFill
```

floodFill Fills an area

Parameters

startingPoint	First point to be colored
color	Fill color

7.25.4.15 getColor()

```
Color cf::Window2D::getColor
```

7.25.4.16 getHeight()

```
int cf::Window2D::getHeight
```

getHeight Acess to underlying image height

Returns

Height

7.25.4.17 getIntervalX()

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

getIntervalX Const access to interval in x direction

Returns

7.25.4.18 getIntervalY()

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

getIntervalY Const access to interval in y direction

Returns

7.25.4.19 getWidth()

```
int cf::Window2D::getWidth
```

getWidth Acess to underlying image width

Returns

Width

7.25.4.20 getWindowDisplayScale()

```
float cf::Window2D::getWindowDisplayScale
```

7.25.4.21 saveImage()

```
void cf::Window2D::saveImage
```

saveImage Saves current image to harddrive

Parameters

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

7.25.4.22 setColor()

```
void cf::Window2D::setColor
```

7.25.4.23 setInterval()

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

setInterval Set new interval

Parameters

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

7.25.4.24 setWindowDisplayScale()

```
void cf::Window2D::setWindowDisplayScale
```

setWindowDisplayScale Scales the image before displaying

Parameters

scale	Window scale size

7.25.4.25 show()

```
void cf::Window2D::show
```

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

7.25.4.26 waitKey()

```
unsigned char cf::Window2D::waitKey
```

waitKey Block access until key input on window

Parameters

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

Returns

7.25.4.27 waitMouseInput() [1/2]

```
cf::Point cf::Window2D::waitMouseInput
```

waitMouseInput Blocks until mouse input has been given

Returns

7.25.4.28 waitMouseInput() [2/2]

```
void cf::Window2D::waitMouseInput
```

waitMouseInput Blocks until mouse input has been given

Parameters

x	X-Window position
y	Y-Window position

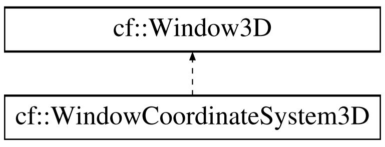
The documentation for this struct was generated from the following file:

- include/windowCoordinateSystem.h

7.26 cf::WindowCoordinateSystem3D Struct Reference

```
#include <windowCoordinateSystem3D.h>
```

Inheritance diagram for cf::WindowCoordinateSystem3D:



Public Types

- enum MULTI_VECTOR_TYPE {
  POINT, POINT_PAIR, LINE, CIRCLE,
  PLANE, SPHERE, UNKOWN }
- enum SPACE_TYPE { IPNS, OPNS }

Public Member Functions

- WindowCoordinateSystem3D (int ∗argc, char ∗∗argv, const Interval &interval={-1.5, 1.5}, int width=800, int height=600, const char ∗title="chaos and fractals")
- void drawPlane (const glm::vec4 &vec, const cf::Color &color=cf::Color::RED, uint8_t alpha=127)
- void drawPlane (const glm::vec3 &normal, const glm::vec3 &point, const cf::Color &color=cf::Color::RED, uint8_t alpha=127)
- void drawPoint (const glm::vec3 &pos, const cf::Color &color=cf::Color::BLACK, uint8_t alpha=255, float radius=0.05f)
- void drawLine (const glm::vec3 &point, const glm::vec3 &dir, const cf::Color &color=cf::Color::BLUE, float lineThickness=3.f)
- void drawSphere (const glm::vec3 &center, float radius, uint8_t alpha=255, const cf::Color &color=cf::Color::GREEN)
- void drawCircle (const glm::vec3 &center, const glm::vec3 normal, float radius, const cf::Color &color=cf::Color::GREY, float lineThickness=5.f)
- void clearWindow (const cf::Color &color=cf::Color::WHITE)
- unsigned char waitKey ()
- template<typename _Function >
  int beginDrawing (_Function &&f)
- template<typename _ValueType >
  MULTI_VECTOR_TYPE getMultiVectorType (SPACE_TYPE spaceType, const cf::MultiVector< _ValueType > &mulVec) const
- template<typename _ValueType >
  void drawMultiVector (SPACE_TYPE spaceType, const cf::MultiVector< _ValueType > &vec, const cf::Color &color, uint8_t alpha)

Additional Inherited Members

7.26.1 Member Enumeration Documentation

7.26.1.1 MULTI_VECTOR_TYPE

`enum cf::WindowCoordinateSystem3D::MULTI_VECTOR_TYPE`

Enumerator

POINT	
POINT_PAIR	
LINE	
CIRCLE	
PLANE	
SPHERE	
UNKOWN	

7.26.1.2 SPACE_TYPE

`enum cf::WindowCoordinateSystem3D::SPACE_TYPE`

Enumerator

IPNS	
OPNS	

7.26.2 Constructor & Destructor Documentation

7.26.2.1 WindowCoordinateSystem3D()

```
cf::WindowCoordinateSystem3D::WindowCoordinateSystem3D (
             int * argc,
             char ** argv,
             const Interval & interval = {-1.5, 1.5},
             int width = 800,
             int height = 600,
             const char * title = "chaos and fractals" )
```

7.26.3 Member Function Documentation

7.26.3.1 beginDrawing()

```
template<typename _Function >
int cf::WindowCoordinateSystem3D::beginDrawing (
             _Function && f )  [inline]
```

7.26.3.2 clearWindow()

```
void cf::WindowCoordinateSystem3D::clearWindow (
             const cf::Color & color = cf::Color::WHITE )
```

7.26.3.3 drawCircle()

```
void cf::WindowCoordinateSystem3D::drawCircle (
             const glm::vec3 & center,
             const glm::vec3 normal,
             float radius,
             const cf::Color & color = cf::Color::GREY,
             float lineThickness = 5.f )
```

7.26.3.4 drawLine()

```
void cf::WindowCoordinateSystem3D::drawLine (
            const glm::vec3 & point,
            const glm::vec3 & dir,
            const cf::Color & color = cf::Color::BLUE,
            float lineThickness = 3.f )
```

7.26.3.5 drawMultiVector()

```
template<typename _ValueType >
void cf::WindowCoordinateSystem3D::drawMultiVector (
            SPACE_TYPE spaceType,
            const cf::MultiVector< _ValueType > & vec,
            const cf::Color & color,
            uint8_t alpha )  [inline]
```

7.26.3.6 drawPlane() [1/2]

```
void cf::WindowCoordinateSystem3D::drawPlane (
            const glm::vec4 & vec,
            const cf::Color & color = cf::Color::RED,
            uint8_t alpha = 127 )
```

7.26.3.7 drawPlane() [2/2]

```
void cf::WindowCoordinateSystem3D::drawPlane (
            const glm::vec3 & normal,
            const glm::vec3 & point,
            const cf::Color & color = cf::Color::RED,
            uint8_t alpha = 127 )
```

7.26.3.8 drawPoint()

```
void cf::WindowCoordinateSystem3D::drawPoint (
            const glm::vec3 & pos,
            const cf::Color & color = cf::Color::BLACK,
            uint8_t alpha = 255,
            float radius = 0.05f )
```

7.26.3.9 drawSphere()

```
void cf::WindowCoordinateSystem3D::drawSphere (
            const glm::vec3 & center,
            float radius,
            uint8_t alpha = 255,
            const cf::Color & color = cf::Color::GREEN )
```

7.26.3.10 getMultiVectorType()

```
template<typename _ValueType >
MULTI_VECTOR_TYPE cf::WindowCoordinateSystem3D::getMultiVectorType (
            SPACE_TYPE spaceType,
            const cf::MultiVector< _ValueType > & mulVec ) const  [inline]
```

TODO maybe check for valid point pair

7.26.3.11 waitKey()

```
unsigned char cf::WindowCoordinateSystem3D::waitKey ( )
```

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

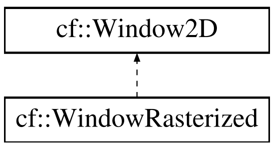
- include/windowCoordinateSystem3D.h

7.27 cf::WindowRasterized Struct Reference

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

```
#include <windowRasterized.h>
```

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 std::string &windowName="Chaos and Fractals", const cf::Color &startColor={0, 0, 0})

    WindowRasterized Constructor.
- WindowRasterized (const std::string &filePath)

    WindowRasterized Load image from file path.
- virtual ∼WindowRasterized ()=default
- void clear (const cf::Color &color=cf::Color::WHITE)
- void drawCircle (cf::Point center, int radius, int lineWidth, const cf::Color &color)

    drawCircle Draws a circle around the center
- void drawCircle (const cf::Circle &circle)

    drawCircle Draws a circle from circle class
- void drawLine (cf::Point point1, cf::Point point2, int lineWidth, const cf::Color &color)

    drawLine Draws a line from point1 to point2
- void drawLine (const cf::Line &line)

    drawLine Draws a line from line class
- void drawRectangle (cf::Point point1, cf::Point point2, int lineWidth, const cf::Color &color)

    drawRectangle Draws a rectangle from two diagonal points
- void drawRectangle (const cf::Rect &rect)

    drawRectangle Draws a rect from rect class
- 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 flippHorizontal ()

    flippHorizontal Flipp image horizontally
- void flippVertical ()

    flippHorizontal Flipp image vertically
- void floodFill (cf::Point startingPoint, const cf::Color &color)

    floodFill Fills an area
- Color getColor (float x, float y) const
- int getHeight () const

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

    getImage Direct access to the underlying image
- int getWidth () const

    getWidth Acess to underlying image width
- float getWindowDisplayScale () const
- void resize (int pixelWidth, int pixelHeight)

    resize Resize underlying image
- void saveImage (const char ∗filePath) const

    saveImage Saves current image to harddrive
- void setColor (float x, float y, const Color &color)
- void setWindowDisplayScale (float scale)

    setWindowDisplayScale Scales the image before displaying
- void show () const

    show Show image, on first call it may require additional time to display content correctly (in those cases use wait↩ Key(1000))
- 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
- cf::Point waitMouseInput ()

    waitMouseInput Blocks until mouse input has been given

Additional Inherited Members

7.27.1 Detailed Description

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

7.27.2 Member Enumeration Documentation

7.27.2.1 LineType

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

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

7.27.3 Constructor & Destructor Documentation

7.27.3.1 WindowRasterized() [1/2]

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

WindowRasterized Constructor.

Parameters

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

7.27.3.2 WindowRasterized() [2/2]

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

WindowRasterized Load image from file path.

Parameters

filePath	Path to file

7.27.3.3 ∼WindowRasterized()

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

7.27.4 Member Function Documentation

7.27.4.1 clear()

```
void cf::Window2D::clear
```

7.27.4.2 drawCircle() [1/2]

```
void cf::Window2D::drawCircle
```

drawCircle Draws a circle around the center

Parameters

point	Point within interval_x and interval_y
radius	Circle radius in pixel (not effected by intervals)
lineWidth	Pixelwidth of line (not effected by intervals), negative values fills the rectangle
color	Circle color

7.27.4.3 drawCircle() [2/2]

```
void cf::Window2D::drawCircle
```

drawCircle Draws a circle from circle class

Parameters

circle	

7.27.4.4 drawLine() [1/2]

```
void cf::Window2D::drawLine
```

drawLine Draws a line from point1 to point2

Parameters

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

7.27.4.5 drawLine() [2/2]

```
void cf::Window2D::drawLine
```

drawLine Draws a line from line class

Parameters

line	

7.27.4.6 drawRectangle() [1/2]

```
void cf::Window2D::drawRectangle
```

drawRectangle Draws a rectangle from two diagonal points

Parameters

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

7.27.4.7 drawRectangle() [2/2]

```
void cf::Window2D::drawRectangle
```

drawRectangle Draws a rect from rect class

Parameters

rect	

7.27.4.8 drawSpecializedLine()

```
void cf::Window2D::drawSpecializedLine
```

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

Parameters

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

7.27.4.9 flippHorizontal()

```
void cf::Window2D::flippHorizontal
```

flippHorizontal Flipp image horizontally

7.27.4.10 flippVertical()

```
void cf::Window2D::flippVertical
```

flippHorizontal Flipp image vertically

7.27.4.11 floodFill()

```
void cf::Window2D::floodFill
```

floodFill Fills an area

Parameters

startingPoint	First point to be colored
color	Fill color

7.27.4.12 getColor()

```
Color cf::Window2D::getColor
```

7.27.4.13 getHeight()

```
int cf::Window2D::getHeight
```

getHeight Acess to underlying image height

Returns

Height

7.27.4.14 getImage()

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

getImage Direct access to the underlying image

Returns

Image handle

7.27.4.15 getWidth()

```
int cf::Window2D::getWidth
```

getWidth Acess to underlying image width

Returns

Width

7.27.4.16 getWindowDisplayScale()

```
float cf::Window2D::getWindowDisplayScale
```

7.27.4.17 resize()

```
void cf::Window2D::resize
```

resize Resize underlying image

Parameters

pixelWidth	New width
pixelHeight	New height

7.27.4.18 saveImage()

```
void cf::Window2D::saveImage
```

saveImage Saves current image to harddrive

Parameters

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

7.27.4.19 setColor()

```
void cf::Window2D::setColor
```

7.27.4.20 setWindowDisplayScale()

```
void cf::Window2D::setWindowDisplayScale
```

setWindowDisplayScale Scales the image before displaying

Parameters

scale	Window scale size

7.27.4.21 show()

```
void cf::Window2D::show
```

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

7.27.4.22 waitKey()

```
unsigned char cf::Window2D::waitKey
```

waitKey Block access until key input on window

Parameters

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

Returns

7.27.4.23 waitMouseInput() [1/2]

```
void cf::Window2D::waitMouseInput
```

waitMouseInput Blocks until mouse input has been given

Parameters

x	X-Window position
y	Y-Window position

7.27.4.24 waitMouseInput() [2/2]

```
cf::Point cf::Window2D::waitMouseInput
```

waitMouseInput Blocks until mouse input has been given

Returns

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

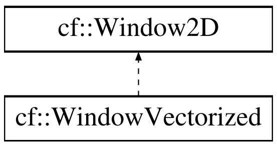
- include/windowRasterized.h

7.28 cf::WindowVectorized Struct Reference

The WindowVectorized struct Default class for images and raster operations.

```
#include <windowVectorized.h>
```

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 std::string &windowName="Chaos and Fractals", const cf::Color &startColor=cf::Color::BLACK)

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

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

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

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

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

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

  convert_intervalLength_to_pixelLength Converts length from interval to pixel
- cf::Color getColor_imageSpace (int 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
- void clear (const cf::Color &color=cf::Color::WHITE)
- 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 drawCircle (cf::Point center, int radius, int lineWidth, const cf::Color &color)

  drawCircle Draws a circle around the center
- void drawCircle (const cf::Circle &circle)

  drawCircle Draws a circle from circle class

- 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 drawCirclePart (const cf::CirclePartition &circlePartition)

  drawCirclePart Draws a circlePartition from circlePartition class
- void drawLine (cf::Point point1, cf::Point point2, int lineWidth, const cf::Color &color)

  drawLine Draws a line from point1 to point2
- void drawLine (const cf::Line &line)

  drawLine Draws a line from line class
- void drawRectangle (cf::Point point1, cf::Point point2, int lineWidth, const cf::Color &color)

  drawRectangle Draws a rectangle from two diagonal points
- void drawRectangle (const cf::Rect &rect)

  drawRectangle Draws a rect from rect class
- 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 floodFill (cf::Point startingPoint, const cf::Color &color)

  floodFill Fills an area
- Color getColor (float x, float y) const
- int getHeight () const

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

  getImage Direct access to the underlying image
- const cf::Interval & getIntervalX () const

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

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

  getWidth Acess to underlying image width
- float getWindowDisplayScale () const
- void saveImage (const char ∗filePath) const

  saveImage Saves current image to harddrive
- void setColor (float x, float y, const Color &color)
- void setWindowDisplayScale (float scale)

  setWindowDisplayScale Scales the image before displaying
- void show () const

  show Show image, on first call it may require additional time to display content correctly (in those cases use wait←Key(1000))
- 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
- cf::Point waitMouseInput ()

  waitMouseInput Blocks until mouse input has been given

Additional Inherited Members

7.28.1 Detailed Description

The WindowVectorized struct Default class for images and raster operations.

7.28.2 Member Enumeration Documentation

7.28.2.1 LineType

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

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

7.28.3 Constructor & Destructor Documentation

7.28.3.1 WindowVectorized() [1/2]

```
cf::WindowVectorized::WindowVectorized (
             int width,
             const cf::Interval & range_x,
             const cf::Interval & range_y,
             const std::string & windowName = "Chaos and Fractals",
             const cf::Color & startColor = cf::Color::BLACK )
```

WindowVectorized Constructor.

Parameters

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

7.28.3.2 WindowVectorized() [2/2]

```
cf::WindowVectorized::WindowVectorized (
             const std::string & filePath,
             int width,
             const cf::Interval & range_x,
             const cf::Interval & range_y )
```

WindowVectorized Image reading constructoor.

Parameters

filePath	Path to image file

Parameters

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

7.28.3.3 ∼WindowVectorized()

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

7.28.4 Member Function Documentation

7.28.4.1 clear()

```
void cf::Window2D::clear
```

7.28.4.2 convert_intervalLength_to_pixelLength()

```
float cf::WindowVectorized::convert_intervalLength_to_pixelLength (
             float intervalLength ) const
```

convert_intervalLength_to_pixelLength Converts length from interval to pixel

Parameters

intervalLength	Length to be converted to pixel length

Returns

7.28.4.3 convert_pixelLength_to_intervalLength()

```
float cf::WindowVectorized::convert_pixelLength_to_intervalLength (
             float pixelLength ) const
```

convert_pixelLength_to_intervalLength Converts length from pixel to interval

Parameters

pixelLength	Length to be converted to the intervall length

Returns

7.28.4.4 drawAxis()

```
void cf::Window2D::drawAxis
```

drawAxis This function draws x and y axis based on Interval

Parameters

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

7.28.4.5 drawCircle() [1/2]

```
void cf::Window2D::drawCircle
```

drawCircle Draws a circle around the center

Parameters

point	Point within interval_x and interval_y
radius	Circle radius in pixel (not effected by intervals)
lineWidth	Pixelwidth of line (not effected by intervals), negative values fills the rectangle
color	Circle color

7.28.4.6 drawCircle() [2/2]

```
void cf::Window2D::drawCircle
```

drawCircle Draws a circle from circle class

Parameters

circle	

7.28.4.7 drawCirclePart() [1/2]

```
void cf::Window2D::drawCirclePart
```

drawCirclePart Draws a part of a circle

Parameters

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

7.28.4.8 drawCirclePart() [2/2]

```
void cf::Window2D::drawCirclePart
```

drawCirclePart Draws a circlePartition from circlePartition class

Parameters

circlePartition	

7.28.4.9 drawLine() [1/2]

```
void cf::Window2D::drawLine
```

drawLine Draws a line from point1 to point2

Parameters

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

7.28.4.10 drawLine() [2/2]

```
void cf::Window2D::drawLine
```

drawLine Draws a line from line class

Parameters

line	

7.28.4.11 drawRectangle() [1/2]

```
void cf::Window2D::drawRectangle
```

drawRectangle Draws a rectangle from two diagonal points

Parameters

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

7.28.4.12 drawRectangle() [2/2]

```
void cf::Window2D::drawRectangle
```

drawRectangle Draws a rect from rect class

Parameters

rect	

7.28.4.13 drawSpecializedLine()

```
void cf::Window2D::drawSpecializedLine
```

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

Parameters

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

7.28.4.14 floodFill()

```
void cf::Window2D::floodFill
```

floodFill Fills an area

Parameters

startingPoint	First point to be colored
color	Fill color

7.28.4.15 getColor()

```
Color cf::Window2D::getColor
```

7.28.4.16 getColor_imageSpace()

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

getColor_imageSpace Get color from image x/y position

Parameters

x	X position
y	Y position

Returns

7.28.4.17 getHeight()

```
int cf::Window2D::getHeight
```

getHeight Acess to underlying image height

Returns

Height

7.28.4.18 getImage()

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

getImage Direct access to the underlying image

Returns

Image handle

7.28.4.19 getIntervalX()

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

getIntervalX Const access to interval in x direction

Returns

7.28.4.20 getIntervalY()

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

getIntervalY Const access to interval in y direction

Returns

7.28.4.21 getWidth()

```
int cf::Window2D::getWidth
```

getWidth Acess to underlying image width

Returns

Width

7.28.4.22 getWindowDisplayScale()

```
float cf::Window2D::getWindowDisplayScale
```

7.28.4.23 saveImage()

```
void cf::Window2D::saveImage
```

saveImage Saves current image to harddrive

Parameters

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

7.28.4.24 setColor()

```
void cf::Window2D::setColor
```

7.28.4.25 setColor_imageSpace()

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

setColor_imageSpace Set color from image x/y position

Parameters

x	X position
y	Y position
color	Color to be set

7.28.4.26 setInterval()

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

setInterval Set new interval

Parameters

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

7.28.4.27 setWindowDisplayScale()

```
void cf::Window2D::setWindowDisplayScale
```

setWindowDisplayScale Scales the image before displaying

Parameters

scale	Window scale size

7.28.4.28 show()

```
void cf::Window2D::show
```

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

7.28.4.29 transformPoint_fromImage_toInterval()

```
cf::Point cf::WindowVectorized::transformPoint_fromImage_toInterval (
            cf::Point point )
```

transformPoint_fromImage_toInterval Transform point from pixel position to interval position

Parameters

point	Point to be transformed

Returns

Transformed point

7.28.4.30 transformPoint_fromInterval_toImage()

```
cf::Point cf::WindowVectorized::transformPoint_fromInterval_toImage (
             cf::Point point )
```

transformPoint_fromInterval_toImage Transform point from interval position to pixel position

Parameters

point	Point to be transformed

Returns

Transformed point

7.28.4.31 waitKey()

```
unsigned char cf::Window2D::waitKey
```

waitKey Block access until key input on window

Parameters

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

Returns

7.28.4.32 waitMouseInput() [1/2]

```
void cf::Window2D::waitMouseInput
```

waitMouseInput Blocks until mouse input has been given

Parameters

x	X-Window position
y	Y-Window position

7.28.4.33 waitMouseInput() [2/2]

```
cf::Point cf::Window2D::waitMouseInput
```

waitMouseInput Blocks until mouse input has been given

Returns

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

- include/windowVectorized.h

Chapter 8

File Documentation

8.1 include/computerGeometry.hpp File Reference

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

Classes

- class cf::Vec3< IS_POINTVECTOR, _ValueType >

    The Vec3 struct General class for vector operations.
- class cf::Vec3< IS_POINTVECTOR, _ValueType >

    The Vec3 struct General class for vector operations.

Namespaces

- cf

Macros

- #define MSG "Error: Direction vector cannot be initialized from a cf::Point"
- #define MSG "Error: direction vector cannot be normalized!"
- #define MSG "Error: Write acces to direction vector's w component is not allowed"
- #define MSG "Error: Length calculation only possible for direction vectors"
- #define MSG "Error: Length calculation only possible for direction vectors"

Typedefs

- typedef Vec3< true, double > cf::PointVector_d
- typedef Vec3< false, double > cf::DirectionVector_d
- typedef Vec3< true, float > cf::PointVector_f
- typedef Vec3< false, float > cf::DirectionVector_f
- typedef Vec3< true, long double > cf::PointVector_ld
- typedef Vec3< false, long double > cf::DirectionVector_ld
- typedef PointVector_d cf::PointVector

  PointVector Specialiaztion of general Vec3.
- typedef DirectionVector_d cf::DirectionVector

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

Functions

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

  operator<< Simple shift operator for output

8.1.1 Macro Definition Documentation

8.1.1.1 MSG [1/5]

```
#define MSG "Error:  Direction vector cannot be initialized from a cf::Point"
```

8.1.1.2 MSG [2/5]

```
#define MSG "Error:  direction vector cannot be normalized!"
```

8.1.1.3 MSG [3/5]

```
#define MSG "Error:  Write acces to direction vector's w component is not allowed"
```

8.1.1.4 MSG [4/5]

```
#define MSG "Error:  Length calculation only possible for direction vectors"
```

8.1.1.5 MSG [5/5]

```
#define MSG "Error:  Length calculation only possible for direction vectors"
```

8.1.2 Function Documentation

8.1.2.1 operator<<()

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

operator<< Simple shift operator for output

Parameters

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

Returns

8.2 include/computerGeometry3D.hpp File Reference

```
#include "utils.h"
#include <algorithm>
#include <type_traits>
```

Classes

- struct cf::MultiVector< _ValueType >
- struct cf::MultiVector< _ValueType >
- struct cf::MultiVector< _ValueType >::Blade

Namespaces

- cf
- cf::literals

Typedefs

- typedef MultiVector< long double > cf::ldMultiVector
- typedef MultiVector< double > cf::dMultiVector
- typedef MultiVector< float > cf::fMultiVector
- typedef MultiVector< double > cf::Vec

Functions

- template<typename _ValueType >
  _ValueType cf::abs (const cf::MultiVector< _ValueType > &multiVector)

8.3 include/IFS.h File Reference

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

Classes

- struct cf::IteratedFunctionSystem

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

Namespaces

- cf

Typedefs

- typedef IteratedFunctionSystem cf::IFS

8.4 include/internal.hpp File Reference

```
#include <functional>
#include <mutex>
```

Classes

- struct cf::internal::_ProtectedFunction< _ReturnType, _Args >
- struct cf::internal::_ProtectedFunction< _ReturnType(_Args...)>

Namespaces

- cf
- cf::internal

8.5 include/LSystem.h File Reference

```
#include <map>
#include <memory>
#include <string>
#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

8.6 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

8.7 include/utils.h File Reference

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

Classes

- struct cf::Direction

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

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

  The Color struct offers a class for rgb access.
- struct cf::Color::SimpleEndlessIterator< _Size >
- struct cf::Console

  The Console struct offers utility functions for 'console'.
- struct cf::SimpleSignal

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
- 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
- template<typename _VectorType = glm::vec3>

  std::vector< _VectorType > cf::readDATFile (const std::string &filePath)

  readDATFile Reads a ∗.dat file

8.7.1 Function Documentation

8.7.1.1 operator<<() [1/5]

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

8.7.1.2 operator<<() [2/5]

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

8.7.1.3 operator<<() [3/5]

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

8.7.1.4 operator<<() [4/5]

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

8.7.1.5 operator<<() [5/5]

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

8.8 include/window2D.h File Reference

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

Classes

- class cf::Window2D

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

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

8.9 include/window3D.h File Reference

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

8.10 include/window3DObjectbased.h File Reference

```
#include "window3D.h"
#include "internal.hpp"
#include <thread>
```

Classes

- struct cf::Window3DObject

Namespaces

- cf

8.11 include/windowCoordinateSystem.h File Reference

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

Classes

- struct cf::WindowCoordinateSystem

  The WindowCoordinateSystem struct Default class for images and raster operations.

Namespaces

- cf

8.12 include/windowCoordinateSystem3D.h File Reference

```
#include "computerGeometry3D.hpp"
#include "window3D.h"
#include <mutex>
#include <thread>
```

Classes

- struct cf::WindowCoordinateSystem3D

Namespaces

- cf

8.13 include/windowRasterized.h File Reference

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

Classes

- struct cf::WindowRasterized

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

Namespaces

- cf

8.14 include/windowVectorized.h File Reference

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

Classes

- struct cf::WindowVectorized

  The WindowVectorized struct Default class for images and raster operations.

Namespaces

- cf

8.15 README.md File Reference

Index

_AdjustCamera
    cf::Window3D, 95
_AdjustColorOpenGL
    cf::Window3D, 95
_CreateUniqueWindowName
    cf::Window2D, 80
_DrawingFunction
    cf::Window3D, 101
_KeyboardCallbackFunction
    cf::Window3D, 101
_MouseCtlClickCallbackFunction
    cf::Window3D, 101
_MouseCtlMotionCallbackFunction
    cf::Window3D, 101
_WindowResizeEvent
    cf::Window3D, 101
_ZoomCamera
    cf::Window3D, 95
_convertFromNewInterval
    cf::Window2D, 80
_convertToNewInterval
    cf::Window2D, 80
_correctYValue
    cf::Window2D, 80
_removeWindowsSpecificCarriageReturn
    cf, 15
_window2foreground
    cf::Window2D, 80
∼Window2D
    cf::Window2D, 79
∼Window3D
    cf::Window3D, 95
∼Window3DObject
    cf::Window3DObject, 105
∼WindowCoordinateSystem
    cf::WindowCoordinateSystem, 112
∼WindowRasterized
    cf::WindowRasterized, 128
∼WindowVectorized
    cf::WindowVectorized, 137

abs
    cf, 15
AbsoluteDirection
    cf::Direction, 36

b
    cf::Color, 32
BLACK
    cf::Color, 32
BLUE
    cf::Color, 32
begin
    cf::LSystem_Controller, 48
beginDrawing
    cf::WindowCoordinateSystem3D, 123
Blade
    cf::MultiVector::Blade, 21

CYAN
    cf::Color, 32
CameraType
    cf::Window3DObject, 105
    cf::Window3D, 93
center
    cf::Circle, 24
    cf::CirclePartition, 26
cf, 11
    _removeWindowsSpecificCarriageReturn, 15
    abs, 15
    dMultiVector, 13
    degree2radian, 15
    DirectionVector, 13
    DirectionVector_d, 13
    DirectionVector_f, 13
    DirectionVector_ld, 13
    fMultiVector, 13
    IFS, 14
    LSystem, 14
    ldMultiVector, 14
    ORB, 14
    PointVector, 14
    PointVector_d, 14
    PointVector_f, 14
    PointVector_ld, 15
    radian2degree, 16
    readAntString, 16
    readDATFile, 16
    readPaletteFromFile, 17
    Vec, 15
cf::Circle, 24
    center, 24
    Circle, 24
    color, 24
    lineWidth, 25
    radius, 25
cf::CirclePartition, 25
    center, 26
    CirclePartition, 26
    color, 26

endAngle, 26
lineWidth, 26
radius, 26
startAngle, 26
cf::Color, 27
b, 32
BLACK, 32
BLUE, 32
CYAN, 32
Color, 28
CreateEndlessColorIterator, 28
g, 32
GREEN, 32
GREY, 32
invert, 29
MAGENTA, 33
ORANGE, 33
operator!=, 29
operator<, 30
operator<<, 31
operator<=, 30
operator>, 31
operator>=, 31
operator∗, 29, 31
operator∗=, 29
operator+, 29
operator+=, 29
operator-, 30
operator-=, 30
operator/, 30, 31
operator/=, 30
operator==, 30
PINK, 33
r, 33
RED, 33
RandomColor, 31
WHITE, 33
YELLOW, 33
cf::Color::SimpleEndlessIterator
operator∗, 63
operator++, 63, 64
operator->, 64
SimpleEndlessIterator, 63
cf::Color::SimpleEndlessIterator< _Size >, 63
cf::Console, 34
clearConsole, 34
printError, 34
printWarning, 35
readFloat, 35
readInt, 35
readString, 35
waitKey, 36
cf::Direction, 36
AbsoluteDirection, 36
getNextiDirection, 37
RelativeDirection, 37
toString, 37, 38
cf::Interval, 38
Interval, 38
max, 39
min, 39
operator<<, 39
translateIntervalPostion, 39
cf::IteratedFunctionSystem, 40
getAllTransformation, 40
getName, 40
getNumTransformations, 40
getRangeX, 41
getRangeY, 41
getTransformation, 41
read, 41
cf::LSystem_Controller, 47
begin, 48
end, 48
LSystem_Controller, 47
cf::LSystem_Controller::iterator, 41
iterator, 42
LSystem_Controller, 42
operator!=, 42
operator∗, 42
operator++, 42
cf::LindenmayerSystem, 43
clearWindowEachTime, 43
getAdjustmentAngle, 43
getAllProductions, 44
getAxiom, 44
getName, 44
getNumProductions, 44
getProduction, 44
getRangeX, 44
getRangeY, 44
getScale, 45
getStartAngle, 45
read, 45
cf::Line, 45
color, 46
Line, 46
lineType, 46
lineWidth, 46
point1, 46
point2, 47
cf::MultiVector
getData, 50
MultiVector, 50, 55
operator &, 51
operator &=, 51
operator _VType, 51
operator<<, 55
operator∗, 51, 52
operator∗=, 52
operator∼, 55
operator^, 55
operator^=, 55
operator+, 52, 53
operator+=, 53
operator-, 53

operator-=, 54
operator/, 54
operator/=, 54
operator=, 54
operator==, 54
operator%, 51
operator%=, 51
setData, 55
cf::MultiVector< _ValueType >, 48
cf::MultiVector< _ValueType >::Blade, 20
cf::MultiVector::Blade
Blade, 21
factor, 23
getCompleteType, 22
operator<<, 23
operator=, 22
operator==, 22
outerProduct, 23
sameType, 22
sortBladeTypes, 22
TYPE_TO_STRING, 22
TYPE, 21
type, 23
type2int, 22
cf::Orbit, 56
getAllFactors, 56
getAllStartingPoints, 56
getName, 57
getNumFactors, 57
getNumStartingPoints, 57
getRangeX, 57
getRangeY, 57
read, 57
cf::Point, 58
operator cv::Point, 59
operator!=, 59
operator∗, 59, 60
operator∗=, 59
operator+, 59
operator+=, 59
operator-, 60
operator-=, 60
operator/, 60, 61
operator/=, 60
operator==, 60
Point, 58
x, 61
y, 61
cf::Rect, 61
color, 62
lineWidth, 62
point1, 62
point2, 62
Rect, 62
cf::SimpleSignal, 64
fireSignal, 64
waitSignal, 64
cf::Vec3
getVector90Degree, 68
getW, 68
getX, 68
getY, 69
isPointVector, 69
length, 69
normalize, 69
operator cf::Point, 70
operator cf::Vec3< PV_RHS, _VType >, 70
operator const glmVec3 &, 70
operator glm::vec3, 70
operator!=, 70
operator<<), 76
operator∗, 71, 72, 75
operator∗=, 72
operator+, 72
operator+=, 72
operator-, 73
operator-=, 73
operator=, 73
operator==, 73
operator%, 71
operator%=, 71
operator[], 74
self_type, 67
setW, 74
setX, 75
setY, 75
value_type, 67
Vec3, 67, 68, 76
cf::Vec3< IS_POINTVECTOR, _ValueType >, 65
cf::Window2D, 76
_CreateUniqueWindowName, 80
_convertFromNewInterval, 80
_convertToNewInterval, 80
_correctYValue, 80
_window2foreground, 80
∼Window2D, 79
clear, 80
drawAxis, 81
drawCircle, 81
drawCirclePart, 82
drawLine, 82, 83
drawRectangle, 83
drawSpecializedLine, 84
flippHorizontal, 84
flippVertical, 84
floodFill, 84
getColor, 85
getHeight, 85
getImage, 85
getIntervalX, 85
getIntervalY, 86
getInvertYAxis, 86
getWidth, 86
getWindowDisplayScale, 86
LineType, 79
m_FristShowCall, 90

m_Image, 90
m_IntervalChanged, 90
m_IntervalX, 90
m_IntervalY, 90
m_InvertYAxis, 90
m_MouseCallBackStorage, 90
m_WindowName, 91
m_WindowScale, 91
operator=, 86
resetInterval, 87
resize, 87
saveImage, 87
setColor, 87
setInvertYAxis, 88
setNewInterval, 88
setWindowDisplayScale, 88
show, 89
waitKey, 89
waitMouseInput, 89
Window2D, 79
cf::Window3DObject, 103
∼Window3DObject, 105
CameraType, 105
clear, 106
createWindow3DObject, 106
disableLighting, 106
drawAxis, 106
drawCube, 106
drawCylinder, 107
drawSphere, 107
enableLighting, 108
exit, 108
forceDisplay, 108
getWindowHeight, 108
getWindowWidth, 108
MouseButton, 105
MouseButtonEvent, 105
printWindowUsage, 108
setCamera, 108
setDrawingFunction, 109
setKeyboardInputFunction, 109
setMaxFPS, 109
setMousePressEvent, 109
setMousePressedMovementFunction, 109
waitKeyPressed, 109
cf::Window3D, 91
_AdjustCamera, 95
_AdjustColorOpenGL, 95
_DrawingFunction, 101
_KeyboardCallbackFunction, 101
_MouseCtlClickCallbackFunction, 101
_MouseCtlMotionCallbackFunction, 101
_WindowResizeEvent, 101
_ZoomCamera, 95
∼Window3D, 95
CameraType, 93
clear, 95
disableLighting, 95
draw, 95
drawAxis, 96
drawCube, 96
drawCylinder, 96, 97
drawSphere, 97
enableLighting, 98
forceDisplay, 98
getWindowHeight, 98
getWindowWidth, 98
handleKeyboardInput, 98
handleMousePressEvent, 99
handleMousePressedMovement, 99
m_AngleAdjustment, 101
m_CameraAdjustment, 102
m_CameraPositionCorrection, 102
m_CameraType, 102
m_DistAdjustment, 102
m_FreeCamera_LookDirection, 102
m_FreeCamera_UpVector, 102
m_FreeCamera_position, 102
m_LookAt, 103
m_LookAtDistance, 103
m_RotationAngle_X, 103
m_RotationAngle_Y, 103
MouseButton, 94
MouseButtonEvent, 94
printWindowUsage, 99
setCamera, 100
setMaxFPS, 100
startDrawing, 100
Window3D, 94
cf::WindowCoordinateSystem, 110
∼WindowCoordinateSystem, 112
clear, 113
convert_intervalLength_to_pixelLength, 113
convert_pixelLength_to_intervalLength, 113
drawAxis, 113
drawCircle, 114
drawCirclePart, 114, 115
drawLine, 115
drawLinearEquation, 115–117
drawPoint, 117
floodFill, 117
getColor, 118
getHeight, 118
getIntervalX, 118
getIntervalY, 118
getWidth, 118
getWindowDisplayScale, 119
LineType, 112
saveImage, 119
setColor, 119
setInterval, 119
setWindowDisplayScale, 120
show, 120
waitKey, 120
waitMouseInput, 120, 121
WindowCoordinateSystem, 112

cf::WindowCoordinateSystem3D, 121
  beginDrawing, 123
  clearWindow, 123
  drawCircle, 123
  drawLine, 123
  drawMultiVector, 124
  drawPlane, 124
  drawPoint, 124
  drawSphere, 124
  getMultiVectorType, 125
  MULTI_VECTOR_TYPE, 122
  SPACE_TYPE, 122
  waitKey, 125
  WindowCoordinateSystem3D, 123
cf::WindowRasterized, 125
  ∼WindowRasterized, 128
  clear, 128
  drawCircle, 128
  drawLine, 129
  drawRectangle, 129
  drawSpecializedLine, 130
  flippHorizontal, 130
  flippVertical, 130
  floodFill, 130
  getColor, 131
  getHeight, 131
  getImage, 131
  getWidth, 131
  getWindowDisplayScale, 131
  LineType, 127
  resize, 131
  saveImage, 132
  setColor, 132
  setWindowDisplayScale, 132
  show, 132
  waitKey, 132
  waitMouseInput, 133
  WindowRasterized, 127
cf::WindowVectorized, 134
  ∼WindowVectorized, 137
  clear, 137
  convert_intervalLength_to_pixelLength, 137
  convert_pixelLength_to_intervalLength, 137
  drawAxis, 138
  drawCircle, 138
  drawCirclePart, 139
  drawLine, 139, 140
  drawRectangle, 140
  drawSpecializedLine, 140
  floodFill, 141
  getColor, 141
  getColor_imageSpace, 141
  getHeight, 141
  getImage, 142
  getIntervalX, 142
  getIntervalY, 142
  getWidth, 142
  getWindowDisplayScale, 143
  LineType, 136
  saveImage, 143
  setColor, 143
  setColor_imageSpace, 143
  setInterval, 144
  setWindowDisplayScale, 144
  show, 144
  transformPoint_fromImage_toInterval, 144
  transformPoint_fromInterval_toImage, 145
  waitKey, 145
  waitMouseInput, 145, 146
  WindowVectorized, 136
cf::internal, 17
cf::internal::_ProtectedFunction< _ReturnType(_←Args...)>, 19
  operator(), 19
  set, 19
cf::internal::_ProtectedFunction< _ReturnType, _Args >, 19
cf::literals, 17
Circle
  cf::Circle, 24
CirclePartition
  cf::CirclePartition, 26
clear
  cf::Window2D, 80
  cf::Window3DObject, 106
  cf::Window3D, 95
  cf::WindowCoordinateSystem, 113
  cf::WindowRasterized, 128
  cf::WindowVectorized, 137
clearConsole
  cf::Console, 34
clearWindow
  cf::WindowCoordinateSystem3D, 123
clearWindowEachTime
  cf::LindenmayerSystem, 43
Color
  cf::Color, 28
color
  cf::Circle, 24
  cf::CirclePartition, 26
  cf::Line, 46
  cf::Rect, 62
computerGeometry.hpp
  MSG, 148
  operator<<, 149
convert_intervalLength_to_pixelLength
  cf::WindowCoordinateSystem, 113
  cf::WindowVectorized, 137
convert_pixelLength_to_intervalLength
  cf::WindowCoordinateSystem, 113
  cf::WindowVectorized, 137
CreateEndlessColorIterator
  cf::Color, 28
createWindow3DObject
  cf::Window3DObject, 106

dMultiVector

cf, 13
degree2radian
cf, 15
DirectionVector
cf, 13
DirectionVector_d
cf, 13
DirectionVector_f
cf, 13
DirectionVector_ld
cf, 13
disableLighting
cf::Window3DObject, 106
cf::Window3D, 95
draw
cf::Window3D, 95
drawAxis
cf::Window2D, 81
cf::Window3DObject, 106
cf::Window3D, 96
cf::WindowCoordinateSystem, 113
cf::WindowVectorized, 138
drawCircle
cf::Window2D, 81
cf::WindowCoordinateSystem, 114
cf::WindowCoordinateSystem3D, 123
cf::WindowRasterized, 128
cf::WindowVectorized, 138
drawCirclePart
cf::Window2D, 82
cf::WindowCoordinateSystem, 114, 115
cf::WindowVectorized, 139
drawCube
cf::Window3DObject, 106
cf::Window3D, 96
drawCylinder
cf::Window3DObject, 107
cf::Window3D, 96, 97
drawLine
cf::Window2D, 82, 83
cf::WindowCoordinateSystem, 115
cf::WindowCoordinateSystem3D, 123
cf::WindowRasterized, 129
cf::WindowVectorized, 139, 140
drawLinearEquation
cf::WindowCoordinateSystem, 115–117
drawMultiVector
cf::WindowCoordinateSystem3D, 124
drawPlane
cf::WindowCoordinateSystem3D, 124
drawPoint
cf::WindowCoordinateSystem, 117
cf::WindowCoordinateSystem3D, 124
drawRectangle
cf::Window2D, 83
cf::WindowRasterized, 129
cf::WindowVectorized, 140
drawSpecializedLine
cf::Window2D, 84
cf::WindowRasterized, 130
cf::WindowVectorized, 140
drawSphere
cf::Window3DObject, 107
cf::Window3D, 97
cf::WindowCoordinateSystem3D, 124

enableLighting
cf::Window3DObject, 108
cf::Window3D, 98
end
cf::LSystem_Controller, 48
endAngle
cf::CirclePartition, 26
exit
cf::Window3DObject, 108

fMultiVector
cf, 13
factor
cf::MultiVector::Blade, 23
fireSignal
cf::SimpleSignal, 64
flippHorizontal
cf::Window2D, 84
cf::WindowRasterized, 130
flippVertical
cf::Window2D, 84
cf::WindowRasterized, 130
floodFill
cf::Window2D, 84
cf::WindowCoordinateSystem, 117
cf::WindowRasterized, 130
cf::WindowVectorized, 141
forceDisplay
cf::Window3DObject, 108
cf::Window3D, 98

g
cf::Color, 32
GREEN
cf::Color, 32
GREY
cf::Color, 32
getAdjustmentAngle
cf::LindenmayerSystem, 43
getAllFactors
cf::Orbit, 56
getAllProductions
cf::LindenmayerSystem, 44
getAllStartingPoints
cf::Orbit, 56
getAllTransformation
cf::IteratedFunctionSystem, 40
getAxiom
cf::LindenmayerSystem, 44
getColor
cf::Window2D, 85

cf::WindowCoordinateSystem, 118
cf::WindowRasterized, 131
cf::WindowVectorized, 141
getColor_imageSpace
cf::WindowVectorized, 141
getCompleteType
cf::MultiVector::Blade, 22
getData
cf::MultiVector, 50
getHeight
cf::Window2D, 85
cf::WindowCoordinateSystem, 118
cf::WindowRasterized, 131
cf::WindowVectorized, 141
getImage
cf::Window2D, 85
cf::WindowRasterized, 131
cf::WindowVectorized, 142
getIntervalX
cf::Window2D, 85
cf::WindowCoordinateSystem, 118
cf::WindowVectorized, 142
getIntervalY
cf::Window2D, 86
cf::WindowCoordinateSystem, 118
cf::WindowVectorized, 142
getInvertYAxis
cf::Window2D, 86
getMultiVectorType
cf::WindowCoordinateSystem3D, 125
getName
cf::IteratedFunctionSystem, 40
cf::LindenmayerSystem, 44
cf::Orbit, 57
getNextiDirection
cf::Direction, 37
getNumFactors
cf::Orbit, 57
getNumProductions
cf::LindenmayerSystem, 44
getNumStartingPoints
cf::Orbit, 57
getNumTransformations
cf::IteratedFunctionSystem, 40
getProduction
cf::LindenmayerSystem, 44
getRangeX
cf::IteratedFunctionSystem, 41
cf::LindenmayerSystem, 44
cf::Orbit, 57
getRangeY
cf::IteratedFunctionSystem, 41
cf::LindenmayerSystem, 44
cf::Orbit, 57
getScale
cf::LindenmayerSystem, 45
getStartAngle
cf::LindenmayerSystem, 45
getTransformation
cf::IteratedFunctionSystem, 41
getVector90Degree
cf::Vec3, 68
getWidth
cf::Window2D, 86
cf::WindowCoordinateSystem, 118
cf::WindowRasterized, 131
cf::WindowVectorized, 142
getWindowDisplayScale
cf::Window2D, 86
cf::WindowCoordinateSystem, 119
cf::WindowRasterized, 131
cf::WindowVectorized, 143
getWindowHeight
cf::Window3DObject, 108
cf::Window3D, 98
getWindowWidth
cf::Window3DObject, 108
cf::Window3D, 98
getW
cf::Vec3, 68
getX
cf::Vec3, 68
getY
cf::Vec3, 69

handleKeyboardInput
cf::Window3D, 98
handleMousePressEvent
cf::Window3D, 99
handleMousePressedMovement
cf::Window3D, 99

IFS
cf, 14
include/IFS.h, 150
include/LSystem.h, 151
include/ORB.h, 151
include/computerGeometry.hpp, 147
include/computerGeometry3D.hpp, 149
include/internal.hpp, 150
include/utils.h, 152
include/window2D.h, 153
include/window3D.h, 154
include/window3DObjectbased.h, 154
include/windowCoordinateSystem.h, 155
include/windowCoordinateSystem3D.h, 155
include/windowRasterized.h, 155
include/windowVectorized.h, 156
Interval
cf::Interval, 38
invert
cf::Color, 29
isPointVector
cf::Vec3, 69
iterator
cf::LSystem_Controller::iterator, 42

LSystem
    cf, 14
LSystem_Controller
    cf::LSystem_Controller, 47
    cf::LSystem_Controller::iterator, 42
ldMultiVector
    cf, 14
length
    cf::Vec3, 69
Line
    cf::Line, 46
LineType
    cf::Window2D, 79
    cf::WindowCoordinateSystem, 112
    cf::WindowRasterized, 127
    cf::WindowVectorized, 136
lineType
    cf::Line, 46
lineWidth
    cf::Circle, 25
    cf::CirclePartition, 26
    cf::Line, 46
    cf::Rect, 62

m_AngleAdjustment
    cf::Window3D, 101
m_CameraAdjustment
    cf::Window3D, 102
m_CameraPositionCorrection
    cf::Window3D, 102
m_CameraType
    cf::Window3D, 102
m_DistAdjustment
    cf::Window3D, 102
m_FreeCamera_LookDirection
    cf::Window3D, 102
m_FreeCamera_UpVector
    cf::Window3D, 102
m_FreeCamera_position
    cf::Window3D, 102
m_FristShowCall
    cf::Window2D, 90
m_Image
    cf::Window2D, 90
m_IntervalChanged
    cf::Window2D, 90
m_IntervalX
    cf::Window2D, 90
m_IntervalY
    cf::Window2D, 90
m_InvertYAxis
    cf::Window2D, 90
m_LookAt
    cf::Window3D, 103
m_LookAtDistance
    cf::Window3D, 103
m_MouseCallBackStorage
    cf::Window2D, 90
m_RotationAngle_X
    cf::Window3D, 103
m_RotationAngle_Y
    cf::Window3D, 103
m_WindowName
    cf::Window2D, 91
m_WindowScale
    cf::Window2D, 91
MAGENTA
    cf::Color, 33
MSG
    computerGeometry.hpp, 148
MULTI_VECTOR_TYPE
    cf::WindowCoordinateSystem3D, 122
max
    cf::Interval, 39
min
    cf::Interval, 39
MouseButton
    cf::Window3DObject, 105
    cf::Window3D, 94
MouseButtonEvent
    cf::Window3DObject, 105
    cf::Window3D, 94
MultiVector
    cf::MultiVector, 50, 55

normalize
    cf::Vec3, 69

ORANGE
    cf::Color, 33
ORB
    cf, 14
operator &
    cf::MultiVector, 51
operator &=
    cf::MultiVector, 51
operator _VType
    cf::MultiVector, 51
operator cf::Point
    cf::Vec3, 70
operator cf::Vec3< PV_RHS, _VType >
    cf::Vec3, 70
operator const glmVec3 &
    cf::Vec3, 70
operator cv::Point
    cf::Point, 59
operator glm::vec3
    cf::Vec3, 70
operator!=
    cf::Color, 29
    cf::LSystem_Controller::iterator, 42
    cf::Point, 59
    cf::Vec3, 70
operator<
    cf::Color, 30
operator<<
    cf::Color, 31
    cf::Interval, 39

cf::MultiVector, 55
cf::MultiVector::Blade, 23
computerGeometry.hpp, 149
utils.h, 153
operator<<)
cf::Vec3, 76
operator<=
cf::Color, 30
operator>
cf::Color, 31
operator>=
cf::Color, 31
operator*
cf::Color, 29, 31
cf::Color::SimpleEndlessIterator, 63
cf::LSystem_Controller::iterator, 42
cf::MultiVector, 51, 52
cf::Point, 59, 60
cf::Vec3, 71, 72, 75
operator*=
cf::Color, 29
cf::MultiVector, 52
cf::Point, 59
cf::Vec3, 72
operator∼
cf::MultiVector, 55
operator^
cf::MultiVector, 55
operator^=
cf::MultiVector, 55
operator()
cf::internal::_ProtectedFunction< _ReturnType(_↩ Args...)>, 19
operator+
cf::Color, 29
cf::MultiVector, 52, 53
cf::Point, 59
cf::Vec3, 72
operator++
cf::Color::SimpleEndlessIterator, 63, 64
cf::LSystem_Controller::iterator, 42
operator+=
cf::Color, 29
cf::MultiVector, 53
cf::Point, 59
cf::Vec3, 72
operator-
cf::Color, 30
cf::MultiVector, 53
cf::Point, 60
cf::Vec3, 73
operator->
cf::Color::SimpleEndlessIterator, 64
operator-=
cf::Color, 30
cf::MultiVector, 54
cf::Point, 60
cf::Vec3, 73
operator/
cf::Color, 30, 31
cf::MultiVector, 54
cf::Point, 60, 61
operator/=
cf::Color, 30
cf::MultiVector, 54
cf::Point, 60
operator=
cf::MultiVector, 54
cf::MultiVector::Blade, 22
cf::Vec3, 73
cf::Window2D, 86
operator==
cf::Color, 30
cf::MultiVector, 54
cf::MultiVector::Blade, 22
cf::Point, 60
cf::Vec3, 73
operator%
cf::MultiVector, 51
cf::Vec3, 71
operator%=
cf::MultiVector, 51
cf::Vec3, 71
operator[]
cf::Vec3, 74
outerProduct
cf::MultiVector::Blade, 23

PINK
cf::Color, 33
Point
cf::Point, 58
point1
cf::Line, 46
cf::Rect, 62
point2
cf::Line, 47
cf::Rect, 62
PointVector
cf, 14
PointVector_d
cf, 14
PointVector_f
cf, 14
PointVector_ld
cf, 15
printError
cf::Console, 34
printWarning
cf::Console, 35
printWindowUsage
cf::Window3DObject, 108
cf::Window3D, 99

r
cf::Color, 33
README.md, 156

RED
cf::Color, 33
radian2degree
cf, 16
radius
cf::Circle, 25
cf::CirclePartition, 26
RandomColor
cf::Color, 31
read
cf::IteratedFunctionSystem, 41
cf::LindenmayerSystem, 45
cf::Orbit, 57
readAntString
cf, 16
readDATFile
cf, 16
readFloat
cf::Console, 35
readInt
cf::Console, 35
readPaletteFromFile
cf, 17
readString
cf::Console, 35
Rect
cf::Rect, 62
RelativeDirection
cf::Direction, 37
resetInterval
cf::Window2D, 87
resize
cf::Window2D, 87
cf::WindowRasterized, 131

SPACE_TYPE
cf::WindowCoordinateSystem3D, 122
sameType
cf::MultiVector::Blade, 22
saveImage
cf::Window2D, 87
cf::WindowCoordinateSystem, 119
cf::WindowRasterized, 132
cf::WindowVectorized, 143
self_type
cf::Vec3, 67
set
cf::internal::_ProtectedFunction< _ReturnType(_←
Args...)>, 19
setCamera
cf::Window3DObject, 108
cf::Window3D, 100
setColor
cf::Window2D, 87
cf::WindowCoordinateSystem, 119
cf::WindowRasterized, 132
cf::WindowVectorized, 143
setColor_imageSpace
cf::WindowVectorized, 143
setData
cf::MultiVector, 55
setDrawingFunction
cf::Window3DObject, 109
setInterval
cf::WindowCoordinateSystem, 119
cf::WindowVectorized, 144
setInvertYAxis
cf::Window2D, 88
setKeyboardInputFunction
cf::Window3DObject, 109
setMaxFPS
cf::Window3DObject, 109
cf::Window3D, 100
setMousePressEvent
cf::Window3DObject, 109
setMousePressedMovementFunction
cf::Window3DObject, 109
setNewInterval
cf::Window2D, 88
setWindowDisplayScale
cf::Window2D, 88
cf::WindowCoordinateSystem, 120
cf::WindowRasterized, 132
cf::WindowVectorized, 144
setW
cf::Vec3, 74
setX
cf::Vec3, 75
setY
cf::Vec3, 75
show
cf::Window2D, 89
cf::WindowCoordinateSystem, 120
cf::WindowRasterized, 132
cf::WindowVectorized, 144
SimpleEndlessIterator
cf::Color::SimpleEndlessIterator, 63
sortBladeTypes
cf::MultiVector::Blade, 22
startAngle
cf::CirclePartition, 26
startDrawing
cf::Window3D, 100

TYPE_TO_STRING
cf::MultiVector::Blade, 22
TYPE
cf::MultiVector::Blade, 21
toString
cf::Direction, 37, 38
transformPoint_fromImage_toInterval
cf::WindowVectorized, 144
transformPoint_fromInterval_toImage
cf::WindowVectorized, 145
translateIntervalPostion
cf::Interval, 39
type
cf::MultiVector::Blade, 23

type2int
    cf::MultiVector::Blade, 22

utils.h
    operator<<, 153

value_type
    cf::Vec3, 67
Vec
    cf, 15
Vec3
    cf::Vec3, 67, 68, 76

WHITE
    cf::Color, 33
waitKey
    cf::Console, 36
    cf::Window2D, 89
    cf::WindowCoordinateSystem, 120
    cf::WindowCoordinateSystem3D, 125
    cf::WindowRasterized, 132
    cf::WindowVectorized, 145
waitKeyPressed
    cf::Window3DObject, 109
waitMouseInput
    cf::Window2D, 89
    cf::WindowCoordinateSystem, 120, 121
    cf::WindowRasterized, 133
    cf::WindowVectorized, 145, 146
waitSignal
    cf::SimpleSignal, 64
Window2D
    cf::Window2D, 79
Window3D
    cf::Window3D, 94
WindowCoordinateSystem
    cf::WindowCoordinateSystem, 112
WindowCoordinateSystem3D
    cf::WindowCoordinateSystem3D, 123
WindowRasterized
    cf::WindowRasterized, 127
WindowVectorized
    cf::WindowVectorized, 136

x
    cf::Point, 61

y
    cf::Point, 61
YELLOW
    cf::Color, 33