## wPlot2D

version: 0.1.0

Generated by Doxygen 1.14.0

1 wPlot2D - ECS-Based 2D Plotting Engine	1
1.1 wPlot2D	
1.1.1 Introduction	. 1
1.1.2 Features	. 1
1.1.3 Links	. 1
1.1.4 Author	. 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 wEngine Namespace Reference	. 11
6.1.1 Function Documentation	. 12
6.1.1.1 getNextComponentTypeID()	. 12
6.1.1.2 getComponentTypeID()	. 12
6.2 wPlot2D Namespace Reference	. 12
6.2.1 Enumeration Type Documentation	. 13
6.2.1.1 AxisType	. 13
6.2.1.2 NotchPosition	. 13
6.2.1.3 TitleAlignment	. 13
7 Class Documentation	15
7.1 wEngine::AssetManager Class Reference	. 15
7.1.1 Detailed Description	. 15
7.1.2 Constructor & Destructor Documentation	. 16
7.1.2.1 AssetManager() [1/2]	. 16
7.1.2.2 AssetManager() [2/2]	. 16
7.1.2.3 ~AssetManager()	. 16
7.1.3 Member Function Documentation	. 16
7.1.3.1 operator=()	. 16
7.1.3.2 LoadFont()	. 16
7.1.3.3 getFont()	. 17
7.1.3.4 RemoveFont()	. 17
7.1.3.5 debugPrintFonts()	. 18
7.2 wPlot2D::AxisEntity Class Reference	. 18

7.2.1 D	Detailed Description	21
7.2.2 C	Constructor & Destructor Documentation	22
	7.2.2.1 AxisEntity()	22
	7.2.2.2 ~AxisEntity()	22
7.2.3 N	Member Function Documentation	22
	7.2.3.1 setColor()	22
	7.2.3.2 setThickness()	22
	7.2.3.3 setArrowSize()	23
	7.2.3.4 addTitle() [1/2]	23
	7.2.3.5 addTitle() [2/2]	23
	7.2.3.6 setTitleFont()	23
	7.2.3.7 setTitleCharacterSize()	24
	7.2.3.8 setTitleColor()	24
	7.2.3.9 setTitleOffset()	24
	7.2.3.10 getTitleOffset()	24
	7.2.3.11 addNotches()	24
	7.2.3.12 setNotchesColor()	25
	7.2.3.13 setNotchesThickness()	25
	7.2.3.14 setNotchesLength()	25
	7.2.3.15 setLabelsFont()	25
	7.2.3.16 setLabelsColor()	26
	7.2.3.17 getLabelsOffset()	26
	7.2.3.18 setLabelsOffset()	26
	7.2.3.19 addLabelsOffset()	26
	7.2.3.20 setLabelsCharacterSize()	26
	7.2.3.21 setLabelsDecimalPlaces()	27
	7.2.3.22 setCustomLabels()	27
	7.2.3.23 render()	27
7.3 wEngine	::ColorComponent Class Reference	28
7.3.1 D	Detailed Description	29
7.3.2 C	Constructor & Destructor Documentation	29
	7.3.2.1 ColorComponent()	29
	7.3.2.2 ~ColorComponent()	30
7.3.3 N	Member Function Documentation	30
	7.3.3.1 getColor()	30
	7.3.3.2 setColor()	30
	7.3.3.3 debugPrint()	30
7.4 wEngine	::Component Class Reference	31
7.4.1 D	Detailed Description	31
7.4.2 C	Constructor & Destructor Documentation	31
	7.4.2.1 ~Component()	31
	7.4.2.2 Component()	32

7.4.3 Member Function Documentation	32
7.4.3.1 enable()	32
7.4.3.2 disable()	32
7.4.3.3 isEnabled()	32
7.4.3.4 setParent()	32
7.4.3.5 getParent()	32
7.5 wPlot2D::DataPlotEntity Class Reference	33
7.5.1 Detailed Description	35
7.5.2 Constructor & Destructor Documentation	35
7.5.2.1 DataPlotEntity()	35
7.5.2.2 ~DataPlotEntity()	36
7.5.3 Member Function Documentation	36
7.5.3.1 getColor()	36
7.5.3.2 getThickness()	36
7.5.3.3 getLineStyle()	36
7.5.3.4 getDashLength()	36
7.5.3.5 getGapLength()	37
7.5.3.6 setColor()	37
7.5.3.7 setThickness()	37
7.5.3.8 setLineStyle()	37
7.5.3.9 setDashLength()	37
7.5.3.10 setGapLength()	38
7.5.3.11 drawDataPlot()	38
7.6 wEngine::DiscontinuityComponent Class Reference	38
7.6.1 Detailed Description	40
7.6.2 Constructor & Destructor Documentation	40
7.6.2.1 DiscontinuityComponent()	40
7.6.2.2 ∼DiscontinuityComponent()	40
7.6.3 Member Function Documentation	41
7.6.3.1 getExcludedIntervals()	41
7.6.3.2 addExcludedInterval()	41
7.6.3.3 clearExcludedIntervals()	41
7.6.3.4 isInExcludedInterval()	41
7.6.3.5 debugPrint()	42
7.7 wEngine::Entity Class Reference	42
7.7.1 Detailed Description	43
7.7.2 Constructor & Destructor Documentation	43
7.7.2.1 Entity()	43
7.7.2.2 ~Entity()	43
7.7.3 Member Function Documentation	43
7.7.3.1 getEntityID()	43
7.7.3.2 clearComponents()	44

7.7.3.3 resetEntityIDCounter()	4	44
7.7.3.4 addComponent()	4	44
7.7.3.5 removeComponent()	4	44
7.7.3.6 hasComponent()	4	45
7.7.3.7 getComponent()	4	45
7.7.3.8 requireComponent()	4	45
7.7.3.9 getInterfaceComponent()	4	46
7.8 wEngine::FontComponent Class Reference	4	47
7.8.1 Detailed Description	4	48
7.8.2 Constructor & Destructor Documentation	4	48
7.8.2.1 FontComponent()	4	48
7.8.2.2 ~FontComponent()	4	49
7.8.3 Member Function Documentation	4	49
7.8.3.1 getFont()	4	49
7.8.3.2 setFont()	4	49
7.8.3.3 debugPrint()	4	49
7.9 wPlot2D::FrameEntity Class Reference		50
7.9.1 Detailed Description	5	52
7.9.2 Constructor & Destructor Documentation		52
7.9.2.1 FrameEntity()		52
7.9.2.2 ~FrameEntity()	5	53
7.9.3 Member Function Documentation	5	53
7.9.3.1 setEnabled()	5	53
7.9.3.2 isEnabled()	5	53
7.9.3.3 getFillColor()		53
7.9.3.4 getOutlineColor()	5	54
7.9.3.5 getThickness()		54
7.9.3.6 getPadding()		54
7.9.3.7 setFillColor()		54
7.9.3.8 setOutlineColor()		54
7.9.3.9 setThickness()		55
7.9.3.10 setPadding()	5	55
7.9.3.11 update()		55
7.9.3.12 render()	5	55
7.10 wEngine::FunctionComponent Class Reference		56
7.10.1 Detailed Description		57
7.10.2 Constructor & Destructor Documentation	5	57
7.10.2.1 FunctionComponent()		57
7.10.2.2 ~FunctionComponent()	5	58
7.10.3 Member Function Documentation	5	58
7.10.3.1 calculate()	5	58
7.10.3.2 debugPrint()		58

7.11 wPlot2D::FunctionEntity Class Reference	59
7.11.1 Detailed Description	61
7.11.2 Constructor & Destructor Documentation	62
7.11.2.1 FunctionEntity()	62
7.11.2.2 ~FunctionEntity()	62
7.11.3 Member Function Documentation	62
7.11.3.1 getPosition()	62
7.11.3.2 getColor()	62
7.11.3.3 getThickness()	63
7.11.3.4 getLineStyle()	63
7.11.3.5 getDashLength()	63
7.11.3.6 getGapLength()	63
7.11.3.7 getOffset()	64
7.11.3.8 getRotation()	64
7.11.3.9 setPosition()	64
7.11.3.10 setColor()	64
7.11.3.11 setThickness()	64
7.11.3.12 setLineStyle()	65
7.11.3.13 setDashLength()	65
7.11.3.14 setGapLength()	65
7.11.3.15 setOffset()	66
7.11.3.16 setRotation()	66
7.11.3.17 setScale()	66
7.11.3.18 addExcludedInterval()	67
7.11.3.19 clearExcludedIntervals()	67
7.11.3.20 alignToYAxis()	67
7.11.3.21 drawFunction()	67
7.12 wPlot2D::GraphicsEntity Class Reference	68
7.12.1 Detailed Description	71
7.12.2 Constructor & Destructor Documentation	72
7.12.2.1 GraphicsEntity()	72
7.12.2.2 ~ GraphicsEntity()	72
7.12.3 Member Function Documentation	72
7.12.3.1 getWindow()	72
7.12.3.2 getWindowSize()	73
7.12.3.3 setWindowSize()	73
7.12.3.4 setWindowTitle()	73
7.12.3.5 setBackgroundColor()	73
7.12.3.6 addFont()	73
7.12.3.7 getFont()	74
7.12.3.8 getOrigin()	74
7.12.3.9 setOrigin()	74

7.12.3.10 getScale()	7	'5
7.12.3.11 setScale()	7	'5
7.12.3.12 getOffset()	7	'5
7.12.3.13 setOffset()	7	'6
7.12.3.14 addAxis()	7	'6
7.12.3.15 addTitle() [1/2]	7	'6
7.12.3.16 addTitle() [2/2]	7	'6
7.12.3.17 addFunction()	7	7
7.12.3.18 addDataPlot()	7	7
7.12.3.19 addLegend()	7	7
7.12.3.20 addText() [1/2]	7	'8
7.12.3.21 addText() [2/2]	7	'8
7.12.3.22 addLine()	7	'8
7.12.3.23 saveToFile()	7	'9
7.13 wPlot2D::LabelEntity Class Reference	7	'9
7.13.1 Detailed Description	8	32
7.13.2 Constructor & Destructor Documentation	8	32
7.13.2.1 LabelEntity()	8	32
7.13.2.2 ~LabelEntity()	8	3
7.13.3 Member Function Documentation	8	3
7.13.3.1 getValue()	8	3
7.13.3.2 getCharacterSize()	8	3
7.13.3.3 getDecimalPlaces()	8	3
7.13.3.4 setFont()	8	3
7.13.3.5 setLabelText()	8	}4
7.13.3.6 setCharacterSize()	8	}4
7.13.3.7 setDecimalPlaces()	8	34
7.13.3.8 setCustomLabels()	8	}4
7.13.3.9 usesCustomLabels()	8	35
7.13.3.10 formatLabel()	8	35
7.13.3.11 render()	8	35
7.14 wPlot2D::LegendEntity Class Reference	8	36
7.14.1 Detailed Description	8	38
7.14.2 Constructor & Destructor Documentation	8	}9
7.14.2.1 LegendEntity()	8	39
7.14.2.2 ~LegendEntity()	8	}9
7.14.3 Member Function Documentation	8	}9
7.14.3.1 addltem() [1/4]	8	39
7.14.3.2 addltem() [2/4]	8	39
7.14.3.3 addltem() [3/4]	9	0
7.14.3.4 addltem() [4/4]	9	0
7.14.3.5 setFrameEnabled()	9	90

7.14.3.6 setFrameFillColor()	90
7.14.3.7 setFrameOutlineColor()	91
7.14.3.8 setFrameThickness()	91
7.14.3.9 setPadding()	91
7.14.3.10 setFont()	91
7.14.3.11 setCharacterSize()	92
7.14.3.12 setTextColor()	92
7.14.3.13 render()	92
7.15 wEngine::LengthComponent Class Reference	93
7.15.1 Detailed Description	94
7.15.2 Constructor & Destructor Documentation	94
7.15.2.1 LengthComponent()	94
$7.15.2.2 \sim \text{LengthComponent()} \ \dots $	95
7.15.3 Member Function Documentation	95
7.15.3.1 getLength()	95
7.15.3.2 setLength()	95
7.15.3.3 debugPrint()	95
7.16 wEngine::LineDrawer Class Reference	96
7.16.1 Detailed Description	96
7.16.2 Member Function Documentation	97
7.16.2.1 drawLine()	97
7.16.2.2 drawPolylineRound()	98
7.17 wPlot2D::LineEntity Class Reference	98
7.17.1 Detailed Description	101
7.17.2 Constructor & Destructor Documentation	101
7.17.2.1 LineEntity()	101
7.17.2.2 ~LineEntity()	102
7.17.3 Member Function Documentation	102
7.17.3.1 setColor()	102
7.17.3.2 setThickness()	102
7.17.3.3 getThickness()	102
7.17.3.4 setLineStyle()	102
7.17.3.5 setDashLength()	103
7.17.3.6 setGapLength()	103
7.17.3.7 getStartPoint()	103
7.17.3.8 getEndPoint()	103
7.17.3.9 hasArrow()	104
7.17.3.10 getArrowSize()	104
7.17.3.11 setArrowSize()	104
7.17.3.12 render()	104
7.18 wEngine::LineStyleComponent Class Reference	105
7 18 1 Detailed Description	106

7.18.2 Member Enumeration Documentation	06
7.18.2.1 LineStyle	06
7.18.3 Constructor & Destructor Documentation	07
7.18.3.1 LineStyleComponent()	07
$7.18.3.2 \sim \text{LineStyleComponent()} \; . \; . \; . \; . \; . \; . \; . \; . \; . \; $	07
7.18.4 Member Function Documentation	07
7.18.4.1 getStyle()	07
7.18.4.2 setStyle()	07
7.18.4.3 getDashLength()	08
7.18.4.4 setDashLength()	08
7.18.4.5 getGapLength()	08
7.18.4.6 setGapLength()	08
7.18.4.7 debugPrint()	09
7.19 wEngine::MathUtils Class Reference	09
7.19.1 Detailed Description	09
7.19.2 Member Function Documentation	09
7.19.2.1 linspace()	09
7.20 wPlot2D::NotchEntity Class Reference	10
7.20.1 Detailed Description	12
7.20.2 Constructor & Destructor Documentation	13
7.20.2.1 NotchEntity()	13
$7.20.2.2 \sim$ NotchEntity()	13
7.20.3 Member Function Documentation	13
7.20.3.1 render()	13
7.21 wEngine::NotchIntervalComponent Class Reference	14
7.21.1 Detailed Description	15
7.21.2 Constructor & Destructor Documentation	15
7.21.2.1 NotchIntervalComponent()	15
$7.21.2.2 \sim NotchIntervalComponent() \qquad . \qquad 1$	16
7.21.3 Member Function Documentation	16
7.21.3.1 getInterval()	16
7.21.3.2 setInterval()	16
7.21.3.3 debugPrint()	16
7.22 wEngine::OffsetComponent Class Reference	17
7.22.1 Detailed Description	18
7.22.2 Constructor & Destructor Documentation	18
7.22.2.1 OffsetComponent()	18
$7.22.2.2 \sim OffsetComponent()$	19
7.22.3 Member Function Documentation	19
7.22.3.1 getOffset()	19
7.22.3.2 setOffset()	19
7.22.3.3 addOffset()	19

7.22.3.4 debugPrint()	120
7.23 wEngine::PaddingComponent Class Reference	120
7.23.1 Detailed Description	121
7.23.2 Constructor & Destructor Documentation	122
7.23.2.1 PaddingComponent()	122
$7.23.2.2 \sim PaddingComponent()  .  .  .  .  .  .  .  .  .  $	122
7.23.3 Member Function Documentation	122
7.23.3.1 setPadding()	122
7.23.3.2 getPadding()	122
7.23.3.3 debugPrint()	122
7.24 wEngine::PathUtils Class Reference	123
7.24.1 Detailed Description	123
7.24.2 Member Function Documentation	124
7.24.2.1 getExecutablePath()	124
7.24.2.2 getExecutableDir()	124
7.25 wEngine::PositionComponent Class Reference	125
7.25.1 Detailed Description	126
7.25.2 Constructor & Destructor Documentation	126
7.25.2.1 PositionComponent()	126
$7.25.2.2 \sim$ PositionComponent()	127
7.25.3 Member Function Documentation	127
7.25.3.1 getPosition()	127
7.25.3.2 getLastPosition()	127
7.25.3.3 setPosition()	127
7.25.3.4 move()	127
7.25.3.5 debugPrint()	128
7.26 wEngine::RotationComponent Class Reference	128
7.26.1 Detailed Description	129
7.26.2 Constructor & Destructor Documentation	129
7.26.2.1 RotationComponent()	129
7.26.2.2 ∼RotationComponent()	130
7.26.3 Member Function Documentation	130
7.26.3.1 setAngle()	130
7.26.3.2 getAngle()	130
7.26.3.3 debugPrint()	130
7.27 wEngine::ScaleComponent Class Reference	131
7.27.1 Detailed Description	132
7.27.2 Constructor & Destructor Documentation	132
7.27.2.1 ScaleComponent()	132
7.27.2.2 ∼ScaleComponent()	133
7.27.3 Member Function Documentation	133
7.27.3.1 getScale()	133

7.27.3.2 setScale()	 133
7.27.3.3 debugPrint()	 133
7.28 wEngine::ThicknessComponent Class Reference	 134
7.28.1 Detailed Description	 135
7.28.2 Constructor & Destructor Documentation	 135
7.28.2.1 ThicknessComponent()	 135
$7.28.2.2 \sim Thickness Component() \qquad \dots \qquad \dots \qquad \dots \qquad \dots$	 136
7.28.3 Member Function Documentation	 136
7.28.3.1 getThickness()	 136
7.28.3.2 setThickness()	 136
7.28.3.3 debugPrint()	 136
7.29 wPlot2D::TitleEntity Class Reference	 137
7.29.1 Detailed Description	 139
7.29.2 Constructor & Destructor Documentation	 140
7.29.2.1 TitleEntity() [1/2]	 140
<b>7.29.2.2 TitleEntity()</b> [2/2]	 140
7.29.2.3 ~TitleEntity()	 140
7.29.3 Member Function Documentation	 140
7.29.3.1 getCharacterSize()	 140
7.29.3.2 getTextSize()	 141
7.29.3.3 setTextColor()	 141
7.29.3.4 setOffset()	 141
7.29.3.5 setCharacterSize()	 141
7.29.3.6 setFont()	 141
7.29.3.7 getFrameOutlineColor()	 142
7.29.3.8 getFrameFillColor()	 142
7.29.3.9 getFrameThickness()	 142
7.29.3.10 getPadding()	 142
7.29.3.11 isFrameEnabled()	 143
7.29.3.12 setFrameEnabled()	 143
7.29.3.13 setFrameOutlineColor()	 143
7.29.3.14 setFrameFillColor()	 143
7.29.3.15 setFrameThickness()	 143
7.29.3.16 setPadding()	 144
7.29.3.17 render()	 144
8 File Documentation	145
8.1 main.cpp File Reference	
8.1.1 Function Documentation	
8.1.1.1 main()	
8.2 wColorComponent.cpp File Reference	
8.2.1 Detailed Description	
C.E. I Dotalica Docomption	 170

8.3 wColorComponent.hpp File Reference
8.4 wColorComponent.hpp
8.5 wDiscontinuityComponent.cpp File Reference
8.5.1 Detailed Description
8.6 wDiscontinuityComponent.hpp File Reference
8.7 wDiscontinuityComponent.hpp
8.8 wFontComponent.cpp File Reference
8.8.1 Detailed Description
8.9 wFontComponent.hpp File Reference
8.10 wFontComponent.hpp
8.11 wFunctionComponent.cpp File Reference
8.11.1 Detailed Description
8.12 wFunctionComponent.hpp File Reference
8.13 wFunctionComponent.hpp
8.14 wLengthComponent.cpp File Reference
8.14.1 Detailed Description
8.15 wLengthComponent.hpp File Reference
8.16 wLengthComponent.hpp
8.17 wLineStyleComponent.cpp File Reference
8.17.1 Detailed Description
8.18 wLineStyleComponent.hpp File Reference
8.19 wLineStyleComponent.hpp
8.20 wNotchIntervalComponent.cpp File Reference
8.20.1 Detailed Description
8.21 wNotchIntervalComponent.hpp File Reference
8.22 wNotchIntervalComponent.hpp
8.23 wOffsetComponent.cpp File Reference
8.23.1 Detailed Description
8.24 wOffsetComponent.hpp File Reference
8.25 wOffsetComponent.hpp
8.26 wPaddingComponent.cpp File Reference
8.26.1 Detailed Description
8.27 wPaddingComponent.hpp File Reference
8.28 wPaddingComponent.hpp
8.29 wPositionComponent.cpp File Reference
8.29.1 Detailed Description
8.30 wPositionComponent.hpp File Reference
8.31 wPositionComponent.hpp
8.32 wRotationComponent.cpp File Reference
8.32.1 Detailed Description
8.33 wRotationComponent.hpp File Reference
8.34 wRotationComponent.hpp

8.35 wScaleComponent.cpp File Reference
8.35.1 Detailed Description
8.36 wScaleComponent.hpp File Reference
8.37 wScaleComponent.hpp
8.38 wThicknessComponent.cpp File Reference
8.38.1 Detailed Description
8.39 wThicknessComponent.hpp File Reference
8.40 wThicknessComponent.hpp
8.41 wComponent.cpp File Reference
8.41.1 Detailed Description
8.42 wComponent.hpp File Reference
8.43 wComponent.hpp
8.44 wEntity.cpp File Reference
8.44.1 Detailed Description
8.45 wEntity.hpp File Reference
8.46 wEntity.hpp
8.47 wAxisEntity.cpp File Reference
8.47.1 Detailed Description
8.48 wAxisEntity.hpp File Reference
8.49 wAxisEntity.hpp
8.50 wDataPlotEntity.cpp File Reference
8.50.1 Detailed Description
8.51 wDataPlotEntity.hpp File Reference
8.52 wDataPlotEntity.hpp
8.53 wFrameEntity.cpp File Reference
8.53.1 Detailed Description
8.54 wFrameEntity.hpp File Reference
8.55 wFrameEntity.hpp
8.56 wFunctionEntity.cpp File Reference
8.56.1 Detailed Description
8.57 wFunctionEntity.hpp File Reference
8.58 wFunctionEntity.hpp
8.59 wGraphicsEntity.cpp File Reference
8.59.1 Detailed Description
8.60 wGraphicsEntity.hpp File Reference
8.61 wGraphicsEntity.hpp
8.62 wLabelEntity.cpp File Reference
8.62.1 Detailed Description
8.63 wLabelEntity.hpp File Reference
8.64 wLabelEntity.hpp
8.65 wLegendEntity.cpp File Reference
8.65.1 Detailed Description

	8.66 wLegendEntity.hpp File Reference	202
	8.67 wLegendEntity.hpp	203
	8.68 wLineEntity.cpp File Reference	204
	8.68.1 Detailed Description	205
	8.69 wLineEntity.hpp File Reference	205
	8.70 wLineEntity.hpp	206
	8.71 wNotchEntity.cpp File Reference	207
	8.71.1 Detailed Description	207
	8.72 wNotchEntity.hpp File Reference	208
	8.73 wNotchEntity.hpp	209
	8.74 wTitleEntity.cpp File Reference	209
	8.74.1 Detailed Description	210
	8.75 wTitleEntity.hpp File Reference	210
	8.76 wTitleEntity.hpp	211
	8.77 wAssetManager.cpp File Reference	212
	8.77.1 Detailed Description	212
	8.78 wAssetManager.hpp File Reference	213
	8.79 wAssetManager.hpp	214
	8.80 wLineDrawer.cpp File Reference	214
	8.80.1 Detailed Description	215
	8.81 wLineDrawer.hpp File Reference	215
	8.82 wLineDrawer.hpp	217
	8.83 wMathUtils.cpp File Reference	217
	8.83.1 Detailed Description	218
	8.84 wMathUtils.hpp File Reference	218
	8.85 wMathUtils.hpp	219
	8.86 wPathUtils.cpp File Reference	219
	8.86.1 Detailed Description	220
	8.87 wPathUtils.hpp File Reference	220
	8.88 wPathUtils.hpp	221
le-	dan.	000
IN	dex	223

# wPlot2D - ECS-Based 2D Plotting Engine

## 1.1 wPlot2D

### 1.1.1 Introduction

wPlot2D is a lightweight C++ plotting library designed to create clean and customizable 2D visualizations. It provides essential features such as axes, labels, titles, legends, and annotations, while allowing users to export high-quality graphics for reports, teaching, or research. The library is built with a modular design, making it easy to integrate into existing C++ projects. Its focus is on clarity, precision, and reproducibility, providing an accessible tool for academic and scientific work.

#### 1.1.2 Features

- · Entity and Component system
- · Dynamic component management with type-safe access
- · Support for SFML-based rendering

### 1.1.3 Links

- GitHub Repository
- itch.io Page
- Project Website

#### 1.1.4 Author

Created by Wilfried Koch.

Copyright @ 2025 Wilfried Koch. All rights reserved.

# **Namespace Index**

## 2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

wEngine	 			 		 					 													11
wPlot2D	 			 							 													12

4 Namespace Index

# **Hierarchical Index**

## 3.1 Class Hierarchy

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

wEngine::AssetManager
wEngine::Component
wEngine::ColorComponent
wEngine::DiscontinuityComponent
wEngine::FontComponent
wEngine::FunctionComponent
wEngine::LengthComponent
wEngine::LineStyleComponent
wEngine::NotchIntervalComponent
wEngine::OffsetComponent
wEngine::PaddingComponent
wEngine::PositionComponent
wEngine::RotationComponent
wEngine::ScaleComponent
wEngine::ThicknessComponent
wEngine::Entity
wPlot2D::AxisEntity
wPlot2D::DataPlotEntity
wPlot2D::FrameEntity
wPlot2D::FunctionEntity
wPlot2D::GraphicsEntity
wPlot2D::LabelEntity
wPlot2D::LegendEntity
wPlot2D::LineEntity
wPlot2D::NotchEntity
wPlot2D::TitleEntity
wEngine::LineDrawer
wEngine::MathUtils
wEngine::PathUtils

6 Hierarchical Index

# **Class Index**

## 4.1 Class List

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

wEngine::AssetManager	
Manages graphical assets such as fonts for reuse across the application	15
wPlot2D::AxisEntity	
Represents a visual axis (X or Y) in a 2D plot with optional notches and title	18
wEngine::ColorComponent	
ECS component that holds a color value	28
wEngine::Component	
Abstract base class for all ECS components	31
wPlot2D::DataPlotEntity	
Entity for plotting raw data points as a connected polyline	33
wEngine::DiscontinuityComponent	
ECS component that manages excluded intervals for function plotting	38
wEngine::Entity	
Represents an entity in the ECS (Entity-Component System) architecture	42
wEngine::FontComponent	
Holds a reference to an SFML font for rendering text	47
wPlot2D::FrameEntity	
Entity representing a rectangular frame around content	50
wEngine::FunctionComponent	
ECS component that stores a mathematical function $f(x)$	56
wPlot2D::FunctionEntity	
Represents a mathematical function as a drawable entity in a 2D plot	59
wPlot2D::GraphicsEntity	
Central entity responsible for graphical rendering in wPlot2D	68
wPlot2D::LabelEntity	
Represents a textual label or a collection of axis labels	79
wPlot2D::LegendEntity	
Represents a legend box that describes functions and data plots	86
wEngine::LengthComponent	
ECS component that defines the length of a drawable object	93
wEngine::LineDrawer	
Utility class for rendering thick lines and polylines with style support	96
wPlot2D::LineEntity	
Entity representing a straight line segment with optional arrowhead	98
wEngine::LineStyleComponent	
ESC component that defines the style of a line (solid, dotted, dashed)	105

8 Class Index

wEngine::	:MathUtils	
	Provides common mathematical helper functions for plotting and geometry	109
wPlot2D::	NotchEntity	
	Represents a single tick mark ("notch") on a 2D axis	110
wEngine::	NotchIntervalComponent	
	ECS component that defines the interval between notches on an axis	114
wEngine::	OffsetComponent	
	ECS component that defines a logical coordinate offset	117
wEngine::	PaddingComponent	
	ECS component representing internal padding for UI-like elements	120
wEngine::	PathUtils	
	Utility class providing static functions for managing executable and resource paths across plat-	
1	forms	123
wEngine::	PositionComponent	
	ECS component storing the position of an entity in 2D space and supports movement tracking	125
wEngine::	RotationComponent	
	ECS component that stores a rotation angle (in degrees)	128
_	ScaleComponent	
	ECS component that defines the scaling factor for an entity in 2D space	131
_	ThicknessComponent	
	ECS component that defines the thickness of a drawable object	134
	TitleEntity	
	Represents a textual label (typically an axis title or main plot title) in a 2D plot	137

# **File Index**

## 5.1 File List

Here is a list of all files with brief descriptions:

main.cpp	145
wColorComponent.cpp	
Implementation of the ColorComponent class	146
wColorComponent.hpp	146
wDiscontinuityComponent.cpp	
Implementation of the DiscontinuityComponent class	148
wDiscontinuityComponent.hpp	149
wFontComponent.cpp	
Implementation of the FontComponent class	150
	151
wFunctionComponent.cpp	
	153
·	154
wLengthComponent.cpp	
	155
	156
wLineStyleComponent.cpp	
Implementation of the LineStyleComponent class	158
·	159
wNotchIntervalComponent.cpp	100
	160
·	161
wOffsetComponent.cpp	101
	163
·	164
wPaddingComponent.cpp	104
	165
	166
	100
wPositionComponent.cpp	100
production and a contract of production and a contract of the	168
and the second of the second o	169
wRotationComponent.cpp	470
Implementation of the RotationComponent class	
wRotationComponent.hpp	1/2
wScaleComponent.cpp	. = =
Implementation of the ScaleComponent class	173

10 File Index

wScaleComponent.hpp	 174
wThicknessComponent.cpp	
Implementation of the ThicknessComponent class	176
wThicknessComponent.hpp	 177
wComponent.cpp	
Implementation of the Component class	178
wComponent.hpp	 179
wEntity.cpp	
Implementation of the Entity class	181
wEntity.hpp	 181
wAxisEntity.cpp	405
Implementation of the AxisEntity class	185
wAxisEntity.hpp	 185
wDataPlotEntity.cpp	400
Implementation of the DataPlotEntity class	188
wDataPlotEntity.hpp	 188
wFrameEntity.cpp	100
Implementation of the FrameEntity class	
wFrameEntity.hpp	 191
wFunctionEntity.cpp Implementation of the FunctionEntity class	193
wFunctionEntity.hpp	193
wGraphicsEntity.cpp	 134
Implementation of the GraphicsEntity class	196
wGraphicsEntity.hpp	196
wCataphilosEntity.cpp	 130
Implementation of the LabelEntity class	199
wLabelEntity.hpp	200
wLegendEntity.cpp	 
Implementation of the LegendEntity class	 202
wLegendEntity.hpp	202
wLineEntity.cpp	
Implementation of the LineEntity class	 204
wLineEntity.hpp	 205
wNotchEntity.cpp	
Implementation of the NotchEntity class	 207
wNotchEntity.hpp	 208
wTitleEntity.cpp	
Implementation of the TitleEntity class	 209
wTitleEntity.hpp	 210
wAssetManager.cpp	
Implementation of the AssetManager class	
wAssetManager.hpp	 213
wLineDrawer.cpp	
Implementation of the LineDrawer class	
wLineDrawer.hpp	 215
wMathUtils.cpp	
Implementation of the MathUtils class	
wMathUtils.hpp	 218
wPathUtils.cpp	040
Implementation of the PathUtils class	
wPathUtils.hpp	 220

## **Namespace Documentation**

## 6.1 wEngine Namespace Reference

#### Classes

· class AssetManager

Manages graphical assets such as fonts for reuse across the application.

class ColorComponent

ECS component that holds a color value.

· class Component

Abstract base class for all ECS components.

class DiscontinuityComponent

ECS component that manages excluded intervals for function plotting.

· class Entity

Represents an entity in the ECS (Entity-Component System) architecture.

class FontComponent

Holds a reference to an SFML font for rendering text.

· class FunctionComponent

ECS component that stores a mathematical function f(x).

class LengthComponent

ECS component that defines the length of a drawable object.

class LineDrawer

Utility class for rendering thick lines and polylines with style support.

class LineStyleComponent

ESC component that defines the style of a line (solid, dotted, dashed).

• class MathUtils

Provides common mathematical helper functions for plotting and geometry.

· class NotchIntervalComponent

ECS component that defines the interval between notches on an axis.

· class OffsetComponent

ECS component that defines a logical coordinate offset.

class PaddingComponent

ECS component representing internal padding for UI-like elements.

class PathUtils

Utility class providing static functions for managing executable and resource paths across platforms.

class PositionComponent

ECS component storing the position of an entity in 2D space and supports movement tracking.

class RotationComponent

ECS component that stores a rotation angle (in degrees).

class ScaleComponent

ECS component that defines the scaling factor for an entity in 2D space.

class ThicknessComponent

ECS component that defines the thickness of a drawable object.

#### **Functions**

- std::size\_t getNextComponentTypeID ()
- template<typename ComponentType>
   std::size\_t getComponentTypeID () noexcept

#### 6.1.1 Function Documentation

#### 6.1.1.1 getNextComponentTypeID()

```
std::size_t wEngine::getNextComponentTypeID () [inline]
```

### 6.1.1.2 getComponentTypeID()

```
template<typename ComponentType>
std::size_t wEngine::getComponentTypeID () [noexcept]
```

## 6.2 wPlot2D Namespace Reference

#### Classes

class AxisEntity

Represents a visual axis (X or Y) in a 2D plot with optional notches and title.

class DataPlotEntity

Entity for plotting raw data points as a connected polyline.

class FrameEntity

Entity representing a rectangular frame around content.

class FunctionEntity

Represents a mathematical function as a drawable entity in a 2D plot.

· class GraphicsEntity

Central entity responsible for graphical rendering in wPlot2D.

· class LabelEntity

Represents a textual label or a collection of axis labels.

class LegendEntity

Represents a legend box that describes functions and data plots.

class LineEntity

Entity representing a straight line segment with optional arrowhead.

· class NotchEntity

Represents a single tick mark ("notch") on a 2D axis.

class TitleEntity

Represents a textual label (typically an axis title or main plot title) in a 2D plot.

#### **Enumerations**

enum class AxisType { X\_AXIS , Y\_AXIS }

Enum representing the type of axis to render.

• enum class NotchPosition { Center , Above , Below }

Enum controlling the visual placement of notches relative to the axis.

• enum class TitleAlignment { Top , Bottom }

Defines the vertical placement of the main plot title.

## 6.2.1 Enumeration Type Documentation

## 6.2.1.1 AxisType

```
enum class wPlot2D::AxisType [strong]
```

Enum representing the type of axis to render.

#### Enumerator

X_AXIS	
Y_AXIS	

### 6.2.1.2 NotchPosition

```
enum class wPlot2D::NotchPosition [strong]
```

Enum controlling the visual placement of notches relative to the axis.

#### Enumerator

Center	
Above	
Below	

## 6.2.1.3 TitleAlignment

```
enum class wPlot2D::TitleAlignment [strong]
```

Defines the vertical placement of the main plot title.

## Enumerator

Тор	
Bottom	

## **Class Documentation**

## 7.1 wEngine::AssetManager Class Reference

Manages graphical assets such as fonts for reuse across the application.

#include <wAssetManager.hpp>

#### **Public Member Functions**

- AssetManager ()=default
- AssetManager (const AssetManager &)=delete
- AssetManager & operator= (const AssetManager &)=delete
- ∼AssetManager ()=default
- void LoadFont (const std::string &name, const std::string &fileName)

Loads a font from file and stores it under a given name.

sf::Font & getFont (const std::string &name)

Retrieves a reference to a previously loaded font.

• void RemoveFont (const std::string &name)

Removes a previously loaded font from memory.

• void debugPrintFonts () const

Prints the list of loaded fonts to standard output.

## 7.1.1 Detailed Description

Manages graphical assets such as fonts for reuse across the application.

This class provides a centralized way to load, access, and manage graphical assets, currently supporting fonts via SFML. Assets are identified by string keys and stored internally to avoid reloading them multiple times.

16 Class Documentation

### 7.1.1.0.1 Key features:

- · Load fonts from file and associate them with a name.
- · Access loaded fonts via their name.
- · Remove fonts from memory when no longer needed.
- · Debug printing of loaded assets.

This manager is non-copyable to ensure centralized ownership and avoid accidental duplication of resources.

Note

If an asset fails to load or is accessed without being loaded first, a std::runtime\_error is thrown.

**Author** 

Wilfried Koch

Copyright

© 2025 Wilfried Koch. All rights reserved.

#### 7.1.2 Constructor & Destructor Documentation

#### 7.1.2.1 AssetManager() [1/2]

```
wEngine::AssetManager::AssetManager () [default]
```

## 7.1.2.2 AssetManager() [2/2]

### 7.1.2.3 ~AssetManager()

```
wEngine::AssetManager::~AssetManager () [default]
```

## 7.1.3 Member Function Documentation

#### 7.1.3.1 operator=()

#### 7.1.3.2 LoadFont()

Loads a font from file and stores it under a given name.

If successful, the font is stored under the given name and can later be retrieved with getFont ( name ). If loading fails, an exception is thrown.

### **Parameters**

name	The unique name used to identify the font.
fileName	The path to the font file on disk.

### **Exceptions**

ime error if the font cannot be loaded from file.
---

## 7.1.3.3 getFont()

Retrieves a reference to a previously loaded font.

### **Parameters**

name	The name of the font previously loaded.
------	---

#### Returns

Reference to the corresponding sf::Font object.

## **Exceptions**

ĺ	std::runtime error	if the font does not exist.
	olddillillo cirol	i the fort dood not exist.

## 7.1.3.4 RemoveFont()

Removes a previously loaded font from memory.

#### **Parameters**

name	The name of the font to remove.
------	---------------------------------

## **Exceptions**

std::runtime_error	if the font does not exist.
--------------------	-----------------------------

18 Class Documentation

## 7.1.3.5 debugPrintFonts()

void wEngine::AssetManager::debugPrintFonts () const

Prints the list of loaded fonts to standard output.

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

- wAssetManager.hpp
- wAssetManager.cpp

## 7.2 wPlot2D::AxisEntity Class Reference

Represents a visual axis (X or Y) in a 2D plot with optional notches and title.

#include <wAxisEntity.hpp>

Inheritance diagram for wPlot2D::AxisEntity:

## wEngine::Entity

- + Entity()
- + ~Entity()
- + getEntityID()
- + clearComponents()
- + addComponent()
- + removeComponent()
- + hasComponent()
- + getComponent()
- + requireComponent()
- + getInterfaceComponent()
- + resetEntityIDCounter()

## wPlot2D::AxisEntity

- + AxisEntity()
- + ~AxisEntity()
- + setColor()
- + setThickness()
- + setArrowSize()
- + addTitle()
- + addTitle()
- + setTitleFont()
- + setTitleCharacterSize()
- + setTitleColor()
  - and 15 more...

### **Public Member Functions**

AxisEntity (sf::Font &font, sf::Vector2f origin, sf::Vector2f scale, sf::Vector2f offset, AxisType type, sf::Vector2f axisRange)

Constructs an AxisEntity with a given orientation, origin, scale, and range.

virtual ∼AxisEntity ()=default

Virtual destructor.

void setColor (sf::Color color)

Sets the color of the axis line.

20 Class Documentation

void setThickness (float thickness)

Sets the thickness of the axis line (in pixels).

• void setArrowSize (float arrowSize)

Sets the size of the arrowhead at the end of the axis.

void addTitle (const std::string &title)

Adds a title to the axis.

void addTitle (const std::wstring &title)

Adds a title to the axis.

void setTitleFont (const sf::Font &font)

Sets the font of the axis title.

• void setTitleCharacterSize (unsigned int size)

Sets the character size of the axis title.

void setTitleColor (sf::Color newColor)

Sets the color of the axis title.

void setTitleOffset (sf::Vector2f titleOffset)

Sets a manual offset for the title position.

• sf::Vector2f getTitleOffset () const

Gets the current title offset.

void addNotches (float interval, NotchPosition position, bool hasLabels=false)

Adds notches along the axis.

void setNotchesColor (const sf::Color &color)

Sets the color of all notches.

· void setNotchesThickness (float thickness)

Sets the thickness of all notches.

void setNotchesLength (float newLength)

Sets the length of all notches.

void setLabelsFont (const sf::Font &font)

Sets the font of all labels.

void setLabelsColor (const sf::Color &color)

Sets the color of all labels.

std::vector< sf::Vector2f > getLabelsOffset () const

Gets the current offset of all labels.

void setLabelsOffset (sf::Vector2f offset)

Sets a new offset for all labels.

void addLabelsOffset (sf::Vector2f delta)

Applies an additional offset to all labels.

void setLabelsCharacterSize (unsigned int newSize)

Sets the character size of all labels.

void setLabelsDecimalPlaces (int places)

Sets the number of decimal places for numeric labels.

void setCustomLabels (const std::vector< std::string > &labels)

Replaces numeric labels with a custom set of strings.

• void render (sf::RenderWindow &window)

Renders the axis (line, arrow, title, notches, labels).

## Public Member Functions inherited from wEngine::Entity

```
• Entity ()
```

virtual ~Entity ()

• unsigned int getEntityID () const

Returns the unique ID associated with this entity.

• void clearComponents ()

Removes all components currently attached to the entity.

• template<typename T, typename... Args>

```
std::shared_ptr< T > addComponent (Args &&... args)
```

Adds a new component of type T to the entity.

• template<typename T>

void removeComponent ()

Removes the component of type T from the entity.

• template<typename T>

bool hasComponent () const noexcept

Checks whether the entity has a component of type T.

• template<typename T>

std::shared\_ptr< T > getComponent () const

Retrieves the component of type T attached to the entity.

• template<typename T>

std::shared\_ptr< T > requireComponent (const std::string &context="") const

Retrieves the component of type T and throws if it's missing.

• template<typename Interface>

std::shared\_ptr< Interface > getInterfaceComponent () const

Returns the first component that implements the specified interface.

#### **Additional Inherited Members**

#### Static Public Member Functions inherited from wEngine::Entity

• static void resetEntityIDCounter ()

Resets the global entity ID counter to zero.

## 7.2.1 Detailed Description

Represents a visual axis (X or Y) in a 2D plot with optional notches and title.

This class manages the rendering of a coordinate axis in a Cartesian 2D system. It supports:

- · Rendering of an axis line with an arrowhead.
- Adding notches (tick marks) with optional labels.
- · Attaching a customizable axis title.

See also

TitleEntity, LabelEntity, NotchEntity, LineEntity

**Author** 

Wilfried Koch

#### Copyright

© 2025 Wilfried Koch. All rights reserved.

## 7.2.2 Constructor & Destructor Documentation

#### 7.2.2.1 AxisEntity()

```
wPlot2D::AxisEntity::AxisEntity (
    sf::Font & font,
    sf::Vector2f origin,
    sf::Vector2f scale,
    sf::Vector2f offset,
    AxisType type,
    sf::Vector2f axisRange)
```

Constructs an AxisEntity with a given orientation, origin, scale, and range.

#### **Parameters**

font	Reference to a font used for the title and labels.  Pixel position of the logical origin (typically from GraphicsEntity).	
origin		
scale	Scaling factor (pixels per logical unit).	
offset	Logical displacement of the axis system.	
type	Axis type (X_AXIS or Y_AXIS).	
axisRange Logical range covered by the axis (e.g., [-5, 5]).		

## 7.2.2.2 $\sim$ AxisEntity()

```
virtual wPlot2D::AxisEntity::~AxisEntity () [virtual], [default]
```

Virtual destructor.

## 7.2.3 Member Function Documentation

#### 7.2.3.1 setColor()

Sets the color of the axis line.

#### **Parameters**

#### **Exceptions**

std::runtime_error	if ColorComponent is missing.
--------------------	-------------------------------

## 7.2.3.2 setThickness()

Sets the thickness of the axis line (in pixels).

#### **Parameters**

thickness	New thickness (must be $>$ 0).
-----------	--------------------------------

## **Exceptions**

std::invalid_argument	if thickness $\leq$ = 0.
std::runtime_error	if ThicknessComponent is missing.

#### 7.2.3.3 setArrowSize()

Sets the size of the arrowhead at the end of the axis.

#### **Parameters**

	arrowSize	Arrowhead size in pixels.
--	-----------	---------------------------

### 7.2.3.4 addTitle() [1/2]

Adds a title to the axis.

#### **Parameters**

```
title Title string (narrow string).
```

## 7.2.3.5 addTitle() [2/2]

Adds a title to the axis.

#### **Parameters**

```
title Title string (wide string).
```

## 7.2.3.6 setTitleFont()

Sets the font of the axis title.

#### **Parameters**

```
font Reference to an SFML font.
```

#### 7.2.3.7 setTitleCharacterSize()

Sets the character size of the axis title.

#### **Parameters**

size	Character size in pixels.
------	---------------------------

#### 7.2.3.8 setTitleColor()

Sets the color of the axis title.

#### **Parameters**

newColor	New text color.
----------	-----------------

#### 7.2.3.9 setTitleOffset()

Sets a manual offset for the title position.

#### **Parameters**

```
titleOffset Pixel offset applied to the title position.
```

## 7.2.3.10 getTitleOffset()

```
sf::Vector2f wPlot2D::AxisEntity::getTitleOffset () const [nodiscard]
```

Gets the current title offset.

Returns

Offset vector in pixels.

#### 7.2.3.11 addNotches()

Adds notches along the axis.

#### **Parameters**

interval	Logical spacing between notches (> 0).
position	Placement relative to axis (Center, Above, Below).
hasLabels	If true, labels are displayed for each notch.

## 7.2.3.12 setNotchesColor()

Sets the color of all notches.

#### **Parameters**

color New notch color.
------------------------

## 7.2.3.13 setNotchesThickness()

```
\begin{tabular}{ll} \beg
```

Sets the thickness of all notches.

#### **Parameters**

thickness	New thickness in pixels.
ti iloiti 1000	Trott amorations in pixolo.

## 7.2.3.14 setNotchesLength()

Sets the length of all notches.

#### **Parameters**

newLength	New length in pixels.
-----------	-----------------------

## 7.2.3.15 setLabelsFont()

Sets the font of all labels.

#### **Parameters**

font Reference to an SFML font.

## 7.2.3.16 setLabelsColor()

Sets the color of all labels.

**Parameters** 

color New text color.

## 7.2.3.17 getLabelsOffset()

```
std::vector< sf::Vector2f > wPlot2D::AxisEntity::getLabelsOffset () const [nodiscard]
```

Gets the current offset of all labels.

Returns

Vector of offsets (one per label).

#### 7.2.3.18 setLabelsOffset()

Sets a new offset for all labels.

**Parameters** 

offset Offset vector in pixels.

#### 7.2.3.19 addLabelsOffset()

Applies an additional offset to all labels.

**Parameters** 

delta Delta offset in pixels.

#### 7.2.3.20 setLabelsCharacterSize()

```
void wPlot2D::AxisEntity::setLabelsCharacterSize (
          unsigned int newSize)
```

Sets the character size of all labels.

#### **Parameters**

newSize Character size in pixels.

## 7.2.3.21 setLabelsDecimalPlaces()

Sets the number of decimal places for numeric labels.

#### **Parameters**

aces Digits after decimal point (>= 0).
---

#### 7.2.3.22 setCustomLabels()

Replaces numeric labels with a custom set of strings.

#### **Parameters**

1.	abels	Vector of user-defined label strings.
----	-------	---------------------------------------

#### 7.2.3.23 render()

Renders the axis (line, arrow, title, notches, labels).

#### **Parameters**

window	Target render window.
--------	-----------------------

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

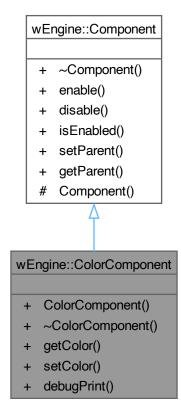
- wAxisEntity.hpp
- wAxisEntity.cpp

# 7.3 wEngine::ColorComponent Class Reference

ECS component that holds a color value.

#include <wColorComponent.hpp>

Inheritance diagram for wEngine::ColorComponent:



## **Public Member Functions**

ColorComponent (sf::Color color=sf::Color::Black)

Constructor with optional color.

- virtual ∼ColorComponent ()=default
- sf::Color getColor () const

Gets the current color.

void setColor (sf::Color newColor)

Sets a new color.

• void debugPrint () const

Outputs the RGBA values of the color to standard output.

## Public Member Functions inherited from wEngine::Component

- virtual ∼Component ()=default
- virtual void enable ()
- virtual void disable ()
- bool isEnabled () const

Checks whether the component is currently active.

void setParent (Entity \*parent)

Sets the parent entity of this component.

• Entity \* getParent () const

Returns the parent entity of this component.

#### **Additional Inherited Members**

## Protected Member Functions inherited from wEngine::Component

· Component ()

Protected constructor to restrict instantiation to derived classes.

## 7.3.1 Detailed Description

ECS component that holds a color value.

This component stores an RGBA color (from SFML) associated with an entity. It can be used to define the rendering color of graphical elements.

#### 7.3.1.0.1 Usage Examples:

· With predefined SFML colors:

```
addComponent< wEngine::ColorComponent >( sf::Color::Red );
```

With a custom RGBA color:

```
sf::Color customColor(128, 64, 200, 255); // R, G, B, A
   addComponent< wEngine::ColorComponent > ( customColor );
```

#### Author

Wilfried Koch

#### Copyright

© 2025 Wilfried Koch. All rights reserved.

## 7.3.2 Constructor & Destructor Documentation

#### 7.3.2.1 ColorComponent()

Constructor with optional color.

#### **Parameters**

```
color The initial color (default is sf::Color::Black).
```

## 7.3.2.2 ~ColorComponent()

```
virtual wEngine::ColorComponent::~ColorComponent () [virtual], [default]
```

## 7.3.3 Member Function Documentation

## 7.3.3.1 getColor()

```
sf::Color wEngine::ColorComponent::getColor () const [nodiscard]
```

Gets the current color.

#### Returns

The color stored in the component.

## 7.3.3.2 setColor()

Sets a new color.

## **Parameters**

## 7.3.3.3 debugPrint()

```
void wEngine::ColorComponent::debugPrint () const
```

Outputs the RGBA values of the color to standard output.

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

- wColorComponent.hpp
- wColorComponent.cpp

## 7.4 wEngine::Component Class Reference

Abstract base class for all ECS components.

#include <wComponent.hpp>

Inheritance diagram for wEngine::Component:



## **Public Member Functions**

- virtual ∼Component ()=default
- virtual void enable ()
- virtual void disable ()
- bool isEnabled () const

Checks whether the component is currently active.

void setParent (Entity \*parent)

Sets the parent entity of this component.

• Entity \* getParent () const

Returns the parent entity of this component.

#### **Protected Member Functions**

· Component ()

Protected constructor to restrict instantiation to derived classes.

## 7.4.1 Detailed Description

Abstract base class for all ECS components.

Defines the minimal interface required by any component: activation control and access to its owning entity.

Intended to be subclassed by specific component implementations.

Created by Wilfried Koch.

Copyright @ 2025 Wilfried Koch. All rights reserved.

## 7.4.2 Constructor & Destructor Documentation

#### 7.4.2.1 ∼Component()

 $\verb|virtual wEngine::Component::\sim|Component () [virtual], [default]|\\$ 

#### 7.4.2.2 Component()

```
wEngine::Component::Component () [protected]
```

Protected constructor to restrict instantiation to derived classes.

#### 7.4.3 Member Function Documentation

## 7.4.3.1 enable()

```
void wEngine::Component::enable () [virtual]
```

#### 7.4.3.2 disable()

```
void wEngine::Component::disable () [virtual]
```

#### 7.4.3.3 isEnabled()

```
bool wEngine::Component::isEnabled () const [nodiscard]
```

Checks whether the component is currently active.

#### Returns

True if enabled, false if disabled.

## 7.4.3.4 setParent()

Sets the parent entity of this component.

#### **Parameters**

parent A pointer to the entity that owns this component.

#### 7.4.3.5 getParent()

```
Entity * wEngine::Component::getParent () const [nodiscard]
```

Returns the parent entity of this component.

#### Returns

Pointer to the owning Entity.

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

- wComponent.hpp
- wComponent.cpp

# 7.5 wPlot2D::DataPlotEntity Class Reference

Entity for plotting raw data points as a connected polyline.

#include <wDataPlotEntity.hpp>

Inheritance diagram for wPlot2D::DataPlotEntity:

## wEngine::Entity

- + Entity()
- + ~Entity()
- + getEntityID()
- + clearComponents()
- + addComponent()
- + removeComponent()
- + hasComponent()
- + getComponent()
- + requireComponent()
- + getInterfaceComponent()
- + resetEntityIDCounter()



# wPlot2D::DataPlotEntity

- + DataPlotEntity()
- + ~DataPlotEntity()
- + getColor()
- + getThickness()
- + getLineStyle()
- + getDashLength()
- + getGapLength()
- + setColor()
- + setThickness()
- + setLineStyle()
- + setDashLength()
- + setGapLength()
- + drawDataPlot()

#### **Public Member Functions**

DataPlotEntity (const sf::Vector2f origin, const sf::Vector2f scale, const std::vector< sf::Vector2f > &data←
 Points)

Constructs a DataPlotEntity with given origin, scale, and raw data points.

virtual ~DataPlotEntity ()=default

Virtual destructor.

sf::Color getColor () const

Get the current line color.

• float getThickness ()

Get the line thickness in pixels.

• wEngine::LineStyleComponent::LineStyle getLineStyle ()

Get the current line style.

float getDashLength ()

Get the dash length for dashed lines.

float getGapLength ()

Get the gap length for dashed/dotted lines.

void setColor (sf::Color color)

Sets the color of the plotted line.

void setThickness (float thickness)

Sets the line thickness in pixels.

• void setLineStyle (wEngine::LineStyleComponent::LineStyle style)

Sets the line style (Solid, Dashed, or Dotted).

void setDashLength (float dashLength)

Sets the dash length for dashed lines.

void setGapLength (float gapLength)

Sets the gap length between dashes or dots.

void drawDataPlot (sf::RenderWindow &window)

Draws the connected data points to the window.

## Public Member Functions inherited from wEngine::Entity

```
• Entity ()
```

- virtual  $\sim$ Entity ()
- unsigned int getEntityID () const

Returns the unique ID associated with this entity.

• void clearComponents ()

Removes all components currently attached to the entity.

 $\bullet \;\; template {<} typename \; T\!, \; typename... \; Args {>} \\$ 

```
std::shared_ptr< T > addComponent (Args &&... args)
```

Adds a new component of type T to the entity.

 $\bullet \ \ template {<} typename \ T{>}$ 

void removeComponent ()

Removes the component of type T from the entity.

• template<typename T>

bool hasComponent () const noexcept

Checks whether the entity has a component of type T.

• template<typename T>

```
std::shared_ptr< T > getComponent () const
```

Retrieves the component of type T attached to the entity.

```
    template < typename T >
        std::shared_ptr < T > requireComponent (const std::string &context="") const
        Retrieves the component of type T and throws if it's missing.
```

• template<typename Interface>

```
std::shared_ptr< Interface > getInterfaceComponent () const
```

Returns the first component that implements the specified interface.

#### **Additional Inherited Members**

## Static Public Member Functions inherited from wEngine::Entity

static void resetEntityIDCounter ()

Resets the global entity ID counter to zero.

## 7.5.1 Detailed Description

Entity for plotting raw data points as a connected polyline.

Stores a vector of raw (x,y) points and draws straight line segments between them. Each segment is rendered using the current line style (Solid, Dashed, Dotted), with configurable color, thickness, dash length, and gap length.

#### Note

Unlike FunctionEntity, this class does not evaluate a function — it directly uses the provided data points. The points are still transformed by the entity's origin and scale before rendering.

#### Author

Wilfried Koch

## Copyright

© 2025 Wilfried Koch. All rights reserved.

#### 7.5.2 Constructor & Destructor Documentation

#### 7.5.2.1 DataPlotEntity()

Constructs a DataPlotEntity with given origin, scale, and raw data points.

#### **Parameters**

origin	Origin of the plot in window coordinates.
scale	Scaling factors applied to x and y values.
dataPoints	Vector of raw (x,y) points to plot.

#### 7.5.2.2 ∼DataPlotEntity()

```
virtual wPlot2D::DataPlotEntity::~DataPlotEntity () [virtual], [default]
```

Virtual destructor.

#### 7.5.3 Member Function Documentation

## 7.5.3.1 getColor()

```
sf::Color wPlot2D::DataPlotEntity::getColor () const [nodiscard]
```

Get the current line color.

Returns

The SFML color used for rendering the polyline.

## 7.5.3.2 getThickness()

```
float wPlot2D::DataPlotEntity::getThickness () [nodiscard]
```

Get the line thickness in pixels.

Returns

Current thickness value.

#### 7.5.3.3 getLineStyle()

```
wEngine::LineStyleComponent::LineStyle wPlot2D::DataPlotEntity::getLineStyle () [nodiscard]
```

Get the current line style.

Returns

Solid, Dashed, or Dotted.

### 7.5.3.4 getDashLength()

```
float wPlot2D::DataPlotEntity::getDashLength () [nodiscard]
```

Get the dash length for dashed lines.

Returns

Dash length in pixels.

### 7.5.3.5 getGapLength()

```
float wPlot2D::DataPlotEntity::getGapLength () [nodiscard]
```

Get the gap length for dashed/dotted lines.

Returns

Gap length in pixels.

## 7.5.3.6 setColor()

Sets the color of the plotted line.

#### **Parameters**

```
color New SFML color.
```

#### 7.5.3.7 setThickness()

Sets the line thickness in pixels.

## **Parameters**

```
thickness Line width.
```

## 7.5.3.8 setLineStyle()

Sets the line style (Solid, Dashed, or Dotted).

## **Parameters**

```
style New line style.
```

## 7.5.3.9 setDashLength()

Sets the dash length for dashed lines.

#### **Parameters**

dashLength l	Length of each dash in pixels.
--------------	--------------------------------

## 7.5.3.10 setGapLength()

Sets the gap length between dashes or dots.

#### **Parameters**

gapterigiti   terigiti of the gap in pixels.	gapLength	Length of the gap in pixels.
--	-----------	------------------------------

#### 7.5.3.11 drawDataPlot()

Draws the connected data points to the window.

The data points are transformed by scale and origin, then connected with styled line segments using LineDrawer ← ::drawLine.

#### **Parameters**

```
window Target SFML render window.
```

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

- wDataPlotEntity.hpp
- wDataPlotEntity.cpp

# 7.6 wEngine::DiscontinuityComponent Class Reference

ECS component that manages excluded intervals for function plotting.

```
#include <wDiscontinuityComponent.hpp>
```

Inheritance diagram for wEngine::DiscontinuityComponent:

# wEngine::Component + ~Component() enable() disable() isEnabled() setParent() getParent() Component() wEngine::DiscontinuityComponent DiscontinuityComponent() ~DiscontinuityComponent() getExcludedIntervals() addExcludedInterval() clearExcludedIntervals() isInExcludedInterval() debugPrint()

#### **Public Member Functions**

- DiscontinuityComponent ()=default
- virtual ~DiscontinuityComponent ()=default
- const std::vector< std::pair< double, double >> & getExcludedIntervals () const

Gives read-only access to the list of excluded intervals.

• void addExcludedInterval (double min, double max)

Adds an excluded interval to the list.

• void clearExcludedIntervals ()

Removes all excluded intervals.

• bool isInExcludedInterval (double x) const

Checks if a value falls into one of the excluded intervals.

void debugPrint () const

## Public Member Functions inherited from wEngine::Component

virtual ∼Component ()=default

- virtual void enable ()
- virtual void disable ()
- bool isEnabled () const

Checks whether the component is currently active.

void setParent (Entity \*parent)

Sets the parent entity of this component.

Entity \* getParent () const

Returns the parent entity of this component.

#### **Additional Inherited Members**

## Protected Member Functions inherited from wEngine::Component

· Component ()

Protected constructor to restrict instantiation to derived classes.

## 7.6.1 Detailed Description

ECS component that manages excluded intervals for function plotting.

This component allows the user to explicitly define intervals of the domain where a function should not be drawn (e.g., around asymptotes or undefined values). During rendering, points falling inside these intervals are skipped to avoid unwanted connections across discontinuities.

**Author** 

Wilfried Koch

Copyright

© 2025 Wilfried Koch. All rights reserved.

## 7.6.2 Constructor & Destructor Documentation

#### 7.6.2.1 DiscontinuityComponent()

 $\label{prop:prop:def:def:weight} w \texttt{Engine::DiscontinuityComponent::DiscontinuityComponent} \ \ () \quad [\texttt{default}]$ 

#### 7.6.2.2 ~DiscontinuityComponent()

virtual wEngine::DiscontinuityComponent::~DiscontinuityComponent () [virtual], [default]

## 7.6.3 Member Function Documentation

#### 7.6.3.1 getExcludedIntervals()

Gives read-only access to the list of excluded intervals.

#### Returns

A constant reference to the vector of (min, max) pairs.

#### 7.6.3.2 addExcludedInterval()

Adds an excluded interval to the list.

#### **Parameters**

min	Lower bound of the interval.
max	Upper bound of the interval.

#### **Exceptions**

```
std::invalid_argument | if min >= max.
```

## 7.6.3.3 clearExcludedIntervals()

```
\verb"void wEngine": \verb"DiscontinuityComponent": \verb"clearExcludedIntervals"" ()
```

Removes all excluded intervals.

#### 7.6.3.4 isInExcludedInterval()

Checks if a value falls into one of the excluded intervals.

#### **Parameters**

```
x Value to test.
```

#### Returns

True if x is inside any excluded interval, false otherwise.

#### 7.6.3.5 debugPrint()

void wEngine::DiscontinuityComponent::debugPrint () const

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

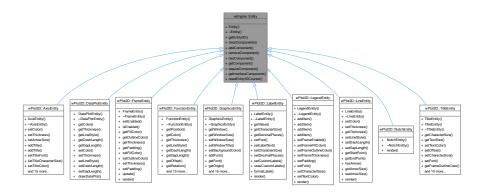
- wDiscontinuityComponent.hpp
- wDiscontinuityComponent.cpp

## 7.7 wEngine::Entity Class Reference

Represents an entity in the ECS (Entity-Component System) architecture.

```
#include <wEntity.hpp>
```

Inheritance diagram for wEngine::Entity:



#### **Public Member Functions**

- Entity ()
- virtual ~Entity ()
- unsigned int getEntityID () const

Returns the unique ID associated with this entity.

• void clearComponents ()

Removes all components currently attached to the entity.

• template<typename T, typename... Args>

std::shared\_ptr< T > addComponent (Args &&... args)

Adds a new component of type T to the entity.

• template<typename T>

void removeComponent ()

Removes the component of type T from the entity.

• template<typename T>

bool hasComponent () const noexcept

Checks whether the entity has a component of type T.

• template<typename T>

std::shared\_ptr< T > getComponent () const

Retrieves the component of type T attached to the entity.

• template<typename T>

std::shared\_ptr< T > requireComponent (const std::string &context="") const

Retrieves the component of type T and throws if it's missing.

• template<typename Interface>

std::shared\_ptr< Interface > getInterfaceComponent () const

Returns the first component that implements the specified interface.

#### **Static Public Member Functions**

static void resetEntityIDCounter ()

Resets the global entity ID counter to zero.

#### 7.7.1 Detailed Description

Represents an entity in the ECS (Entity-Component System) architecture.

Each entity is uniquely identified and can dynamically manage a collection of components. Components are stored in a type-safe map and accessed by type.

The class provides utility methods to add, remove, retrieve and query components, as well as retrieve components through interfaces.

Note

Entities do not define behavior directly: behavior is defined by the components attached to them.

**Author** 

Wilfried Koch

#### Copyright

© 2025 Wilfried Koch. All rights reserved.

#### 7.7.2 Constructor & Destructor Documentation

#### 7.7.2.1 Entity()

```
wEngine::Entity::Entity ()
```

#### 7.7.2.2 ∼Entity()

```
wEngine::Entity::~Entity () [virtual]
```

## 7.7.3 Member Function Documentation

#### 7.7.3.1 getEntityID()

```
unsigned int wEngine::Entity::getEntityID () const [nodiscard]
```

Returns the unique ID associated with this entity.

## Returns

Unsigned integer representing the entity's ID.

#### 7.7.3.2 clearComponents()

```
void wEngine::Entity::clearComponents ()
```

Removes all components currently attached to the entity.

## 7.7.3.3 resetEntityIDCounter()

```
void wEngine::Entity::resetEntityIDCounter () [static]
```

Resets the global entity ID counter to zero.

This affects all subsequently created entities. Use with caution, especially in multi-entity systems.

## 7.7.3.4 addComponent()

Adds a new component of type T to the entity.

Constructs the component using the provided arguments and attaches it to the entity.

#### **Template Parameters**

T	Component type, must inherit from wEngine::Component.
Args	Variadic arguments used to construct the component.

#### **Parameters**

	args	Constructor arguments forwarded to the component.
--	------	---

## Returns

Shared pointer to the newly created component.

## **Exceptions**

std::runtime_error	if a component of the same type already exists in the entity.
--------------------	---

#### 7.7.3.5 removeComponent()

```
template<typename T>
void wEngine::Entity::removeComponent () [inline]
```

Removes the component of type T from the entity.

If no such component exists, this operation does nothing.

#### **Template Parameters**

```
T | Component type to remove.
```

#### 7.7.3.6 hasComponent()

```
template<typename T>
bool wEngine::Entity::hasComponent () const [inline], [nodiscard], [noexcept]
```

Checks whether the entity has a component of type T.

#### **Template Parameters**

```
T Component type to check.
```

#### Returns

True if the component is present, false otherwise.

#### 7.7.3.7 getComponent()

```
template<typename T>
std::shared_ptr< T > wEngine::Entity::getComponent () const [inline], [nodiscard]
```

Retrieves the component of type T attached to the entity.

#### **Template Parameters**

```
T Component type to retrieve.
```

## Returns

Shared pointer to the component if found, or nullptr otherwise.

## 7.7.3.8 requireComponent()

Retrieves the component of type T and throws if it's missing.

This method is similar to <code>getComponent()</code>, but throws a std::runtime\_error if the component is not found. Useful for critical systems where components must be present.

#### **Template Parameters**

The type of the component.	
----------------------------	--

#### **Parameters**

context	Optional string to specify the context of the call (e.g., method name).
---------	---

#### Returns

A shared pointer to the required component.

#### **Exceptions**

```
std::runtime_error if the component is not found.
```

#### 7.7.3.9 getInterfaceComponent()

```
template<typename Interface>
std::shared_ptr< Interface > wEngine::Entity::getInterfaceComponent () const [inline], [nodiscard]
```

Returns the first component that implements the specified interface.

Checks all components attached to the entity using dynamic casting. If a component matches the given interface type, it is returned.

#### **Template Parameters**

#### Returns

A shared pointer to the matching component, or nullptr if none found.

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

- · wEntity.hpp
- · wEntity.cpp

# 7.8 wEngine::FontComponent Class Reference

Holds a reference to an SFML font for rendering text.

#include <wFontComponent.hpp>

Inheritance diagram for wEngine::FontComponent:

# wEngine::Component + ~Component() + enable() + disable() + isEnabled() + setParent() + getParent() # Component() wEngine::FontComponent + FontComponent() + ~FontComponent() + getFont() + setFont()

debugPrint()

## **Public Member Functions**

FontComponent (const sf::Font &font)

Constructs the FontComponent with a reference to the font.

∼FontComponent () override=default

Virtual destructor.

· const sf::Font & getFont () const

Returns the stored font reference.

void setFont (const sf::Font &font)

Updates the stored font reference.

• void debugPrint () const

Prints debug information about the stored font.

## Public Member Functions inherited from wEngine::Component

- virtual ∼Component ()=default
- virtual void enable ()
- virtual void disable ()
- bool isEnabled () const

Checks whether the component is currently active.

void setParent (Entity \*parent)

Sets the parent entity of this component.

• Entity \* getParent () const

Returns the parent entity of this component.

#### **Additional Inherited Members**

## Protected Member Functions inherited from wEngine::Component

· Component ()

Protected constructor to restrict instantiation to derived classes.

## 7.8.1 Detailed Description

Holds a reference to an SFML font for rendering text.

This component allows entities to store and access an sf::Font reference without needing to pass the AssetManager explicitly.

Note

The font must outlive the entity that uses it.

**Author** 

Wilfried Koch

Copyright

© 2025 Wilfried Koch. All rights reserved.

#### 7.8.2 Constructor & Destructor Documentation

## 7.8.2.1 FontComponent()

Constructs the FontComponent with a reference to the font.

#### **Parameters**

font Reference to an externally managed sf::Font.

#### 7.8.2.2 ∼FontComponent()

```
\verb|wEngine::FontComponent::\sim FontComponent () [override], [default]|\\
```

Virtual destructor.

#### 7.8.3 Member Function Documentation

#### 7.8.3.1 getFont()

```
const sf::Font & wEngine::FontComponent::getFont () const
```

Returns the stored font reference.

#### Returns

A constant reference to the sf::Font.

#### 7.8.3.2 setFont()

Updates the stored font reference.

#### **Parameters**

font Reference to an externally managed sf::Font.

Note

The font must outlive this component.

#### 7.8.3.3 debugPrint()

```
void wEngine::FontComponent::debugPrint () const
```

Prints debug information about the stored font.

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

- wFontComponent.hpp
- wFontComponent.cpp

# 7.9 wPlot2D::FrameEntity Class Reference

Entity representing a rectangular frame around content.

#include <wFrameEntity.hpp>

Inheritance diagram for wPlot2D::FrameEntity:

## wEngine::Entity

- + Entity()
- + ~Entity()
- + getEntityID()
- + clearComponents()
- + addComponent()
- + removeComponent()
- + hasComponent()
- + getComponent()
- + requireComponent()
- + getInterfaceComponent()
- + resetEntityIDCounter()



# wPlot2D::FrameEntity

- + FrameEntity()
- + ~FrameEntity()
- + setEnabled()
- + isEnabled()
- + getFillColor()
- + getOutlineColor()
- + getThickness()
- + getPadding()
- + setFillColor()
- + setOutlineColor()
- + setThickness()
- + setPadding()
- + update()
- + render()

#### **Public Member Functions**

FrameEntity (bool enabled=true)

Constructs a frame entity.

virtual ∼FrameEntity ()=default

Virtual destructor.

void setEnabled (bool enabled)

Enables or disables the frame.

· bool isEnabled () const

Checks whether the frame is currently enabled.

• sf::Color getFillColor () const

Gets the current fill color of the frame.

• sf::Color getOutlineColor () const

Gets the current outline color of the frame.

• float getThickness () const

Gets the current outline thickness of the frame.

sf::Vector2f getPadding () const

Gets the current padding applied around the content.

void setFillColor (const sf::Color &color)

Sets the fill color of the frame.

void setOutlineColor (const sf::Color &color)

Sets the outline color of the frame.

void setThickness (float thickness)

Sets the outline thickness of the frame.

void setPadding (const sf::Vector2f &padding)

Sets the padding around the content.

• void update (const sf::FloatRect &contentBounds, const sf::Vector2f &position)

Updates the size and position of the frame based on content bounds.

void render (sf::RenderWindow &window)

Renders the frame to the given render window.

#### Public Member Functions inherited from wEngine::Entity

- Entity ()
- virtual ~Entity ()
- · unsigned int getEntityID () const

Returns the unique ID associated with this entity.

• void clearComponents ()

Removes all components currently attached to the entity.

• template<typename T, typename... Args>

std::shared\_ptr< T > addComponent (Args &&... args)

Adds a new component of type T to the entity.

• template<typename T>

void removeComponent ()

Removes the component of type T from the entity.

• template<typename T>

bool hasComponent () const noexcept

Checks whether the entity has a component of type T.

• template<typename T>

 $std::shared\_ptr < T > getComponent$  () const

Retrieves the component of type T attached to the entity.

```
    template<typename T>
        std::shared_ptr< T > requireComponent (const std::string &context="") const
        Retrieves the component of type T and throws if it's missing.
    template<typename Interface>
        std::shared_ptr< Interface > getInterfaceComponent () const
        Returns the first component that implements the specified interface.
```

#### **Additional Inherited Members**

# Static Public Member Functions inherited from wEngine::Entity

static void resetEntityIDCounter ()
 Resets the global entity ID counter to zero.

## 7.9.1 Detailed Description

Entity representing a rectangular frame around content.

A FrameEntity is typically used to visually surround titles, legends, or other graphical content. It supports:

- Toggle visibility (enabled flag),
- · Fill and outline colors,
- · Outline thickness,
- · Padding around the content.

The size of the frame is dynamically updated from the content bounds (see update ( )).

**Author** 

Wilfried Koch

Copyright

© 2025 Wilfried Koch. All rights reserved.

## 7.9.2 Constructor & Destructor Documentation

## 7.9.2.1 FrameEntity()

Constructs a frame entity.

#### **Parameters**

enabled Whether the frame should be enabled initially.

## 7.9.2.2 $\sim$ FrameEntity()

```
virtual wPlot2D::FrameEntity::~FrameEntity () [virtual], [default]
```

Virtual destructor.

#### 7.9.3 Member Function Documentation

#### 7.9.3.1 setEnabled()

```
void wPlot2D::FrameEntity::setEnabled (
          bool enabled)
```

Enables or disables the frame.

#### **Parameters**

enabled	New enabled state.
---------	--------------------

## 7.9.3.2 isEnabled()

```
bool wPlot2D::FrameEntity::isEnabled () const [nodiscard]
```

Checks whether the frame is currently enabled.

Returns

True if the frame is enabled, false otherwise.

## 7.9.3.3 getFillColor()

```
sf::Color wPlot2D::FrameEntity::getFillColor () const [nodiscard]
```

Gets the current fill color of the frame.

Returns

Fill color.

#### 7.9.3.4 getOutlineColor()

```
sf::Color wPlot2D::FrameEntity::getOutlineColor () const [nodiscard]
```

Gets the current outline color of the frame.

Returns

Outline color.

#### 7.9.3.5 getThickness()

```
float wPlot2D::FrameEntity::getThickness () const [nodiscard]
```

Gets the current outline thickness of the frame.

Returns

Outline thickness (in pixels).

## 7.9.3.6 getPadding()

```
sf::Vector2f wPlot2D::FrameEntity::getPadding () const [nodiscard]
```

Gets the current padding applied around the content.

Returns

Padding as (x, y).

## 7.9.3.7 setFillColor()

Sets the fill color of the frame.

**Parameters** 

```
color New fill color.
```

#### 7.9.3.8 setOutlineColor()

Sets the outline color of the frame.

#### **Parameters**

## 7.9.3.9 setThickness()

```
void wPlot2D::FrameEntity::setThickness ( {\tt float} \ thickness)
```

Sets the outline thickness of the frame.

#### **Parameters**

outline thickness (must be	thickness	New outline thickness (mu
----------------------------	-----------	---------------------------

#### 7.9.3.10 setPadding()

Sets the padding around the content.

#### **Parameters**

padding	Padding as (x, y).
---------	--------------------

## 7.9.3.11 update()

Updates the size and position of the frame based on content bounds.

#### **Parameters**

contentBounds	The bounding box of the content (width/height).
position	The position of the frame's center in pixels.

#### 7.9.3.12 render()

Renders the frame to the given render window.

#### **Parameters**

window	Target render window.
--------	-----------------------

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

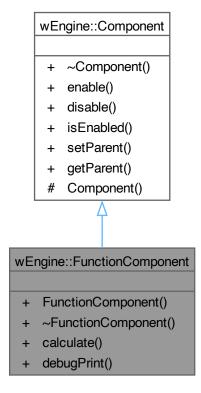
- wFrameEntity.hpp
- wFrameEntity.cpp

# 7.10 wEngine::FunctionComponent Class Reference

ECS component that stores a mathematical function f(x).

#include <wFunctionComponent.hpp>

Inheritance diagram for wEngine::FunctionComponent:



#### **Public Member Functions**

- FunctionComponent (std::function< double(double) > function)

  Constructs a FunctionComponent with a given function.
- virtual ~FunctionComponent ()=default
- double calculate (double x) const

Evaluates the stored function at a given x.

• void debugPrint () const

# Public Member Functions inherited from wEngine::Component

- virtual ∼Component ()=default
- virtual void enable ()
- virtual void disable ()
- bool isEnabled () const

Checks whether the component is currently active.

void setParent (Entity \*parent)

Sets the parent entity of this component.

• Entity \* getParent () const

Returns the parent entity of this component.

#### **Additional Inherited Members**

# Protected Member Functions inherited from wEngine::Component

· Component ()

Protected constructor to restrict instantiation to derived classes.

# 7.10.1 Detailed Description

ECS component that stores a mathematical function f(x).

This component wraps a std::function < double ( double ) > and provides an interface to evaluate the function at any given x-coordinate. It is mainly used by FunctionEntity to render mathematical curves.

Author

Wilfried Koch

# Copyright

© 2025 Wilfried Koch. All rights reserved.

### 7.10.2 Constructor & Destructor Documentation

### 7.10.2.1 FunctionComponent()

Constructs a FunctionComponent with a given function.

### **Parameters**

function A callable object of type double (double).

# **Exceptions**

std::invalid_argument	if the provided function is empty.
-----------------------	------------------------------------

# 7.10.2.2 ~FunctionComponent()

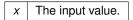
```
virtual wEngine::FunctionComponent::~FunctionComponent () [virtual], [default]
```

### 7.10.3 Member Function Documentation

# 7.10.3.1 calculate()

Evaluates the stored function at a given x.

### **Parameters**



# Returns

The result f(x).

# **Exceptions**

```
std::runtime_error if no function is set.
```

# 7.10.3.2 debugPrint()

```
void wEngine::FunctionComponent::debugPrint () const
```

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

- wFunctionComponent.hpp
- wFunctionComponent.cpp

# 7.11 wPlot2D::FunctionEntity Class Reference

Represents a mathematical function as a drawable entity in a 2D plot.

#include <wFunctionEntity.hpp>

Inheritance diagram for wPlot2D::FunctionEntity:

# wEngine::Entity

- + Entity()
- + ~Entity()
- + getEntityID()
- + clearComponents()
- + addComponent()
- + removeComponent()
- + hasComponent()
- + getComponent()
- + requireComponent()
- + getInterfaceComponent()
- + resetEntityIDCounter()



# wPlot2D::FunctionEntity

- + FunctionEntity()
- + ~FunctionEntity()
- + getPosition()
- + getColor()
- + getThickness()
- + getLineStyle()
- + getDashLength()
- + getGapLength()
- + getOffset()
- + getRotation()
  - and 13 more...

#### **Public Member Functions**

• FunctionEntity (const sf::Vector2f origin, const sf::Vector2f scale, std::function< double(double) > func)

Construct a new FunctionEntity.

virtual ~FunctionEntity ()=default

Virtual destructor.

• sf::Vector2f getPosition () const

Get the position (origin) of the function in pixel space.

• sf::Color getColor () const

Get the color of the function curve.

float getThickness () const

Get the line thickness of the function curve.

• wEngine::LineStyleComponent::LineStyle getLineStyle () const

Get the line style of the function curve.

· float getDashLength () const

Get the dash length when the line style is Dashed.

• float getGapLength () const

Get the gap length between dashes or dots.

sf::Vector2f getOffset () const

Get the current offset applied to the function curve.

• float getRotation () const

Get the current rotation angle of the function curve.

void setPosition (sf::Vector2f position)

Set the position (origin) of the function in pixel space.

void setColor (sf::Color color)

Set the color of the function curve.

void setThickness (float thickness)

Set the line thickness of the function curve.

void setLineStyle (wEngine::LineStyleComponent::LineStyle style)

Set the line style of the function curve.

void setDashLength (float dashLength)

Set the length of each dash when the line style is Dashed.

void setGapLength (float gapLength)

Set the length of the gap between dashes or dots.

void setOffset (float offsetX, float offsetY)

Set an offset applied to the function curve.

• void setRotation (float angleDegrees)

Set the rotation angle of the function curve.

void setScale (sf::Vector2f scale)

Sets the scaling factors for the function graph.

void addExcludedInterval (double min, double max)

Add an excluded interval where the function should not be drawn.

• void clearExcludedIntervals ()

Clear all excluded intervals.

void alignToYAxis (float normalizedOffsetX=0.0f, float normalizedOffsetY=0.0f)

Rotates the function by 90 degrees and swaps scales accordingly.

• void drawFunction (sf::RenderWindow &window, double startX, double endX, size\_t nbPoints=1000)

Draw the function on the target window.

# Public Member Functions inherited from wEngine::Entity

```
• Entity ()
```

- virtual ~Entity ()
- unsigned int getEntityID () const

Returns the unique ID associated with this entity.

• void clearComponents ()

Removes all components currently attached to the entity.

• template<typename T, typename... Args>

```
std::shared_ptr< T > addComponent (Args &&... args)
```

Adds a new component of type T to the entity.

• template<typename T>

void removeComponent ()

Removes the component of type T from the entity.

template<typename T>

bool hasComponent () const noexcept

Checks whether the entity has a component of type T.

• template<typename T>

std::shared\_ptr< T > getComponent () const

Retrieves the component of type T attached to the entity.

• template<typename T>

std::shared\_ptr< T > requireComponent (const std::string &context="") const

Retrieves the component of type T and throws if it's missing.

• template<typename Interface>

std::shared\_ptr< Interface > getInterfaceComponent () const

Returns the first component that implements the specified interface.

# **Additional Inherited Members**

# Static Public Member Functions inherited from wEngine::Entity

• static void resetEntityIDCounter ()

Resets the global entity ID counter to zero.

# 7.11.1 Detailed Description

Represents a mathematical function as a drawable entity in a 2D plot.

A FunctionEntity manages all components required to render a curve:

- Origin and scale (mapping logical space to pixels).
- · Color, thickness, and line style (solid, dashed, dotted).
- · Offset and rotation of the curve.
- · Discontinuities handled by excluded intervals.

The function is sampled at evenly spaced x-values and rendered as a polyline. Excluded intervals and invalid values (NaN, Inf) split the curve into separate segments.

Author

Wilfried Koch

### Copyright

© 2025 Wilfried Koch. All rights reserved.

# 7.11.2 Constructor & Destructor Documentation

### 7.11.2.1 FunctionEntity()

Construct a new FunctionEntity.

### **Parameters**

origin	Logical origin of the coordinate system (pixels).
scale	Scale factors for x and y (pixels per unit).
func	Function of type double(double) to be plotted.

# 7.11.2.2 ~FunctionEntity()

```
virtual wPlot2D::FunctionEntity::~FunctionEntity () [virtual], [default]
```

Virtual destructor.

# 7.11.3 Member Function Documentation

# 7.11.3.1 getPosition()

```
sf::Vector2f wPlot2D::FunctionEntity::getPosition () const [nodiscard]
```

Get the position (origin) of the function in pixel space.

# Returns

The current origin as an sf::Vector2f.

# 7.11.3.2 getColor()

```
sf::Color wPlot2D::FunctionEntity::getColor () const [nodiscard]
```

Get the color of the function curve.

### Returns

The current curve color.

# 7.11.3.3 getThickness()

float wPlot2D::FunctionEntity::getThickness () const [nodiscard]

Get the line thickness of the function curve.

# Returns

The thickness in pixels.

# 7.11.3.4 getLineStyle()

wEngine::LineStyleComponent::LineStyle wPlot2D::FunctionEntity::getLineStyle () const [nodiscard]

Get the line style of the function curve.

### Returns

The current line style (Solid, Dashed, or Dotted).

# 7.11.3.5 getDashLength()

float wPlot2D::FunctionEntity::getDashLength () const [nodiscard]

Get the dash length when the line style is Dashed.

#### Returns

Dash length in pixels.

Note

This value has no effect if the style is not Dashed.

# 7.11.3.6 getGapLength()

float wPlot2D::FunctionEntity::getGapLength () const [nodiscard]

Get the gap length between dashes or dots.

# Returns

Gap length in pixels.

### Note

This setting affects both Dashed and Dotted line styles.

# 7.11.3.7 getOffset()

```
sf::Vector2f wPlot2D::FunctionEntity::getOffset () const [nodiscard]
```

Get the current offset applied to the function curve.

Returns

A 2D vector containing the (x,y) offset in pixels.

# 7.11.3.8 getRotation()

```
float wPlot2D::FunctionEntity::getRotation () const [nodiscard]
```

Get the current rotation angle of the function curve.

Returns

The rotation angle in degrees.

### 7.11.3.9 setPosition()

Set the position (origin) of the function in pixel space.

**Parameters** 

position The new orig	in as an sf::Vector2f.
-----------------------	------------------------

# 7.11.3.10 setColor()

Set the color of the function curve.

**Parameters** 

```
color The new color as an sf::Color.
```

# 7.11.3.11 setThickness()

Set the line thickness of the function curve.

#### **Parameters**

thickness	The new line thickness in pixels.
-----------	-----------------------------------

# 7.11.3.12 setLineStyle()

Set the line style of the function curve.

### **Parameters**

```
style The new style (Solid, Dashed, or Dotted).
```

# 7.11.3.13 setDashLength()

```
\begin{tabular}{ll} \begin{tabular}{ll} void & wPlot2D::FunctionEntity::setDashLength ( \\ & float & dashLength) \end{tabular}
```

Set the length of each dash when the line style is Dashed.

### **Parameters**

dashLength	The dash length in pixels (must be $> 0$ ).

# **Exceptions**

```
std::invalid_argument | if dashLength <= 0.
```

Note

This setting has no effect if the line style is not Dashed.

# 7.11.3.14 setGapLength()

Set the length of the gap between dashes or dots.

### **Parameters**

gapLength	The gap length in pixels (must be $\geq$ = 0).
-----------	--

# **Exceptions**

std::invalid_argument	if $gapLength < 0$ .
-----------------------	----------------------

Note

This setting affects both Dashed and Dotted line styles.

# 7.11.3.15 setOffset()

Set an offset applied to the function curve.

The offset is applied after scaling and rotation, allowing the curve to be shifted horizontally and vertically relative to its logical origin.

### **Parameters**

offsetX	Horizontal offset in pixels.
offsetY	Vertical offset in pixels.

# 7.11.3.16 setRotation()

Set the rotation angle of the function curve.

The rotation is applied around the logical origin of the graph. The angle is expressed in degrees, with positive values corresponding to counter-clockwise rotation.

#### **Parameters**

angleDegrees	Rotation angle in degrees.
--------------	----------------------------

# 7.11.3.17 setScale()

Sets the scaling factors for the function graph.

This method updates the ScaleComponent so the user can control how much the x and y coordinates are stretched on screen.

#### **Parameters**

	scale	The new scaling vector (scale.x, scale.y).
-	Scare	The new scaling vector (scale.x, scale.y).

# 7.11.3.18 addExcludedInterval()

Add an excluded interval where the function should not be drawn.

Useful to handle discontinuities such as vertical asymptotes.

#### **Parameters**

min	Left bound of the interval.
max	Right bound of the interval.

# 7.11.3.19 clearExcludedIntervals()

```
void wPlot2D::FunctionEntity::clearExcludedIntervals ()
```

Clear all excluded intervals.

# 7.11.3.20 alignToYAxis()

Rotates the function by 90 degrees and swaps scales accordingly.

This is a common operation when we want to interpret the function's values along the X-axis instead of the Y-axis (or vice versa). The method:

- Sets a 90 degrees rotation.
- Swaps scale.x and scale.y to preserve unit consistency.
- Applies an optional normalized offset for alignment.

### **Parameters**

normalizedOffsetX	Relative offset along X after rotation, expressed in units of (oldScaleX / oldScaleY).
normalizedOffsetY	Relative offset along Y after rotation, expressed in units of (oldScaleY / oldScaleX).

### 7.11.3.21 drawFunction()

Draw the function on the target window.

# **Parameters**

window	Render target.
startX	Start of the logical x-range.
endX	End of the logical x-range.
nbPoints	Number of points to sample (default: 1000).

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

- wFunctionEntity.hpp
- wFunctionEntity.cpp

# 7.12 wPlot2D::GraphicsEntity Class Reference

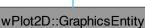
Central entity responsible for graphical rendering in wPlot2D.

#include <wGraphicsEntity.hpp>

Inheritance diagram for wPlot2D::GraphicsEntity:

# wEngine::Entity

- + Entity()
- + ~Entity()
- + getEntityID()
- + clearComponents()
- + addComponent()
- + removeComponent()
- + hasComponent()
- + getComponent()
- + requireComponent()
- + getInterfaceComponent()
- + resetEntityIDCounter()



- + GraphicsEntity()
- + ~GraphicsEntity()
- + getWindow()
- + getWindowSize()
- + setWindowSize()
- + setWindowTitle()
- + setBackgroundColor()
- + addFont()
- + getFont()
- + getOrigin()
  - and 15 more...

# **Public Member Functions**

• GraphicsEntity (const std::string &windowTitle="wPlot2D", const sf::Vector2u &windowSize={ 1600, 1600 }, const sf::Vector2f &originFactor={ 0.5f, 0.5f }, const sf::Vector2f &scaleFactor={ 0.1f, 0.1f })

Constructs the graphics entity and initializes the rendering window and components.

• virtual  $\sim$ GraphicsEntity ()=default

Virtual destructor.

• sf::RenderWindow & getWindow ()

Gives access to the internal SFML window.

• sf::Vector2u getWindowSize () const

Retrieves the current window size.

void setWindowSize (const sf::Vector2u &newSize)

Sets a new window size.

void setWindowTitle (const std::string &title)

Updates the window title.

void setBackgroundColor (const sf::Color &color)

Clears the window with a background color.

void addFont (const std::string &name, const std::string &fileName)

Adds a font to the AssetManager.

sf::Font & getFont (const std::string name)

Retrieves a previously loaded font.

• sf::Vector2f getOrigin () const

Returns the current logical origin (in pixels).

void setOrigin (sf::Vector2f originFactor)

Sets a new logical origin (normalized).

• sf::Vector2f getScale () const

Returns the scale factors (pixels per logical unit).

void setScale (sf::Vector2f scaleFactor)

Sets new scale factors (normalized).

• sf::Vector2f getOffset () const

Returns the current logical offset.

void setOffset (sf::Vector2f offset)

Sets the logical offset applied to the axes.

AxisEntity \* addAxis (AxisType type, sf::Vector2f axisRange)

Adds an axis (X or Y) to the scene.

TitleEntity \* addTitle (const std::string &title, TitleAlignment alignment=TitleAlignment::Bottom)

Adds a main plot title (top or bottom).

TitleEntity \* addTitle (const std::wstring &title, TitleAlignment alignment=TitleAlignment::Bottom)

Adds a main plot title (top or bottom).

FunctionEntity \* addFunction (std::function < double(double) > func, double startX, double endX, size\_t nb
 — Points=1000)

Adds a mathematical function to the scene.

DataPlotEntity \* addDataPlot (const std::vector < sf::Vector2f > &dataPoints)

Adds a raw data plot (connected points).

• LegendEntity \* addLegend (const sf::Vector2f &position, bool hasFrame=true)

Adds a legend box at a given position.

TitleEntity \* addText (const std::string &text, sf::Vector2f position)

Adds arbitrary text to the scene.

• TitleEntity \* addText (const std::wstring &text, sf::Vector2f position)

Adds arbitrary text to the scene.

LineEntity \* addLine (const sf::Vector2f &start, const sf::Vector2f &end, bool withArrow=false)

Adds a line segment to the scene.

• void saveToFile (const std::string &filename)

Saves a screenshot of the current window.

# Public Member Functions inherited from wEngine::Entity

```
• Entity ()

    virtual ∼Entity ()

· unsigned int getEntityID () const
     Returns the unique ID associated with this entity.

    void clearComponents ()

     Removes all components currently attached to the entity.
• template<typename T, typename... Args>
  std::shared_ptr< T > addComponent (Args &&... args)
     Adds a new component of type T to the entity.
• template<typename T>
  void removeComponent ()
     Removes the component of type T from the entity.
• template<typename T>
  bool hasComponent () const noexcept
      Checks whether the entity has a component of type T.
• template<typename T>
  std::shared_ptr< T > getComponent () const
     Retrieves the component of type T attached to the entity.
• template<typename T>
  std::shared_ptr< T > requireComponent (const std::string &context="") const
     Retrieves the component of type T and throws if it's missing.
```

# Additional Inherited Members

• template<typename Interface>

# Static Public Member Functions inherited from wEngine::Entity

std::shared\_ptr< Interface > getInterfaceComponent () const Returns the first component that implements the specified interface.

static void resetEntityIDCounter ()
 Resets the global entity ID counter to zero.

# 7.12.1 Detailed Description

Central entity responsible for graphical rendering in wPlot2D.

The GraphicsEntity manages the creation and control of the SFML rendering window. It provides high-level methods to add and configure graphical entities:

- Axes (X and Y),
- Titles (main or custom text),
- · Functions and raw data plots,
- · Legends,
- · Lines (with or without arrows).

It also handles window configuration (title, size, background color) and allows exporting the final rendering to an image file.

#### Note

This class is intended to be the main entry point for user interaction with the rendering system.

#### **Author**

Wilfried Koch

### Copyright

© 2025 Wilfried Koch. All rights reserved.

### 7.12.2 Constructor & Destructor Documentation

### 7.12.2.1 GraphicsEntity()

Constructs the graphics entity and initializes the rendering window and components.

### **Parameters**

windowTitle	The title displayed on the window (default: "wPlot2D").
windowSize	Window dimensions in pixels (default: 1600×1600).
originFactor	Normalized factor in [0,1]x[0,1] specifying the origin's relative position (default: ( 0.5f, 0.5f ).
scaleFactor	Normalized factor specifying the size of one logical unit relative to window dimensions (default: (0.1f, 0.1f)).

# **Exceptions**

std::invalid argument	if originFactor not in [0,1].

# 7.12.2.2 $\sim$ Graphics Entity()

```
\label{eq:continuous} \mbox{virtual wPlot2D::GraphicsEntity::$$\sim$GraphicsEntity () [virtual], [default]$}
```

### 7.12.3 Member Function Documentation

# 7.12.3.1 getWindow()

Virtual destructor.

```
sf::RenderWindow & wPlot2D::GraphicsEntity::getWindow () [nodiscard]
```

Gives access to the internal SFML window.

# Returns

Reference to the internal sf::RenderWindow.

### 7.12.3.2 getWindowSize()

```
sf::Vector2u wPlot2D::GraphicsEntity::getWindowSize () const [nodiscard]
```

Retrieves the current window size.

Returns

Window size in pixels.

# 7.12.3.3 setWindowSize()

Sets a new window size.

**Parameters** 

newSize Window dimensions in pixels.

# 7.12.3.4 setWindowTitle()

Updates the window title.

**Parameters** 

title New window title.

# 7.12.3.5 setBackgroundColor()

Clears the window with a background color.

**Parameters** 

```
color Background fill color.
```

# 7.12.3.6 addFont()

Adds a font to the AssetManager.

### **Parameters**

name	Identifier string for the font.
fileName	Path to the font file.

# 7.12.3.7 getFont()

Retrieves a previously loaded font.

### **Parameters**

ame Identif	fier of the font.
-------------	-------------------

# Returns

Reference to the sf::Font.

# **Exceptions**

std::runtime_error	if the font is not found.
--------------------	---------------------------

# 7.12.3.8 getOrigin()

```
sf::Vector2f wPlot2D::GraphicsEntity::getOrigin () const [nodiscard]
```

Returns the current logical origin (in pixels).

# Returns

Origin position in pixel coordinates.

# **Exceptions**

```
std::runtime_error if the PositionComponent is missing.
```

# 7.12.3.9 setOrigin()

Sets a new logical origin (normalized).

# **Parameters**

	originFactor	in [0,1]×[0,1] new relative origin.
--	--------------	-------------------------------------

# **Exceptions**

std::invalid_argument   if originFactor is outside [	0,1].
--	-------

# 7.12.3.10 getScale()

```
sf::Vector2f wPlot2D::GraphicsEntity::getScale () const [nodiscard]
```

Returns the scale factors (pixels per logical unit).

Returns

Scaling vector.

# **Exceptions**

std::runtime_error i	if the ScaleComponent is missing.
----------------------	-----------------------------------

# 7.12.3.11 setScale()

Sets new scale factors (normalized).

### **Parameters**

```
scaleFactor new scaling factor
```

# 7.12.3.12 getOffset()

```
sf::Vector2f wPlot2D::GraphicsEntity::getOffset () const [nodiscard]
```

Returns the current logical offset.

Returns

Offset vector in logical units.

# **Exceptions**

std::runtime_error	if the OffsetComponent is missing.	
--------------------	------------------------------------	--

# 7.12.3.13 setOffset()

Sets the logical offset applied to the axes.

#### **Parameters**

offset	Displacement in logical units.
--------	--------------------------------

# 7.12.3.14 addAxis()

Adds an axis (X or Y) to the scene.

#### **Parameters**

type	Axis type.
axisRange	Logical range for the axis.

### Returns

Pointer to the created AxisEntity.

# 7.12.3.15 addTitle() [1/2]

Adds a main plot title (top or bottom).

### **Parameters**

title	Title text (UTF-8).
alignment	Vertical alignment (default: bottom).

# Returns

Pointer to the created TitleEntity.

# 7.12.3.16 addTitle() [2/2]

Adds a main plot title (top or bottom).

#### **Parameters**

title	Title text (UTF-16/32).
alignment	Vertical alignment (default: bottom).

#### Returns

Pointer to the created TitleEntity.

# 7.12.3.17 addFunction()

```
FunctionEntity * wPlot2D::GraphicsEntity::addFunction (
    std::function< double(double) > func,
    double startX,
    double endX,
    size_t nbPoints = 1000) [nodiscard]
```

Adds a mathematical function to the scene.

### **Parameters**

func	Function of type double(double).
startX	Domain start (logical).
endX	Domain end (logical).
nbPoints	Sampling resolution (default 1000).

### Returns

Pointer to the created FunctionEntity.

# 7.12.3.18 addDataPlot()

Adds a raw data plot (connected points).

#### **Parameters**

dataPoints	Vector of (x,y) coordinates.
------------	------------------------------

# Returns

Pointer to the created DataPlotEntity.

# 7.12.3.19 addLegend()

Adds a legend box at a given position.

#### **Parameters**

position	Normalized position inside window [0,1]x[0,1].
hasFrame	Whether the legend frame is visible (default true).

### Returns

Pointer to the created LegendEntity.

# 7.12.3.20 addText() [1/2]

Adds arbitrary text to the scene.

#### **Parameters**

text	Text string (UTF-8).
position	Normalized position in [0,1]x[0,1].

#### Returns

Pointer to the created TitleEntity.

# 7.12.3.21 addText() [2/2]

Adds arbitrary text to the scene.

# **Parameters**

text	Text string (UTF-16/32).
position	Normalized position in $[0,1]x[0,1]$ .

### Returns

Pointer to the created TitleEntity.

# 7.12.3.22 addLine()

Adds a line segment to the scene.

#### **Parameters**

start	Start point in logical coordinates.
end	End point in logical coordinates.
withArrow	Whether to render an arrowhead at the end.

# Returns

Pointer to the created LineEntity.

# 7.12.3.23 saveToFile()

Saves a screenshot of the current window.

# **Parameters**

filename Output file path (supported: png, bmp,	a, jpg).
---	----------

# **Exceptions**

std::runtime_error	if saving fails.
--------------------	------------------

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

- · wGraphicsEntity.hpp
- · wGraphicsEntity.cpp

# 7.13 wPlot2D::LabelEntity Class Reference

Represents a textual label or a collection of axis labels.

```
#include <wLabelEntity.hpp>
```

Inheritance diagram for wPlot2D::LabelEntity:

# wEngine::Entity

- + Entity()
- + ~Entity()
- + getEntityID()
- + clearComponents()
- + addComponent()
- + removeComponent()
- + hasComponent()
- + getComponent()
- + requireComponent()
- + getInterfaceComponent()
- + resetEntityIDCounter()



# wPlot2D::LabelEntity

- + LabelEntity()
- + ~LabelEntity()
- + getValue()
- + getCharacterSize()
- + getDecimalPlaces()
- + setFont()
- + setLabelText()
- + setCharacterSize()
- + setDecimalPlaces()
- + setCustomLabels()
- + usesCustomLabels()
- + formatLabel()
- + render()

# **Public Member Functions**

- LabelEntity (const sf::Font &font, AxisType type, sf::Vector2f initialPosition)
   Constructs a LabelEntity with a given font, axis orientation and initial position.
- virtual ~LabelEntity ()=default

Virtual destructor.

float getValue () const

Returns the numeric value associated with the label.

· unsigned int getCharacterSize () const

Returns the current character size of the label text.

• int getDecimalPlaces () const

Returns the number of decimal places currently used for numeric formatting.

void setFont (const sf::Font &font)

Sets a new font for the label.

void setLabelText (std::string text)

Defines the text content of the label.

· void setCharacterSize (unsigned int newSize)

Sets a new character size for the labels.

void setDecimalPlaces (int places)

Sets the number of decimal places for numeric labels.

void setCustomLabels (const std::string &labels)

Sets a custom label string.

bool usesCustomLabels () const

Indicates whether the entity is currently using custom labels.

std::string formatLabel (float value)

Formats a numeric value into a label string.

void render (sf::RenderWindow &window)

Renders the label on the given SFML window.

# Public Member Functions inherited from wEngine::Entity

- Entity ()
- virtual ~Entity ()
- unsigned int getEntityID () const

Returns the unique ID associated with this entity.

void clearComponents ()

Removes all components currently attached to the entity.

 $\bullet \;\; template {<} typename \; T, \; typename... \; Args {>} \\$ 

```
std::shared_ptr< T > addComponent (Args &&... args)
```

Adds a new component of type T to the entity.

• template<typename T>

void removeComponent ()

Removes the component of type T from the entity.

• template<typename T>

bool hasComponent () const noexcept

Checks whether the entity has a component of type T.

• template<typename T>

```
std::shared\_ptr < T > getComponent () const
```

Retrieves the component of type T attached to the entity.

• template<typename T>

std::shared\_ptr< T > requireComponent (const std::string &context="") const

Retrieves the component of type T and throws if it's missing.

• template<typename Interface>

```
std::shared_ptr< Interface > getInterfaceComponent () const
```

Returns the first component that implements the specified interface.

### **Additional Inherited Members**

# Static Public Member Functions inherited from wEngine::Entity

static void resetEntityIDCounter ()
 Resets the global entity ID counter to zero.

# 7.13.1 Detailed Description

Represents a textual label or a collection of axis labels.

A LabelEntity manages the rendering of formatted text associated with axis notches. Labels can be generated dynamically (from numeric values, with controlled precision) or defined manually via custom strings.

The class relies on SFML's sf::Text for rendering and provides customization of style (font, color, character size) and placement (axis orientation, offset relative to the axis).

Note

Typically, a LabelEntity is aggregated inside an AxisEntity to display labels alongside axis notches.

### See also

**AxisEntity** 

### Author

Wilfried Koch

### Copyright

© 2025 Wilfried Koch. All rights reserved.

# 7.13.2 Constructor & Destructor Documentation

# 7.13.2.1 LabelEntity()

Constructs a LabelEntity with a given font, axis orientation and initial position.

### **Parameters**

font	Reference to an SFML font (must remain valid during the lifetime of the entity).
type	Axis orientation (AxisType::X_AXIS or AxisType::Y_AXIS).
initialPosition	Position where the label will be anchored.

# 7.13.2.2 ~LabelEntity()

```
virtual wPlot2D::LabelEntity::~LabelEntity () [virtual], [default]
```

Virtual destructor.

# 7.13.3 Member Function Documentation

# 7.13.3.1 getValue()

```
float wPlot2D::LabelEntity::getValue () const [nodiscard]
```

Returns the numeric value associated with the label.

Returns

The value stored in the label.

# 7.13.3.2 getCharacterSize()

```
unsigned int wPlot2D::LabelEntity::getCharacterSize () const [nodiscard]
```

Returns the current character size of the label text.

Returns

Character size in pixels.

# 7.13.3.3 getDecimalPlaces()

```
int wPlot2D::LabelEntity::getDecimalPlaces () const [nodiscard]
```

Returns the number of decimal places currently used for numeric formatting.

Returns

Number of digits after the decimal point.

# 7.13.3.4 setFont()

Sets a new font for the label.

#### **Parameters**

font Reference to an SFML font (must remain valid during the lifetime of the entity).

# 7.13.3.5 setLabelText()

Defines the text content of the label.

If custom labels are enabled, this method updates the string that will be rendered. Otherwise, it is generally managed internally via numeric formatting.

#### **Parameters**

text The new string to assign to the label.

### 7.13.3.6 setCharacterSize()

Sets a new character size for the labels.

# Parameters

newSize Character size in pixels.

# 7.13.3.7 setDecimalPlaces()

Sets the number of decimal places for numeric labels.

#### **Parameters**

places Digits after the decimal point (must be  $\geq = 0$ ).

### 7.13.3.8 setCustomLabels()

Sets a custom label string.

This enables "custom label mode". When active, numeric formatting is ignored and the provided string is displayed instead.

#### **Parameters**

labels	Custom string to display as a label.
--------	--------------------------------------

### 7.13.3.9 usesCustomLabels()

```
bool wPlot2D::LabelEntity::usesCustomLabels () const [nodiscard]
```

Indicates whether the entity is currently using custom labels.

### **Returns**

True if custom labels are active, false if numeric formatting is used.

# 7.13.3.10 formatLabel()

Formats a numeric value into a label string.

If custom labels are active, the stored custom string is returned. Otherwise, the numeric value is converted using the current number of decimal places.

#### **Parameters**

value Numeric value	to format.
---------------------	------------

### Returns

A string ready to be displayed as a label.

# 7.13.3.11 render()

Renders the label on the given SFML window.

### **Parameters**

window	Reference to the render window.

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

- wLabelEntity.hpp
- wLabelEntity.cpp

# 7.14 wPlot2D::LegendEntity Class Reference

Represents a legend box that describes functions and data plots.

#include <wLegendEntity.hpp>

Inheritance diagram for wPlot2D::LegendEntity:

# wEngine::Entity

- + Entity()
- + ~Entity()
- + getEntityID()
- + clearComponents()
- + addComponent()
- + removeComponent()
- + hasComponent()
- + getComponent()
- + requireComponent()
- + getInterfaceComponent()
- + resetEntityIDCounter()



# wPlot2D::LegendEntity

- + LegendEntity()
- + ~LegendEntity()
- + addItem()
- + addItem()
- + addItem()
- + addItem()
- + setFrameEnabled()
- + setFrameFillColor()
- + setFrameOutlineColor()
- + setFrameThickness()
- + setPadding()
- + setFont()
- + setCharacterSize()
- + setTextColor()
- + render()

#### **Public Member Functions**

LegendEntity (const sf::Font &font, const sf::Vector2f &position, bool hasFrame=true)

Constructs a LegendEntity.

virtual ~LegendEntity ()=default

Virtual destructor.

• void addItem (const std::string &label, FunctionEntity \*function)

Adds a new legend item associated with a function.

void addItem (const std::wstring &label, FunctionEntity \*function)

Adds a new legend item associated with a function (wide string).

void addltem (const std::string &label, DataPlotEntity \*plot)

Adds a new legend item associated with a data plot.

void addItem (const std::wstring &label, DataPlotEntity \*plot)

Adds a new legend item associated with a data plot (wide string).

void setFrameEnabled (bool enabled)

Enables or disables the surrounding frame of the legend.

void setFrameFillColor (const sf::Color &color)

Sets the fill color of the legend frame.

void setFrameOutlineColor (const sf::Color &color)

Sets the outline color of the legend frame.

void setFrameThickness (float thickness)

Sets the outline thickness of the legend frame.

void setPadding (const sf::Vector2f &padding)

Sets the internal padding between items and the frame borders.

void setFont (const sf::Font &font)

Updates the font used for all legend labels.

void setCharacterSize (unsigned int size)

Sets the character size of the legend text.

void setTextColor (const sf::Color &color)

Sets the color of the legend labels.

void render (sf::RenderWindow &window)

Renders the legend (all items and optional frame) to the target window.

# Public Member Functions inherited from wEngine::Entity

- Entity ()
- virtual ~Entity ()
- unsigned int getEntityID () const

Returns the unique ID associated with this entity.

• void clearComponents ()

Removes all components currently attached to the entity.

• template<typename T, typename... Args>

std::shared\_ptr< T > addComponent (Args &&... args)

Adds a new component of type T to the entity.

• template<typename T>

void removeComponent ()

Removes the component of type T from the entity.

• template<typename T>

bool hasComponent () const noexcept

Checks whether the entity has a component of type T.

```
    template<typename T>
        std::shared_ptr< T > getComponent () const
        Retrieves the component of type T attached to the entity.
    template<typename T>
        std::shared_ptr< T > requireComponent (const std::string &context="") const
        Retrieves the component of type T and throws if it's missing.
```

template<typename Interface>

std::shared\_ptr< Interface > getInterfaceComponent () const

Returns the first component that implements the specified interface.

### **Additional Inherited Members**

# Static Public Member Functions inherited from wEngine::Entity

static void resetEntityIDCounter ()
 Resets the global entity ID counter to zero.

# 7.14.1 Detailed Description

Represents a legend box that describes functions and data plots.

A LegendEntity provides a visual legend for plotted entities such as FunctionEntity and DataPlotEntity. Each legend item is composed of:

- A sample line (LineEntity) with the same style, thickness, and color as the source entity.
- A text label (sf::Text) describing the entity.

The legend can optionally be surrounded by a frame (FrameEntity) with configurable padding, outline color, thickness, and fill color.

# 7.14.1.0.1 Components and Features:

- Configurable font and text size.
- · Support for UTF-8 and wide string labels.
- · Dynamic addition of items from existing plotted entities.
- Automatic alignment of line + text pairs inside the legend box.

See also

FunctionEntity, DataPlotEntity, LineEntity, FrameEntity

Note

The font passed in the constructor must remain valid during the lifetime of the legend, as SFML does not copy font data internally.

Author

Wilfried Koch

Copyright

© 2025 Wilfried Koch

# 7.14.2 Constructor & Destructor Documentation

# 7.14.2.1 LegendEntity()

Constructs a LegendEntity.

Initializes the legend with a given font, anchor position, and optional frame.

### **Parameters**

font	Reference to the font used for labels (must remain valid).
position	Anchor position of the legend box in window coordinates.
hasFrame	Whether to display a surrounding frame (default: true).

# 7.14.2.2 ~LegendEntity()

```
virtual wPlot2D::LegendEntity::~LegendEntity () [virtual], [default]
```

Virtual destructor.

### 7.14.3 Member Function Documentation

# 7.14.3.1 addltem() [1/4]

Adds a new legend item associated with a function.

# Parameters

label	Label text (UTF-8 string).
function	Pointer to the source FunctionEntity.

# 7.14.3.2 addltem() [2/4]

Adds a new legend item associated with a function (wide string).

### **Parameters**

label	Label text (wide string).
function	Pointer to the source FunctionEntity.

# 7.14.3.3 addltem() [3/4]

Adds a new legend item associated with a data plot.

#### **Parameters**

label	Label text (UTF-8 string).
plot	Pointer to the source DataPlotEntity.

# 7.14.3.4 addltem() [4/4]

Adds a new legend item associated with a data plot (wide string).

### **Parameters**

label	Label text (wide string).
plot	Pointer to the source DataPlotEntity.

# 7.14.3.5 setFrameEnabled()

Enables or disables the surrounding frame of the legend.

### **Parameters**

ena	bled	True to display the frame, false to hide it.

# 7.14.3.6 setFrameFillColor()

Sets the fill color of the legend frame.

#### **Parameters**

color | SFML color applied to the frame's background.

### 7.14.3.7 setFrameOutlineColor()

Sets the outline color of the legend frame.

#### **Parameters**

color SFML color applied to the frame's border.

### 7.14.3.8 setFrameThickness()

```
\begin{tabular}{ll} \beg
```

Sets the outline thickness of the legend frame.

#### **Parameters**

thickness | Thickness in pixels (positive for outside expansion).

# 7.14.3.9 setPadding()

Sets the internal padding between items and the frame borders.

Padding defines horizontal and vertical margins in pixels.

#### **Parameters**

```
    padding Vector (x, y) where:
    x = horizontal padding (left/right),
    y = vertical padding (top/bottom).
```

### 7.14.3.10 setFont()

Updates the font used for all legend labels.

# Note

The font must remain valid during the legend's lifetime, as SFML does not copy font data internally.

#### **Parameters**

font Reference to an externally managed sf::Font.

# 7.14.3.11 setCharacterSize()

Sets the character size of the legend text.

#### **Parameters**

size Font size in pixels.

### 7.14.3.12 setTextColor()

Sets the color of the legend labels.

### **Parameters**

color | SFML color applied to all legend text.

# 7.14.3.13 render()

Renders the legend (all items and optional frame) to the target window.

Each item is drawn with its sample line and label text, aligned inside the legend box. The optional frame is drawn behind all items.

# **Parameters**

window The target SFML render window.

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

- · wLegendEntity.hpp
- wLegendEntity.cpp

# 7.15 wEngine::LengthComponent Class Reference

ECS component that defines the length of a drawable object.

#include <wLengthComponent.hpp>

Inheritance diagram for wEngine::LengthComponent:

# wEngine::Component + ~Component() + enable() + disable() + isEnabled() + setParent() + getParent() # Component() wEngine::LengthComponent + LengthComponent() + ~LengthComponent() + getLength() + setLength() + debugPrint()

# **Public Member Functions**

• LengthComponent (float length=2.0f)

Constructs the component with an initial positive length.

- virtual ~LengthComponent ()=default
- float getLength () const

Returns the current length.

void setLength (float newLength)

Sets a new length value.

• void debugPrint () const

Outputs the current length value to the console for debugging.

# Public Member Functions inherited from wEngine::Component

- virtual ∼Component ()=default
- virtual void enable ()
- virtual void disable ()
- bool isEnabled () const

Checks whether the component is currently active.

void setParent (Entity \*parent)

Sets the parent entity of this component.

• Entity \* getParent () const

Returns the parent entity of this component.

### **Additional Inherited Members**

# Protected Member Functions inherited from wEngine::Component

· Component ()

Protected constructor to restrict instantiation to derived classes.

# 7.15.1 Detailed Description

ECS component that defines the length of a drawable object.

This component stores a positive float value representing the length (in pixels) of lines or shapes. The value must be strictly positive.

**Exceptions** 

std::invalid argument	if the value is zero or negative	.
Siuirivaiiu_argumeni	I lille value is zero or nega	uve.

**Author** 

Wilfried Koch

Copyright

© 2025 Wilfried Koch. All rights reserved.

### 7.15.2 Constructor & Destructor Documentation

# 7.15.2.1 LengthComponent()

Constructs the component with an initial positive length.

### **Parameters**

length   Initial length value (default	t is 2.0f).
--	-------------

### **Exceptions**

std::invalid_argument	if the value is zero or negative.
-----------------------	-----------------------------------

# 7.15.2.2 ~LengthComponent()

```
virtual wEngine::LengthComponent::~LengthComponent () [virtual], [default]
```

### 7.15.3 Member Function Documentation

### 7.15.3.1 getLength()

```
float wEngine::LengthComponent::getLength () const [nodiscard]
```

Returns the current length.

### Returns

A positive float representing the line length (in pixels).

# 7.15.3.2 setLength()

Sets a new length value.

### **Parameters**

newLength	A strictly positive float.
-----------	----------------------------

# **Exceptions**

	std::invalid argument	if the value is zero or negative.
--	-----------------------	-----------------------------------

### 7.15.3.3 debugPrint()

```
void wEngine::LengthComponent::debugPrint () const
```

Outputs the current length value to the console for debugging.

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

- wLengthComponent.hpp
- wLengthComponent.cpp

# 7.16 wEngine::LineDrawer Class Reference

Utility class for rendering thick lines and polylines with style support.

#include <wLineDrawer.hpp>

### **Static Public Member Functions**

static float drawLine (sf::RenderWindow &window, const sf::Vector2f &point1, const sf::Vector2f &point2, const sf::Color &color, float thickness, LineStyleComponent::LineStyle style=LineStyleComponent::LineStyle::Solid, float dashLength=20.0f, float gapLength=5.0f, float patternOffset=0.0f)

Draws a single thick line segment between two points.

static void drawPolylineRound (sf::RenderWindow &window, const std::vector< sf::Vector2f > &points, const sf::Color &color, float thickness, LineStyleComponent::LineStyle style=LineStyleComponent::LineStyle::Solid, float dashLength=20.0f, float gapLength=5.0f, unsigned int arcResolution=12)

Draws a polyline (sequence of connected line segments) with optional round joins.

# 7.16.1 Detailed Description

Utility class for rendering thick lines and polylines with style support.

The LineDrawer provides static methods to draw line segments and polylines with configurable thickness, color, and style (Solid, Dashed, Dotted).

### 7.16.1.0.1 Features:

- · Thick line rendering via quads (two triangles per segment).
- Support for dashed and dotted patterns using configurable dash/gap lengths.
- · Dash/dot continuity across multiple connected segments using a shared pattern offset.
- Optional round joins at corners of polylines (applied only when style == Solid).

### 7.16.1.0.2 Usage:

- Use drawLine() to render a single thick segment.
- Use drawPolylineRound() to render a sequence of connected points with optional round joins.
- To maintain consistent dash/dot alignment across segments, pass the returned patternOffset from drawLine() into the next segment.

### Warning

Round joins are currently only applied for Solid style. For dashed or dotted lines, joins would produce inconsistent results and are therefore omitted.

### See also

LineStyleComponent for configuring line style options.

### **Author**

Wilfried Koch

### Copyright

© 2025 Wilfried Koch. All rights reserved.

### 7.16.2 Member Function Documentation

### 7.16.2.1 drawLine()

```
float wEngine::LineDrawer::drawLine (
    sf::RenderWindow & window,
    const sf::Vector2f & point1,
    const sf::Vector2f & point2,
    const sf::Color & color,
    float thickness,
    LineStyleComponent::LineStyle style = LineStyleComponent::LineStyle::Solid,
    float dashLength = 20.0f,
    float gapLength = 5.0f,
    float patternOffset = 0.0f) [static]
```

Draws a single thick line segment between two points.

### 7.16.2.1.1 Style behavior:

- Solid: Renders a single quad covering the full segment.
- Dashed: Repeats a dash/gap pattern along the segment.
- Dotted: Places successive dots along the segment, using thickness as dot length.

# 7.16.2.1.2 Pattern control:

- dashLength sets the visible length of each dash (Dashed style).
- gapLength sets the empty space between dashes or dots.
- thickness is reused as the dot length if style == Dotted.
- patternOffset maintains pattern alignment between consecutive calls.

### **Parameters**

window	Render target.
point1	First endpoint of the line.
point2	Second endpoint of the line.
color	Line color.
thickness	Line thickness in pixels.
style	Line style (Solid, Dashed, Dotted).
dashLength	Dash length (used if style == Dashed).
gapLength	Gap length between dashes or dots.
patternOffset	Initial offset within the dash/dot pattern.

### Returns

Updated pattern offset after this segment (pass to next segment for continuity).

### 7.16.2.2 drawPolylineRound()

```
void wEngine::LineDrawer::drawPolylineRound (
    sf::RenderWindow & window,
    const std::vector< sf::Vector2f > & points,
    const sf::Color & color,
    float thickness,
    LineStyleComponent::LineStyle style = LineStyleComponent::LineStyle::Solid,
    float dashLength = 20.0f,
    float gapLength = 5.0f,
    unsigned int arcResolution = 12) [static]
```

Draws a polyline (sequence of connected line segments) with optional round joins.

- Each segment [p1, p2] is rendered using drawLine(), with pattern continuity preserved.
- If style == Solid and a next segment exists: a circular arc is approximated using triangles to smooth the corner at [p2].

### **Parameters**

window	Render target.
points	List of polyline points (must contain at least 2).
color	Polyline color.
thickness	Line thickness in pixels.
style	Line style (Solid, Dashed, Dotted).
dashLength	Dash length (used if style == Dashed).
gapLength	Gap length between dashes or dots.
arcResolution	Number of triangles used to approximate each round join (higher = smoother).

### Note

For Dashed or Dotted styles, round joins are skipped.

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

- · wLineDrawer.hpp
- · wLineDrawer.cpp

# 7.17 wPlot2D::LineEntity Class Reference

Entity representing a straight line segment with optional arrowhead.

```
#include <wLineEntity.hpp>
```

Inheritance diagram for wPlot2D::LineEntity:

# wEngine::Entity

- + Entity()
- + ~Entity()
- + getEntityID()
- + clearComponents()
- + addComponent()
- + removeComponent()
- + hasComponent()
- + getComponent()
- + requireComponent()
- + getInterfaceComponent()
- + resetEntityIDCounter()



# wPlot2D::LineEntity

- + LineEntity()
- + ~LineEntity()
- + setColor()
- + setThickness()
- + getThickness()
- + setLineStyle()
- + setDashLength()
- + setGapLength()
- + getStartPoint()
- + getEndPoint()
- + hasArrow()
- + getArrowSize()
- + setArrowSize()
- + render()

# **Public Member Functions**

LineEntity (const sf::Vector2f &origin, const sf::Vector2f &scale, const sf::Vector2f &start, const sf::Vector2f &end, bool withArrow=false)

Construct a line entity between two points.

virtual ∼LineEntity ()=default

Virtual destructor.

· void setColor (sf::Color color)

Sets the color of the line and arrowhead.

void setThickness (float thickness)

Sets the thickness of the line.

float getThickness () const

Returns the current thickness of the line.

void setLineStyle (wEngine::LineStyleComponent::LineStyle style)

Sets the visual style of the line.

void setDashLength (float dashLength)

Sets the dash length for dashed lines.

void setGapLength (float gapLength)

Sets the gap length between dashes or dots.

sf::Vector2f getStartPoint () const

Returns the starting point of the line.

sf::Vector2f getEndPoint () const

Returns the ending point of the line.

• bool hasArrow () const

Checks if the line has an arrowhead.

• float getArrowSize () const

Returns the arrowhead size factor.

• void setArrowSize (float arrowSize)

Sets the arrowhead size factor.

· void render (sf::RenderWindow &window)

Renders the line (and optional arrowhead).

### Public Member Functions inherited from wEngine::Entity

- Entity ()
- virtual ~Entity ()
- unsigned int getEntityID () const

Returns the unique ID associated with this entity.

void clearComponents ()

Removes all components currently attached to the entity.

• template<typename T, typename... Args>

```
std::shared_ptr< T > addComponent (Args &&... args)
```

Adds a new component of type T to the entity.

template<typename T>

void removeComponent ()

Removes the component of type T from the entity.

template<typename T>

bool hasComponent () const noexcept

Checks whether the entity has a component of type T.

template<typename T>

```
std::shared_ptr< T > getComponent () const
```

Retrieves the component of type T attached to the entity.

template<typename T>

std::shared\_ptr< T > requireComponent (const std::string &context="") const

Retrieves the component of type T and throws if it's missing.

template<typename Interface>

std::shared\_ptr< Interface > getInterfaceComponent () const

Returns the first component that implements the specified interface.

### **Additional Inherited Members**

# Static Public Member Functions inherited from wEngine::Entity

static void resetEntityIDCounter ()
 Resets the global entity ID counter to zero.

# 7.17.1 Detailed Description

Entity representing a straight line segment with optional arrowhead.

This entity provides configurable line rendering within the plot area:

- Supports Solid, Dashed, and Dotted styles (via LineStyleComponent).
- · Customizable color, thickness, dash length, and gap length.
- · Optional arrowhead at the end (useful for axes or vectors).

Coordinates are expressed in logical units and transformed by the entity's origin and scale before being rendered.

### See also

```
wEngine::LineDrawer for the rendering implementation wEngine::LineStyleComponent for style configuration
```

### **Author**

Wilfried Koch

### Copyright

© 2025 Wilfried Koch. All rights reserved.

### 7.17.2 Constructor & Destructor Documentation

# 7.17.2.1 LineEntity()

Construct a line entity between two points.

### **Parameters**

origin	Origin of the plot (reference point).
scale	Scaling factor to convert logical coordinates into pixels.
start	Line starting point (logical coordinates).
end	Line ending point (logical coordinates).
withArrow	Whether to draw an arrowhead at the end.

# 7.17.2.2 ∼LineEntity()

```
virtual wPlot2D::LineEntity::~LineEntity () [virtual], [default]
```

Virtual destructor.

# 7.17.3 Member Function Documentation

# 7.17.3.1 setColor()

Sets the color of the line and arrowhead.

### **Parameters**

color	New color to apply.
-------	---------------------

### 7.17.3.2 setThickness()

Sets the thickness of the line.

### **Parameters**

thickness	Thickness in pixels.
-----------	----------------------

# 7.17.3.3 getThickness()

```
float wPlot2D::LineEntity::getThickness () const [nodiscard]
```

Returns the current thickness of the line.

Returns

Thickness in pixels.

### 7.17.3.4 setLineStyle()

Sets the visual style of the line.

### **Parameters**

```
style Solid, Dashed, or Dotted.
```

### 7.17.3.5 setDashLength()

```
\begin{tabular}{ll} \begin{tabular}{ll} void & wPlot2D::LineEntity::setDashLength ( \\ & float & dashLength) \end{tabular}
```

Sets the dash length for dashed lines.

### **Parameters**

dashLength Length of each dash	n in pixels.
--------------------------------	--------------

### 7.17.3.6 setGapLength()

Sets the gap length between dashes or dots.

### **Parameters**

ı	ganl ength	Length of the gap in pixels.
	gaptongin	Longin of the gap in pixels.

# 7.17.3.7 getStartPoint()

```
sf::Vector2f wPlot2D::LineEntity::getStartPoint () const [nodiscard]
```

Returns the starting point of the line.

# Returns

Start point in logical coordinates.

# 7.17.3.8 getEndPoint()

```
sf::Vector2f wPlot2D::LineEntity::getEndPoint () const [nodiscard]
```

Returns the ending point of the line.

### Returns

End point in logical coordinates.

### 7.17.3.9 hasArrow()

```
bool wPlot2D::LineEntity::hasArrow () const [nodiscard]
```

Checks if the line has an arrowhead.

Returns

True if an arrowhead is drawn, false otherwise.

# 7.17.3.10 getArrowSize()

```
float wPlot2D::LineEntity::getArrowSize () const [nodiscard]
```

Returns the arrowhead size factor.

Returns

Arrow size relative to line thickness.

# 7.17.3.11 setArrowSize()

Sets the arrowhead size factor.

**Parameters** 

```
arrowSize Arrow size relative to line thickness.
```

## 7.17.3.12 render()

Renders the line (and optional arrowhead).

**Parameters** 

window Target render window	١.
-----------------------------	----

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

- wLineEntity.hpp
- wLineEntity.cpp

# 7.18 wEngine::LineStyleComponent Class Reference

ESC component that defines the style of a line (solid, dotted, dashed).

#include <wLineStyleComponent.hpp>

Inheritance diagram for wEngine::LineStyleComponent:

### wEngine::Component

- + ~Component()
- + enable()
- + disable()
- + isEnabled()
- + setParent()
- + getParent()
- # Component()

# wEngine::LineStyleComponent

- + LineStyleComponent()
- + ~LineStyleComponent()
- + getStyle()
- + setStyle()
- + getDashLength()
- + setDashLength()
- + getGapLength()
- + setGapLength()
- + debugPrint()

# **Public Types**

enum class LineStyle { Solid , Dotted , Dashed }
 Available styles for line rendering.

### **Public Member Functions**

• LineStyleComponent (LineStyle style=LineStyle::Solid)

Constructs a LineStyleComponent with an optional style.

- virtual ~LineStyleComponent ()=default
- · LineStyle getStyle () const

Returns the current line style.

• void setStyle (LineStyle style)

Sets the current line style.

float getDashLength () const

Returns the dash length (used for Dashed style).

void setDashLength (float dashLength)

Sets the dash length.

• float getGapLength () const

Returns the gap length (used for Dotted and Dashed styles).

void setGapLength (float gapLength)

Sets the gap length.

· void debugPrint () const

# Public Member Functions inherited from wEngine::Component

- virtual ∼Component ()=default
- virtual void enable ()
- virtual void disable ()
- bool isEnabled () const

Checks whether the component is currently active.

void setParent (Entity \*parent)

Sets the parent entity of this component.

• Entity \* getParent () const

Returns the parent entity of this component.

### **Additional Inherited Members**

# Protected Member Functions inherited from wEngine::Component

• Component ()

Protected constructor to restrict instantiation to derived classes.

# 7.18.1 Detailed Description

ESC component that defines the style of a line (solid, dotted, dashed).

This component controls how lines are drawn in the rendering pipeline. For dotted and dashed styles, both dash length and gap length can be configured.

**Author** 

Wilfried Koch

Copyright

© 2025 Wilfried Koch. All rights reserved.

### 7.18.2 Member Enumeration Documentation

### 7.18.2.1 LineStyle

enum class wEngine::LineStyleComponent::LineStyle [strong]

Available styles for line rendering.

### Enumerator

Solid	
Dotted	
Dashed	

### 7.18.3 Constructor & Destructor Documentation

# 7.18.3.1 LineStyleComponent()

Constructs a LineStyleComponent with an optional style.

### **Parameters**

```
style Line style to use (default: Solid).
```

### 7.18.3.2 ∼LineStyleComponent()

```
virtual wEngine::LineStyleComponent::~LineStyleComponent () [virtual], [default]
```

### 7.18.4 Member Function Documentation

# 7.18.4.1 getStyle()

```
{\tt LineStyleComponent::LineStyle \ wEngine::LineStyleComponent::getStyle \ () \ const \ [nodiscard]}
```

Returns the current line style.

### Returns

The style as LineStyle.

### 7.18.4.2 setStyle()

Sets the current line style.

### **Parameters**

style New line style to apply.

### 7.18.4.3 getDashLength()

```
float wEngine::LineStyleComponent::getDashLength () const [nodiscard]
```

Returns the dash length (used for Dashed style).

Returns

Dash length in pixels.

## 7.18.4.4 setDashLength()

Sets the dash length.

**Parameters** 

### **Exceptions**

```
std::invalid_argument | if dashLength <= 0.
```

### 7.18.4.5 getGapLength()

```
float wEngine::LineStyleComponent::getGapLength () const [nodiscard]
```

Returns the gap length (used for Dotted and Dashed styles).

Returns

Gap length in pixels.

# 7.18.4.6 setGapLength()

Sets the gap length.

**Parameters** 

gapLength	Gap length in pixels.
946-09	0.ap .0g pc.c.

### **Exceptions**

```
std::invalid_argument | if gapLength < 0.
```

### 7.18.4.7 debugPrint()

```
void wEngine::LineStyleComponent::debugPrint () const
```

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

- wLineStyleComponent.hpp
- wLineStyleComponent.cpp

# 7.19 wEngine::MathUtils Class Reference

Provides common mathematical helper functions for plotting and geometry.

```
#include <wMathUtils.hpp>
```

### **Static Public Member Functions**

• static std::vector< double > linspace (double start, double end, size\_t nbPoints)

Generates a linearly spaced vector of values between two bounds.

# 7.19.1 Detailed Description

Provides common mathematical helper functions for plotting and geometry.

This utility class groups static methods that are frequently needed when working with numerical data, discretization, and rendering curves.

Note

All methods are static and do not require instantiation.

Author

Wilfried Koch

Copyright

© 2025 Wilfried Koch. All rights reserved.

### 7.19.2 Member Function Documentation

### 7.19.2.1 linspace()

Generates a linearly spaced vector of values between two bounds.

This function produces a vector of evenly spaced points between start and end (inclusive).

### **Parameters**

start	Starting value.
end	Ending value.
nbPoints	Number of points to generate (must be $\geq$ = 2).

### Returns

A std::vector< double > containing evenly spaced values.

# **Exceptions**

```
std::invalid_argument | if start >= end or nbPoints < 2.
```

```
auto values = MathUtils::linspace( 0.0, 1.0, 5 );
// values = { 0.0, 0.25, 0.5, 0.75, 1.0 }
```

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

- wMathUtils.hpp
- · wMathUtils.cpp

# 7.20 wPlot2D::NotchEntity Class Reference

Represents a single tick mark ("notch") on a 2D axis.

```
#include <wNotchEntity.hpp>
```

Inheritance diagram for wPlot2D::NotchEntity:

# wEngine::Entity + Entity() + ~Entity() + getEntityID() + clearComponents() + addComponent() + removeComponent() + hasComponent() + getComponent() + requireComponent() + getInterfaceComponent() + resetEntityIDCounter() wPlot2D::NotchEntity + NotchEntity() ~NotchEntity() + render()

# **Public Member Functions**

NotchEntity (AxisType type)

Constructs a NotchEntity aligned to a given axis.

• virtual  $\sim$ NotchEntity ()=default

Virtual destructor.

• void render (sf::RenderWindow &window)

Renders the notch using SFML.

# Public Member Functions inherited from wEngine::Entity

- Entity ()
- virtual ∼Entity ()
- unsigned int getEntityID () const

Returns the unique ID associated with this entity.

void clearComponents ()

Removes all components currently attached to the entity.

```
• template<typename T, typename... Args>
  std::shared_ptr< T > addComponent (Args &&... args)
     Adds a new component of type T to the entity.

    template<typename T>

  void removeComponent ()
     Removes the component of type T from the entity.

    template<typename T>

 bool hasComponent () const noexcept
      Checks whether the entity has a component of type T.
• template<typename T>
  std::shared_ptr< T > getComponent () const
     Retrieves the component of type T attached to the entity.
• template<typename T>
  std::shared_ptr< T > requireComponent (const std::string &context="") const
     Retrieves the component of type T and throws if it's missing.
• template<typename Interface>
  std::shared_ptr< Interface > getInterfaceComponent () const
     Returns the first component that implements the specified interface.
```

### **Additional Inherited Members**

# Static Public Member Functions inherited from wEngine::Entity

static void resetEntityIDCounter ()
 Resets the global entity ID counter to zero.

# 7.20.1 Detailed Description

Represents a single tick mark ("notch") on a 2D axis.

A NotchEntity is a visual element used to mark intervals along a coordinate axis, helping users interpret scale in a plot. It is rendered as a small filled rectangle, oriented perpendicularly to its associated axis ( $X_AXIS$ ).

### 7.20.1.0.1 Components required:

- PositionComponent: specifies the top-left pixel position.
- ColorComponent: defines the notch color.
- ThicknessComponent: defines the thickness (along the axis).
- $\bullet$  LengthComponent: defines the length (perpendicular to the axis).

### 7.20.1.0.2 Orientation:

```
• AxisType::X_AXIS: vertical notch (aligned with Y),
```

• AxisType::Y\_AXIS: horizontal notch (aligned with X).

The actual rendering logic is handled internally using sf::RectangleShape.

Note

Components must be added externally (typically by AxisEntity::addNotches()).

See also

AxisEntity, AxisType, PositionComponent, LengthComponent, ThicknessComponent

**Author** 

Wilfried Koch

Copyright

© 2025 Wilfried Koch. All rights reserved.

### 7.20.2 Constructor & Destructor Documentation

### 7.20.2.1 NotchEntity()

Constructs a NotchEntity aligned to a given axis.

**Parameters** 

```
type The axis type (X or Y) which determines notch orientation.
```

### 7.20.2.2 ~NotchEntity()

```
virtual wPlot2D::NotchEntity::~NotchEntity () [virtual], [default]
```

Virtual destructor.

### 7.20.3 Member Function Documentation

### 7.20.3.1 render()

Renders the notch using SFML.

Builds a rectangle from ECS components (position, thickness, length, color) and draws it in the render window.

### **Exceptions**

std::runtime_error	if any required component is missing.
--------------------	---------------------------------------

### **Parameters**

window	The render window to draw onto.

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

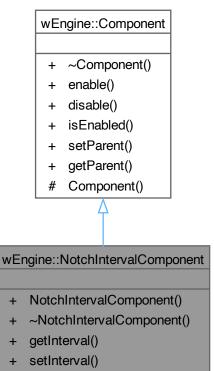
- wNotchEntity.hpp
- wNotchEntity.cpp

# 7.21 wEngine::NotchIntervalComponent Class Reference

ECS component that defines the interval between notches on an axis.

#include <wNotchIntervalComponent.hpp>

Inheritance diagram for wEngine::NotchIntervalComponent:



debugPrint()

### **Public Member Functions**

NotchIntervalComponent (float interval=1.0f)

Constructs a NotchIntervalComponent with a given interval.

- virtual ~NotchIntervalComponent ()=default
- float getInterval () const

Returns the current spacing between notches.

void setInterval (float newInterval)

Sets a new interval between notches.

void debugPrint () const

Outputs the current interval to standard output for debugging.

# Public Member Functions inherited from wEngine::Component

- virtual ∼Component ()=default
- virtual void enable ()
- virtual void disable ()
- bool isEnabled () const

Checks whether the component is currently active.

void setParent (Entity \*parent)

Sets the parent entity of this component.

Entity \* getParent () const

Returns the parent entity of this component.

### **Additional Inherited Members**

# Protected Member Functions inherited from wEngine::Component

· Component ()

Protected constructor to restrict instantiation to derived classes.

# 7.21.1 Detailed Description

ECS component that defines the interval between notches on an axis.

This component stores the spacing (in logical units) between visual ticks or markers on an axis, such as those used in coordinate grids or charts.

It enables the placement of regular visual markers and may be queried by rendering systems.

**Author** 

Wilfried Koch

Copyright

© 2025 Wilfried Koch. All rights reserved.

# 7.21.2 Constructor & Destructor Documentation

### 7.21.2.1 NotchIntervalComponent()

Constructs a NotchIntervalComponent with a given interval.

### **Parameters**

$\mid$ interval $\mid$ Distance between notches in logical units (must be $>$ 0). Default is 1.0
--

### **Exceptions**

std::invalid_argument	if interval is not strictly positive.
-----------------------	---------------------------------------

### 7.21.2.2 ~NotchIntervalComponent()

```
virtual wEngine::NotchIntervalComponent::~NotchIntervalComponent () [virtual], [default]
```

### 7.21.3 Member Function Documentation

### 7.21.3.1 getInterval()

```
float wEngine::NotchIntervalComponent::getInterval () const [nodiscard]
```

Returns the current spacing between notches.

### Returns

The interval, expressed in logical units.

# 7.21.3.2 setInterval()

Sets a new interval between notches.

### **Parameters**

newInterval	New spacing value (must be $> 0$ ).

### **Exceptions**

std::invalid_argument	if newInterval is not strictly positive.
-----------------------	--

### 7.21.3.3 debugPrint()

```
void wEngine::NotchIntervalComponent::debugPrint () const
```

Outputs the current interval to standard output for debugging.

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

- wNotchIntervalComponent.hpp
- wNotchIntervalComponent.cpp

# 7.22 wEngine::OffsetComponent Class Reference

ECS component that defines a logical coordinate offset.

#include <wOffsetComponent.hpp>

Inheritance diagram for wEngine::OffsetComponent:

# wEngine::Component + ~Component() + enable() + disable() + isEnabled() + setParent() + getParent() # Component() wEngine::OffsetComponent + OffsetComponent() + ~OffsetComponent() + setOffset() + addOffset() + debugPrint()

# **Public Member Functions**

- OffsetComponent (sf::Vector2f offset=sf::Vector2f(0.0f, 0.0f))
  - Constructs the component with an optional offset.
- virtual ∼OffsetComponent ()=default
- sf::Vector2f getOffset () const

Returns the current offset.

void setOffset (sf::Vector2f offset)

Updates the logical offset value.

• void addOffset (sf::Vector2f delta)

Increments the current offset by a given vector.

• void debugPrint () const

Outputs the current offset value to the console for debugging.

# Public Member Functions inherited from wEngine::Component

- virtual ∼Component ()=default
- virtual void enable ()
- virtual void disable ()
- · bool isEnabled () const

Checks whether the component is currently active.

void setParent (Entity \*parent)

Sets the parent entity of this component.

• Entity \* getParent () const

Returns the parent entity of this component.

### **Additional Inherited Members**

### Protected Member Functions inherited from wEngine::Component

· Component ()

Protected constructor to restrict instantiation to derived classes.

# 7.22.1 Detailed Description

ECS component that defines a logical coordinate offset.

This component stores a 2D offset (in logical units) applied to graphical elements such as axes, curves, titles, labels and data points. It allows changing the visual reference frame without affecting pixel-based positioning.

### 7.22.1.0.1 Usage Examples:

- With offset = (0,0), logical coordinates are drawn as-is.
- With offset = ( -20, 0 ), the visual origin is shifted 20 units to the right.

The offset is often combined with the origin and scale to compute actual pixel positions.

Author

Wilfried Koch

Copyright

© 2025 Wilfried Koch. All rights reserved.

### 7.22.2 Constructor & Destructor Documentation

# 7.22.2.1 OffsetComponent()

Constructs the component with an optional offset.

### **Parameters**

offset A 2D vector representing the logical offset (default is (0.0f, 0.0f)).

# 7.22.2.2 ~OffsetComponent()

```
virtual wEngine::OffsetComponent::~OffsetComponent () [virtual], [default]
```

# 7.22.3 Member Function Documentation

### 7.22.3.1 getOffset()

```
sf::Vector2f wEngine::OffsetComponent::getOffset () const [nodiscard]
```

Returns the current offset.

### Returns

A 2D vector representing the logical offset.

# 7.22.3.2 setOffset()

Updates the logical offset value.

### **Parameters**

offset The new offset to apply.

### 7.22.3.3 addOffset()

Increments the current offset by a given vector.

### **Parameters**

delta The additional offset to add to the current value.

### 7.22.3.4 debugPrint()

void wEngine::OffsetComponent::debugPrint () const

Outputs the current offset value to the console for debugging.

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

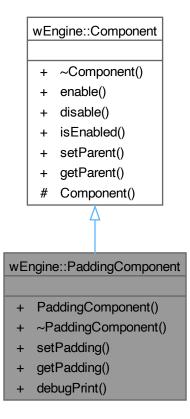
- wOffsetComponent.hpp
- wOffsetComponent.cpp

# 7.23 wEngine::PaddingComponent Class Reference

ECS component representing internal padding for UI-like elements.

#include <wPaddingComponent.hpp>

Inheritance diagram for wEngine::PaddingComponent:



### **Public Member Functions**

PaddingComponent (sf::Vector2f padding=sf::Vector2f(0.0f, 0.0f))

Constructs a PaddingComponent with optional initial padding.

- virtual ∼PaddingComponent ()=default
- void setPadding (sf::Vector2f padding)

Sets the padding vector.

sf::Vector2f getPadding () const

Returns the current padding values.

· void debugPrint () const

Prints the current padding values to standard output for debugging.

# Public Member Functions inherited from wEngine::Component

- virtual ∼Component ()=default
- virtual void enable ()
- virtual void disable ()
- bool isEnabled () const

Checks whether the component is currently active.

void setParent (Entity \*parent)

Sets the parent entity of this component.

• Entity \* getParent () const

Returns the parent entity of this component.

### **Additional Inherited Members**

### Protected Member Functions inherited from wEngine::Component

· Component ()

Protected constructor to restrict instantiation to derived classes.

### 7.23.1 Detailed Description

ECS component representing internal padding for UI-like elements.

This component encapsulates a 2D padding vector (horizontal and vertical) that can be used to add internal spacing between a visual element (e.g., a title or a frame) and its boundary.

# 7.23.1.0.1 Padding Convention:

- · x corresponds to horizontal padding (left and right),
- · y corresponds to vertical padding (top and bottom).

**Author** 

Wilfried Koch

### Copyright

© 2025 Wilfried Koch. All rights reserved.

### 7.23.2 Constructor & Destructor Documentation

### 7.23.2.1 PaddingComponent()

Constructs a PaddingComponent with optional initial padding.

### **Parameters**

```
padding Padding vector (x, y) (default is ( 0.0f, 0.0f )).
```

### 7.23.2.2 ~PaddingComponent()

```
virtual wEngine::PaddingComponent::~PaddingComponent () [virtual], [default]
```

### 7.23.3 Member Function Documentation

### 7.23.3.1 setPadding()

Sets the padding vector.

### **Parameters**

padding	New padding values (x, y).
padding	new padding values (x, y).

### 7.23.3.2 getPadding()

```
sf::Vector2f wEngine::PaddingComponent::getPadding () const [nodiscard]
```

Returns the current padding values.

### Returns

Padding vector (x, y).

### 7.23.3.3 debugPrint()

```
void wEngine::PaddingComponent::debugPrint () const
```

Prints the current padding values to standard output for debugging.

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

- wPaddingComponent.hpp
- wPaddingComponent.cpp

# 7.24 wEngine::PathUtils Class Reference

Utility class providing static functions for managing executable and resource paths across platforms.

#include <wPathUtils.hpp>

### **Static Public Member Functions**

static std::string getExecutablePath ()

Returns the absolute path to the current executable.

static std::string getExecutableDir ()

Returns the directory containing the current executable.

# 7.24.1 Detailed Description

Utility class providing static functions for managing executable and resource paths across platforms.

This class retrieves the absolute path of the current executable or its parent directory, in a portable way (macOS, Windows, Linux). It is particularly useful for locating resources such as fonts, images, or configuration files relative to the application binary.

All methods are static and do not require instantiation.

Note

If the path cannot be determined, a std::runtime\_error is thrown.

### 7.24.1.0.1 Notes on unusual headers:

- <mach-o/dyld.h> (macOS): provides \_NSGetExecutablePath, part of the Mach-O dynamic loader API. Documentation: see Apple Developer docs (man page: man 3 dyld).
- <windows.h> (Windows): provides GetModuleFileNameA, part of the Win32 API. Documentation: see Microsoft Learn (GetModuleFileName function).
- <unistd.h>+/proc/self/exe (Linux): /proc/self/exe is a symbolic link exposing the running executable, documented in man 5 proc.

These APIs are not C++ standard and must be used with care, as they are platform-dependent.

See also

std::filesystem for path manipulations.

Author

Wilfried Koch

Copyright

© 2025 Wilfried Koch. All rights reserved.

# 7.24.2 Member Function Documentation

### 7.24.2.1 getExecutablePath()

```
std::string wEngine::PathUtils::getExecutablePath () [static], [nodiscard]
```

Returns the absolute path to the current executable.

Platform-specific APIs are used:

- macOS: \_NSGetExecutablePath
- Windows: GetModuleFileNameA
- Linux: /proc/self/exe

### Returns

Absolute path to the binary (e.g., "/path/to/MyApp.app/Contents/MacOS/MyApp").

# **Exceptions**

std::runtime error	if the path cannot be resolved.

### 7.24.2.2 getExecutableDir()

```
std::string wEngine::PathUtils::getExecutableDir () [static], [nodiscard]
```

Returns the directory containing the current executable.

This is often used as a base path to resolve relative resource paths.

### Returns

Directory containing the binary (e.g., "/path/to/MyApp.app/Contents/MacOS").

### **Exceptions**

```
std::runtime_error if the path cannot be resolved.
```

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

- · wPathUtils.hpp
- · wPathUtils.cpp

# 7.25 wEngine::PositionComponent Class Reference

ECS component storing the position of an entity in 2D space and supports movement tracking.

#include <wPositionComponent.hpp>

Inheritance diagram for wEngine::PositionComponent:

# wEngine::Component + ~Component() enable() disable() isEnabled() setParent() getParent() Component() wEngine::PositionComponent + PositionComponent() ~PositionComponent() getPosition() getLastPosition() setPosition() move() debugPrint()

### **Public Member Functions**

- PositionComponent (sf::Vector2f position=sf::Vector2f(0.0f, 0.0f))
  - Constructs a PositionComponent with the given position.
- virtual ∼PositionComponent ()=default
- sf::Vector2f getPosition () const

Gets the current position of the component.

• sf::Vector2f getLastPosition () const

Returns the previous position (before the last move).

void setPosition (sf::Vector2f newPosition)

Sets the 2D position to a new value.

• void move (const sf::Vector2f &offset)

Moves the position by an offset.

• void debugPrint () const

Outputs the position (x, y) to standard output for debugging.

# Public Member Functions inherited from wEngine::Component

- virtual ∼Component ()=default
- virtual void enable ()
- · virtual void disable ()
- bool isEnabled () const

Checks whether the component is currently active.

void setParent (Entity \*parent)

Sets the parent entity of this component.

• Entity \* getParent () const

Returns the parent entity of this component.

### **Additional Inherited Members**

# Protected Member Functions inherited from wEngine::Component

· Component ()

Protected constructor to restrict instantiation to derived classes.

# 7.25.1 Detailed Description

ECS component storing the position of an entity in 2D space and supports movement tracking.

This component holds a 2D vector representing the current and previous spatial position of its parent entity.

### **7.25.1.0.1 Usage Examples:**

· Default position at origin:

```
addComponent< wEngine::PositionComponent >( );
```

· Custom position:

```
addComponent< wEngine::PositionComponent >( sf::Vector2f( 100.0f, 200.0f ) );
```

Author

Wilfried Koch

### Copyright

© 2025 Wilfried Koch. All rights reserved.

### 7.25.2 Constructor & Destructor Documentation

### 7.25.2.1 PositionComponent()

Constructs a PositionComponent with the given position.

### **Parameters**

position	Initial position (default is ( 0.0f, 0.0f )).	
p		

### 7.25.2.2 ~PositionComponent()

```
virtual wEngine::PositionComponent::~PositionComponent () [virtual], [default]
```

# 7.25.3 Member Function Documentation

### 7.25.3.1 getPosition()

```
sf::Vector2f wEngine::PositionComponent::getPosition () const [nodiscard]
```

Gets the current position of the component.

### Returns

The current 2D position vector.

### 7.25.3.2 getLastPosition()

```
sf::Vector2f wEngine::PositionComponent::getLastPosition () const [nodiscard]
```

Returns the previous position (before the last move).

### Returns

The last recorded position.

# 7.25.3.3 setPosition()

Sets the 2D position to a new value.

### **Parameters**

```
newPosition The new position.
```

### 7.25.3.4 move()

Moves the position by an offset.

### **Parameters**

offset The offset to add to the current position.

Stores the current position as the last position before applying the offset.

### 7.25.3.5 debugPrint()

void wEngine::PositionComponent::debugPrint () const

Outputs the position (x, y) to standard output for debugging.

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

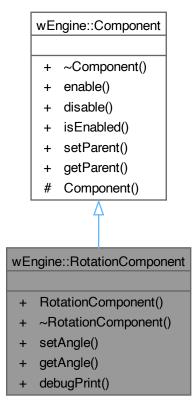
- wPositionComponent.hpp
- wPositionComponent.cpp

# 7.26 wEngine::RotationComponent Class Reference

ECS component that stores a rotation angle (in degrees).

#include <wRotationComponent.hpp>

Inheritance diagram for wEngine::RotationComponent:



### **Public Member Functions**

• RotationComponent (float angle=0.0f)

Construct a RotationComponent with an initial angle.

- virtual ∼RotationComponent ()=default
- void setAngle (float angle)

Set the rotation angle.

• float getAngle () const

Get the current rotation angle.

· void debugPrint () const

Outputs the current angle value to the console for debugging.

### Public Member Functions inherited from wEngine::Component

- virtual ∼Component ()=default
- virtual void enable ()
- virtual void disable ()
- bool isEnabled () const

Checks whether the component is currently active.

void setParent (Entity \*parent)

Sets the parent entity of this component.

• Entity \* getParent () const

Returns the parent entity of this component.

### **Additional Inherited Members**

### Protected Member Functions inherited from wEngine::Component

· Component ()

Protected constructor to restrict instantiation to derived classes.

### 7.26.1 Detailed Description

ECS component that stores a rotation angle (in degrees).

This component can be attached to any entity that needs to be rotated relative to its logical origin. The angle is expressed in degrees (positive values correspond to counter-clockwise rotation).

### 7.26.1.0.1 Usage Example:

```
addComponent< wEngine::RotationComponent >( 45.0f ); // Rotate 45° counter-clockwise
Author
    Wilfried Koch
```

Copyright

© 2025 Wilfried Koch. All rights reserved.

### 7.26.2 Constructor & Destructor Documentation

### 7.26.2.1 RotationComponent()

Construct a RotationComponent with an initial angle.

### **Parameters**

angle	Initial rotation angle in degrees (default: 0.0f).

### 7.26.2.2 ~RotationComponent()

```
virtual wEngine::RotationComponent::~RotationComponent () [virtual], [default]
```

### 7.26.3 Member Function Documentation

### 7.26.3.1 setAngle()

Set the rotation angle.

### **Parameters**

ation angle in degrees.	angle
-------------------------	-------

### 7.26.3.2 getAngle()

```
float wEngine::RotationComponent::getAngle () const [nodiscard]
```

Get the current rotation angle.

### Returns

The stored angle in degrees.

### 7.26.3.3 debugPrint()

```
void wEngine::RotationComponent::debugPrint () const
```

Outputs the current angle value to the console for debugging.

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

- wRotationComponent.hpp
- · wRotationComponent.cpp

# 7.27 wEngine::ScaleComponent Class Reference

ECS component that defines the scaling factor for an entity in 2D space.

#include <wScaleComponent.hpp>

Inheritance diagram for wEngine::ScaleComponent:

# wEngine::Component + ~Component() + enable() + disable() + isEnabled() + setParent() + getParent() # Component() wEngine::ScaleComponent + ScaleComponent() + ~ScaleComponent() + getScale() + debugPrint()

### **Public Member Functions**

ScaleComponent (sf::Vector2f scale={ 1.0f, 1.0f })

Constructs a ScaleComponent with the given scale.

- virtual ∼ScaleComponent ()=default
- sf::Vector2f getScale () const

Retrieves the current scale factor.

• void setScale (sf::Vector2f newScale)

Sets a new scale factor.

• void debugPrint () const

Outputs the scale to standard output for debugging.

### Public Member Functions inherited from wEngine::Component

- virtual ∼Component ()=default
- virtual void enable ()
- virtual void disable ()
- bool isEnabled () const

Checks whether the component is currently active.

void setParent (Entity \*parent)

Sets the parent entity of this component.

• Entity \* getParent () const

Returns the parent entity of this component.

### **Additional Inherited Members**

### Protected Member Functions inherited from wEngine::Component

· Component ()

Protected constructor to restrict instantiation to derived classes.

### 7.27.1 Detailed Description

ECS component that defines the scaling factor for an entity in 2D space.

The scale determines how much the entity is scaled along the X and Y axes. It is typically used to transform logical coordinates into pixel coordinates.

Author

Wilfried Koch

### Copyright

© 2025 Wilfried Koch. All rights reserved.

### 7.27.2 Constructor & Destructor Documentation

### 7.27.2.1 ScaleComponent()

Constructs a ScaleComponent with the given scale.

### **Parameters**

scale	Initial scale vector (must be strictly positive, defaults is (1.0f, 1.0f).
-------	--

### **Exceptions**

std::invalid_argument	if any scale component is non-positive.
-----------------------	---

### 7.27.2.2 ~ScaleComponent()

```
\verb|virtual wEngine::ScaleComponent::~ScaleComponent () [virtual], [default]|\\
```

### 7.27.3 Member Function Documentation

### 7.27.3.1 getScale()

```
sf::Vector2f wEngine::ScaleComponent::getScale () const [nodiscard]
```

Retrieves the current scale factor.

### Returns

The scale as an sf::Vector2f (X and Y scale).

### 7.27.3.2 setScale()

Sets a new scale factor.

### **Parameters**

newScale	New scale vector (must be strictly positive).
----------	---

### **Exceptions**

### 7.27.3.3 debugPrint()

```
void wEngine::ScaleComponent::debugPrint () const
```

Outputs the scale to standard output for debugging.

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

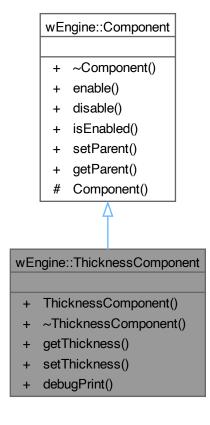
- wScaleComponent.hpp
- wScaleComponent.cpp

# 7.28 wEngine::ThicknessComponent Class Reference

ECS component that defines the thickness of a drawable object.

#include <wThicknessComponent.hpp>

Inheritance diagram for wEngine::ThicknessComponent:



### **Public Member Functions**

• ThicknessComponent (float thickness=2.0f)

Constructs the component with an initial positive thickness.

- virtual ∼ThicknessComponent ()=default
- float getThickness () const

Returns the current thickness.

void setThickness (float newThickness)

Sets a new thickness value.

• void debugPrint () const

Outputs the current thickness value to the console for debugging.

### Public Member Functions inherited from wEngine::Component

- virtual ∼Component ()=default
- virtual void enable ()
- virtual void disable ()
- bool isEnabled () const

Checks whether the component is currently active.

void setParent (Entity \*parent)

Sets the parent entity of this component.

• Entity \* getParent () const

Returns the parent entity of this component.

### **Additional Inherited Members**

# Protected Member Functions inherited from wEngine::Component

· Component ()

Protected constructor to restrict instantiation to derived classes.

### 7.28.1 Detailed Description

ECS component that defines the thickness of a drawable object.

This component stores a positive float value representing the thickness (in pixels) of lines or shapes (e.g., axes, borders). The value must be strictly positive.

### Examples:

- · A thickness of 1.0f draws a thin line.
- · A thickness of 4.0f produces a bold axis.

### **Exceptions**

std::invalid\_argument | if the value is zero or negative.

### 7.28.1.0.1 Usage Examples:

- · A thickness of 1.0f draws a thin line.
- · A thickness of 4.0f produces a bold axis.

### **Author**

Wilfried Koch

### Copyright

© 2025 Wilfried Koch. All rights reserved.

### 7.28.2 Constructor & Destructor Documentation

### 7.28.2.1 ThicknessComponent()

Constructs the component with an initial positive thickness.

### **Parameters**

thickness	Initial thickness value (default is 2.0f).
-----------	--

### **Exceptions**

std::invalid argument	if the value is zero or negative.

### 7.28.2.2 ~ThicknessComponent()

```
virtual wEngine::ThicknessComponent::~ThicknessComponent () [virtual], [default]
```

### 7.28.3 Member Function Documentation

### 7.28.3.1 getThickness()

```
float wEngine::ThicknessComponent::getThickness () const [nodiscard]
```

Returns the current thickness.

### Returns

A positive float representing the line thickness (in pixels).

### 7.28.3.2 setThickness()

Sets a new thickness value.

### **Parameters**

### Exceptions

std::invalid_argument
-----------------------

### 7.28.3.3 debugPrint()

```
void wEngine::ThicknessComponent::debugPrint () const
```

Outputs the current thickness value to the console for debugging.

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

- wThicknessComponent.hpp
- wThicknessComponent.cpp

# 7.29 wPlot2D::TitleEntity Class Reference

Represents a textual label (typically an axis title or main plot title) in a 2D plot.

#include <wTitleEntity.hpp>

Inheritance diagram for wPlot2D::TitleEntity:

### wEngine::Entity

- + Entity()
- + ~Entity()
- + getEntityID()
- + clearComponents()
- + addComponent()
- + removeComponent()
- + hasComponent()
- + getComponent()
- + requireComponent()
- + getInterfaceComponent()
- + resetEntityIDCounter()



### wPlot2D::TitleEntity

- + TitleEntity()
- + TitleEntity()
- + ~TitleEntity()
- + getCharacterSize()
- + getTextSize()
- + setTextColor()
- + setOffset()
- + setCharacterSize()
- + setFont()
- + getFrameOutlineColor() and 10 more...

### **Public Member Functions**

TitleEntity (const sf::Font &font, const std::string &title, bool hasFrame=false)

Constructs a title entity with specified font and text (UTF-8).

• TitleEntity (const sf::Font &font, const std::wstring &title, bool hasFrame=false)

Constructs a title entity with specified font and wide string (UTF-16/32).

virtual ~TitleEntity ()=default

Virtual destructor.

· unsigned int getCharacterSize () const

Returns the current character size.

sf::FloatRect getTextSize () const

Returns the local bounding box of the title text.

void setTextColor (sf::Color textColor)

Sets the text color.

void setOffset (sf::Vector2f offset)

Sets the offset relative to the base anchor position.

· void setCharacterSize (unsigned int size)

Sets the font size (character size) of the title.

void setFont (const sf::Font &font)

Sets the font reference for the title.

• sf::Color getFrameOutlineColor () const

Returns the current outline color of the frame.

• sf::Color getFrameFillColor () const

Returns the current fill color of the frame.

· float getFrameThickness () const

Returns the frame's outline thickness.

sf::Vector2f getPadding () const

Returns the internal padding of the frame.

• bool isFrameEnabled () const

Returns whether the title has a visible frame.

• void setFrameEnabled (bool enabled)

Enables or disables the visual frame.

• void setFrameOutlineColor (const sf::Color &color)

Sets the frame's outline color.

• void setFrameFillColor (const sf::Color &color)

Sets the fill color of the frame.

void setFrameThickness (float thickness)

Sets the thickness of the frame's outline.

void setPadding (sf::Vector2f padding)

Sets the internal padding of the frame (horizontal and vertical).

void render (sf::RenderWindow &window)

Renders the title and its frame (if enabled) to the window.

### Public Member Functions inherited from wEngine::Entity

- Entity ()
- virtual ~Entity ()
- unsigned int getEntityID () const

Returns the unique ID associated with this entity.

• void clearComponents ()

Removes all components currently attached to the entity.

template < typename T, typename... Args > std::shared\_ptr < T > addComponent (Args &&... args)

Adds a new component of type T to the entity.

• template<typename T>

void removeComponent ()

Removes the component of type T from the entity.

• template<typename T>

bool hasComponent () const noexcept

Checks whether the entity has a component of type T.

• template<typename T>

```
std::shared_ptr< T > getComponent () const
```

Retrieves the component of type T attached to the entity.

template<typename T>

std::shared\_ptr< T > requireComponent (const std::string &context="") const

Retrieves the component of type T and throws if it's missing.

• template<typename Interface>

```
std::shared_ptr< Interface > getInterfaceComponent () const
```

Returns the first component that implements the specified interface.

### **Additional Inherited Members**

# Static Public Member Functions inherited from wEngine::Entity

static void resetEntityIDCounter ()

Resets the global entity ID counter to zero.

### 7.29.1 Detailed Description

Represents a textual label (typically an axis title or main plot title) in a 2D plot.

A TitleEntity displays text using SFML's sf::Text, styled and positioned using ECS components (PositionComponent, OffsetComponent, ColorComponent, FontComponent). Optionally, it can display a surrounding rectangular frame (FrameEntity) with customizable outline, fill color, thickness, and padding.

### 7.29.1.0.1 Notes:

- The font passed to the constructor must remain valid during the lifetime of the entity (SFML does not copy font data).
- The frame is disabled by default unless explicitly enabled at construction or later.

See also

FrameEntity, AxisEntity, GraphicsEntity

Author

Wilfried Koch

### Copyright

© 2025 Wilfried Koch. All rights reserved.

### 7.29.2 Constructor & Destructor Documentation

### 7.29.2.1 TitleEntity() [1/2]

Constructs a title entity with specified font and text (UTF-8).

### **Parameters**

font	Reference to an externally managed font (must remain valid).
title	The string to display.
hasFrame	Whether a surrounding frame should be displayed.

### 7.29.2.2 TitleEntity() [2/2]

Constructs a title entity with specified font and wide string (UTF-16/32).

### Parameters

font	Reference to an externally managed font (must remain valid).
title	The wide string to display.
hasFrame	Whether a surrounding frame should be displayed.

### 7.29.2.3 $\sim$ TitleEntity()

```
virtual wPlot2D::TitleEntity::~TitleEntity () [virtual], [default]
```

Virtual destructor.

### 7.29.3 Member Function Documentation

### 7.29.3.1 getCharacterSize()

```
unsigned int wPlot2D::TitleEntity::getCharacterSize () const [nodiscard]
```

Returns the current character size.

### Returns

Character size (font size in pixels).

### 7.29.3.2 getTextSize()

```
sf::FloatRect wPlot2D::TitleEntity::getTextSize () const [nodiscard]
```

Returns the local bounding box of the title text.

### Returns

A FloatRect representing the size and local origin of the text.

### 7.29.3.3 setTextColor()

Sets the text color.

### **Parameters**

```
textColor | New SFML color.
```

### 7.29.3.4 setOffset()

Sets the offset relative to the base anchor position.

### **Parameters**

offset	Displacement vector in pixels.
--------	--------------------------------

### 7.29.3.5 setCharacterSize()

Sets the font size (character size) of the title.

### **Parameters**

```
size Size in pixels.
```

### 7.29.3.6 setFont()

Sets the font reference for the title.

### **Parameters**

font Reference to an externally managed font (must remain valid).

Note

The font must outlive this entity, otherwise rendering will be invalid.

### 7.29.3.7 getFrameOutlineColor()

```
sf::Color wPlot2D::TitleEntity::getFrameOutlineColor () const [nodiscard]
```

Returns the current outline color of the frame.

Returns

SFML color.

### 7.29.3.8 getFrameFillColor()

```
sf::Color wPlot2D::TitleEntity::getFrameFillColor () const [nodiscard]
```

Returns the current fill color of the frame.

Returns

SFML color.

### 7.29.3.9 getFrameThickness()

```
\verb|float wPlot2D::TitleEntity::getFrameThickness () const [nodiscard]|\\
```

Returns the frame's outline thickness.

Returns

Thickness in pixels.

### 7.29.3.10 getPadding()

```
sf::Vector2f wPlot2D::TitleEntity::getPadding () const [nodiscard]
```

Returns the internal padding of the frame.

Returns

Padding vector  $\{x, y\}$  in pixels.

### 7.29.3.11 isFrameEnabled()

```
bool wPlot2D::TitleEntity::isFrameEnabled () const [nodiscard]
```

Returns whether the title has a visible frame.

Returns

True if frame is enabled.

### 7.29.3.12 setFrameEnabled()

Enables or disables the visual frame.

**Parameters** 

```
enabled True to show the frame, false to hide it.
```

### 7.29.3.13 setFrameOutlineColor()

Sets the frame's outline color.

**Parameters** 

```
color SFML color.
```

### 7.29.3.14 setFrameFillColor()

Sets the fill color of the frame.

**Parameters** 

```
color SFML color.
```

### 7.29.3.15 setFrameThickness()

Sets the thickness of the frame's outline.

### **Parameters**

thickness Outline thickness in pixels	; <u>.</u>
---------------------------------------	------------

### 7.29.3.16 setPadding()

Sets the internal padding of the frame (horizontal and vertical).

This creates a margin between the text and the frame borders. Expressed as (horizontal, vertical) padding in pixels.

### **Parameters**

```
padding Vector of type (left/right, top/bottom) padding.
```

### 7.29.3.17 render()

Renders the title and its frame (if enabled) to the window.

The title position is computed from:

- PositionComponent (anchor point),
- OffsetComponent (displacement),
- the text's local bounds (centered origin).

If the frame is enabled, it is rendered behind the text, centered with the same anchor point and adjusted using the specified padding.

### **Parameters**

window	The target SFML render window.
--------	--------------------------------

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

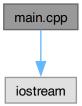
- wTitleEntity.hpp
- wTitleEntity.cpp

# **Chapter 8**

# **File Documentation**

# 8.1 main.cpp File Reference

#include <iostream>
Include dependency graph for main.cpp:



### **Functions**

• int main ()

### 8.1.1 Function Documentation

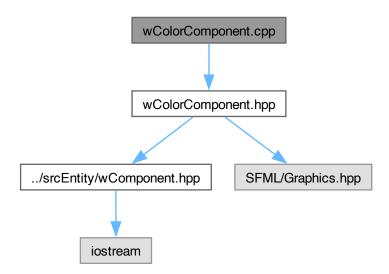
# 8.1.1.1 main()

int main ()

# 8.2 wColorComponent.cpp File Reference

Implementation of the ColorComponent class.

#include "wColorComponent.hpp"
Include dependency graph for wColorComponent.cpp:



### **Namespaces**

• namespace wEngine

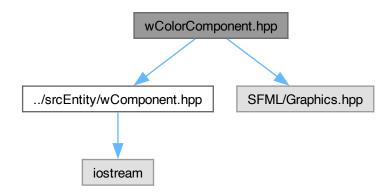
### 8.2.1 Detailed Description

Implementation of the ColorComponent class.

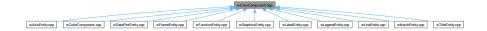
# 8.3 wColorComponent.hpp File Reference

#include "../srcEntity/wComponent.hpp"
#include <SFML/Graphics.hpp>

Include dependency graph for wColorComponent.hpp:



This graph shows which files directly or indirectly include this file:



### Classes

• class wEngine::ColorComponent

ECS component that holds a color value.

### **Namespaces**

namespace wEngine

# 8.4 wColorComponent.hpp

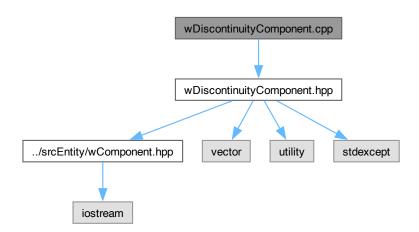
### Go to the documentation of this file.

```
00015 #pragma GCC diagnostic ignored "-Wswitch-default"
00016 #include <SFML/Graphics.hpp>
00017 #pragma GCC diagnostic pop
00018
00019 namespace wEngine
00020 {
00045
         class ColorComponent : public Component
00046
00047
             public:
                 ColorComponent( sf::Color color = sf::Color::Black );
00052
00053
00054
00055
                 * @brief Virtual destructor.
00056
00057
                 virtual ~ColorComponent() = default;
00058
00063
                 [[nodiscard]] sf::Color getColor() const;
00064
00069
                 void setColor( sf::Color newColor );
00070
00074
                 void debugPrint() const;
00075
             private:
00076
                 sf::Color mColor;
00077
        };
00078
00079 }//End of namespace wEngine
08000
00081 #endif
```

# 8.5 wDiscontinuityComponent.cpp File Reference

Implementation of the DiscontinuityComponent class.

#include "wDiscontinuityComponent.hpp"
Include dependency graph for wDiscontinuityComponent.cpp:



### **Namespaces**

• namespace wEngine

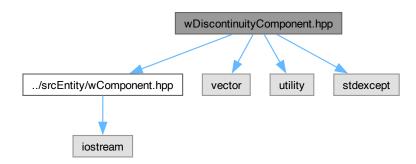
### 8.5.1 Detailed Description

Implementation of the DiscontinuityComponent class.

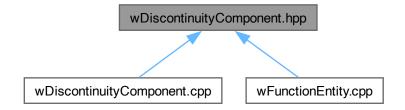
# 8.6 wDiscontinuityComponent.hpp File Reference

```
#include "../srcEntity/wComponent.hpp"
#include <vector>
#include <utility>
#include <stdexcept>
```

Include dependency graph for wDiscontinuityComponent.hpp:



This graph shows which files directly or indirectly include this file:



### Classes

class wEngine::DiscontinuityComponent
 ECS component that manages excluded intervals for function plotting.

### Namespaces

• namespace wEngine

# 8.7 wDiscontinuityComponent.hpp

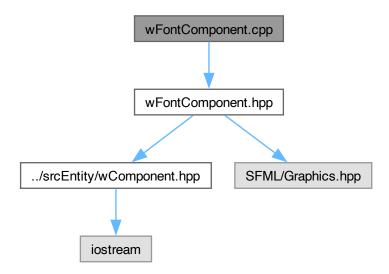
Go to the documentation of this file.

```
00001 /*
00002
00003 Created by Wilfried Koch.
00004 Copyright @ 2025 Wilfried Koch. All rights reserved.
00006 */
00007
00008 #ifndef W_DISCONTINUITY_COMPONENT_HPP
00009 #define W DISCONTINUITY COMPONENT HPP
00010
00011 #include "../srcEntity/wComponent.hpp"
00013 #include <vector>
00014 #include <utility>
00015 #include <stdexcept>
00016
00017 namespace wEngine
00018 {
00019
00033
          class DiscontinuityComponent : public Component
00034
00035
              public:
00036
                  * @brief Default constructor.
00038
00039
                  DiscontinuityComponent() = default;
00040
00041
00042
                   * @brief Virtual destructor.
00044
                  virtual ~DiscontinuityComponent() = default;
00045
00050
                  [[nodiscard]] const std::vector< std::pair< double,double > >& getExcludedIntervals()
     const;
00051
00058
                  void addExcludedInterval( double min, double max );
00059
00063
                  void clearExcludedIntervals();
00064
00070
                  [[nodiscard]] bool isInExcludedInterval( double x ) const;
00071
00072
00073
                   * @brief Outputs the excluded intervals to the console for debugging.
00074
00075
                  void debugPrint() const;
00076
              private:
00077
                  std::vector< std::pair< double, double > > mExcludedIntervals;
00078
         };
00079
00080 }//End of namespace wEngine
00081
00082 #endif
```

# 8.8 wFontComponent.cpp File Reference

Implementation of the FontComponent class.

#include "wFontComponent.hpp"
Include dependency graph for wFontComponent.cpp:



### **Namespaces**

• namespace wEngine

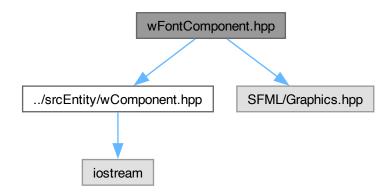
### 8.8.1 Detailed Description

Implementation of the FontComponent class.

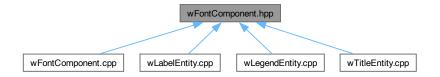
# 8.9 wFontComponent.hpp File Reference

```
#include "../srcEntity/wComponent.hpp"
#include <SFML/Graphics.hpp>
```

Include dependency graph for wFontComponent.hpp:



This graph shows which files directly or indirectly include this file:



### Classes

• class wEngine::FontComponent

Holds a reference to an SFML font for rendering text.

### **Namespaces**

• namespace wEngine

# 8.10 wFontComponent.hpp

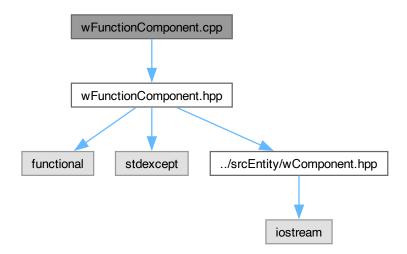
Go to the documentation of this file.

```
00007
00008 #ifndef W_FONT_COMPONENT_HPP
00009 #define W_FONT_COMPONENT_HPP
00010
00011 #include "../srcEntity/wComponent.hpp"
00012
00013 #pragma GCC diagnostic push
00014 #pragma GCC diagnostic ignored "-Wfloat-equal"
00015 #pragma GCC diagnostic ignored "-Wswitch-default"
00016 #include <SFML/Graphics.hpp>
00017 #pragma GCC diagnostic pop
00018
00019 namespace wEngine
00020 {
00021
00035
          class FontComponent : public Component
00036
00037
             public:
                  explicit FontComponent( const sf::Font& font );
00043
00047
                  ~FontComponent() override = default;
00048
00053
                  const sf::Font& getFont() const;
00054
00060
                  void setFont( const sf::Font& font );
00061
00065
                  void debugPrint() const;
00066
00067
            private:
00068
                 const sf::Font* mFont;
00069
        };
00071 }//End of namespace wEngine
00072
00073 #endif
```

# 8.11 wFunctionComponent.cpp File Reference

Implementation of the FunctionComponent class.

#include "wFunctionComponent.hpp"
Include dependency graph for wFunctionComponent.cpp:



### **Namespaces**

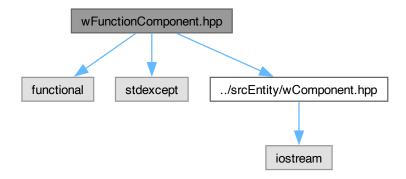
namespace wEngine

# 8.11.1 Detailed Description

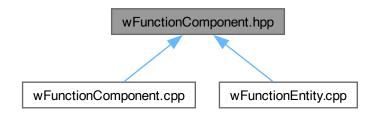
Implementation of the FunctionComponent class.

# 8.12 wFunctionComponent.hpp File Reference

```
#include <functional>
#include <stdexcept>
#include "../srcEntity/wComponent.hpp"
Include dependency graph for wFunctionComponent.hpp:
```



This graph shows which files directly or indirectly include this file:



### Classes

• class wEngine::FunctionComponent

ECS component that stores a mathematical function f(x).

### **Namespaces**

namespace wEngine

# 8.13 wFunctionComponent.hpp

Go to the documentation of this file.

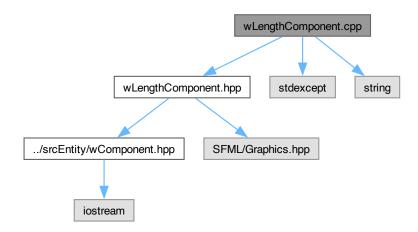
```
00001 /*
00002
00003 Created by Wilfried Koch.
00004 Copyright @ 2025 Wilfried Koch. All rights reserved.
00006 */
00007
00008 #ifndef W_FUNCTION_COMPONENT_HPP
00009 #define W_FUNCTION_COMPONENT_HPP
00011 #include <functional>
00012 #include <stdexcept>
00013
00014 #include "../srcEntity/wComponent.hpp"
00015
00016 namespace wEngine
00017 {
00018
00031
         class FunctionComponent : public Component
00032
00039
                 FunctionComponent( std::function< double( double ) > function );
00040
00041
                  * @brief Virtual destructor.
00042
00043
00044
                  virtual ~FunctionComponent() = default;
00045
00052
                  [[nodiscard]] double calculate( double x ) const;
00053
00054
00055
                  * @brief Prints a message confirming that the function is set.
00056
                  void debugPrint() const;
00058
00059
                 std::function< double( double ) > mFunction;
00060
       };
00061
00062 }//End of namespace wEngine
00064 #endif
```

# 8.14 wLengthComponent.cpp File Reference

Implementation of the LengthComponent class.

```
#include "wLengthComponent.hpp"
#include <stdexcept>
#include <string>
```

Include dependency graph for wLengthComponent.cpp:



### Namespaces

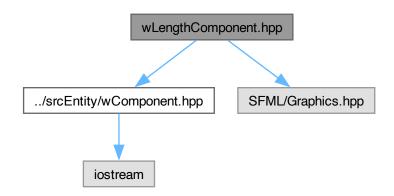
• namespace wEngine

### 8.14.1 Detailed Description

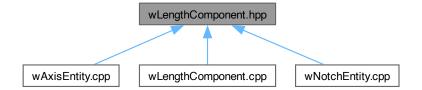
Implementation of the LengthComponent class.

# 8.15 wLengthComponent.hpp File Reference

#include "../srcEntity/wComponent.hpp"
#include <SFML/Graphics.hpp>
Include dependency graph for wLengthComponent.hpp:



This graph shows which files directly or indirectly include this file:



### **Classes**

class wEngine::LengthComponent
 ECS component that defines the length of a drawable object.

### Namespaces

• namespace wEngine

# 8.16 wLengthComponent.hpp

Go to the documentation of this file.

```
00001 /*
00002
00003 Created by Wilfried Koch. 00004 Copyright @ 2025 Wilfried Koch. All rights reserved.
00005
00006 */
00007
00008 #ifndef W_LENGTH_COMPONENT_HPP
00009 #define W_LENGTH_COMPONENT_HPP
00010
00011 #include "../srcEntity/wComponent.hpp"
00012
00013 #pragma GCC diagnostic push
00014 #pragma GCC diagnostic ignored "-Wfloat-equal"
00015 #pragma GCC diagnostic ignored "-Wswitch-default"
00016 #include <SFML/Graphics.hpp>
00017 #pragma GCC diagnostic pop
00018
00019 namespace wEngine
00020 {
00021
          class LengthComponent : public Component
00035
00036
00037
              public:
00043
                   LengthComponent( float length = 2.0f);
0\,0\,0\,4\,4
00045
                   * @brief Virtual destructor.
00046
00047
00048
                   virtual ~LengthComponent() = default;
00049
00054
                   [[nodiscard]] float getLength() const;
00055
00061
                   void setLength( float newLength );
00062
00066
                   void debugPrint() const;
00067
              private:
```

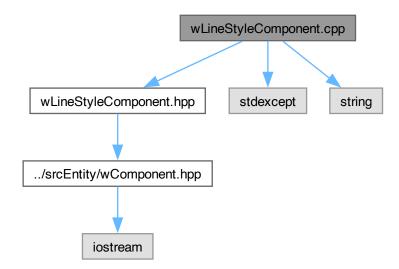
```
00068 float mLength;
00069
00075 void validatePositive( float value ) const;
00076 };
00077
00078 }//End of namespace wEngine
00079
00080 #endif
```

# 8.17 wLineStyleComponent.cpp File Reference

Implementation of the LineStyleComponent class.

```
#include "wLineStyleComponent.hpp"
#include <stdexcept>
#include <string>
```

Include dependency graph for wLineStyleComponent.cpp:



### **Namespaces**

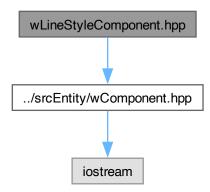
· namespace wEngine

### 8.17.1 Detailed Description

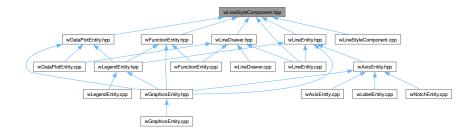
Implementation of the LineStyleComponent class.

# 8.18 wLineStyleComponent.hpp File Reference

#include "../srcEntity/wComponent.hpp"
Include dependency graph for wLineStyleComponent.hpp:



This graph shows which files directly or indirectly include this file:



### Classes

• class wEngine::LineStyleComponent

ESC component that defines the style of a line (solid, dotted, dashed).

### Namespaces

• namespace wEngine

### 8.19 wLineStyleComponent.hpp

### Go to the documentation of this file.

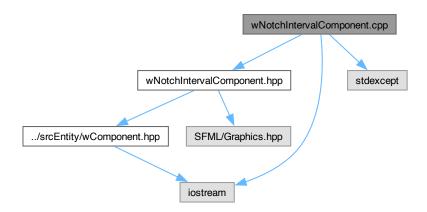
```
00002
00003 Created by Wilfried Koch.
00004 Copyright @ 2025 Wilfried Koch. All rights reserved.
00006 */
00007
00008 #ifndef W_LINE_STYLE_COMPONENT_HPP
00009 #define W LINE STYLE COMPONENT HPP
00010
00011 #include "../srcEntity/wComponent.hpp"
00013 namespace wEngine
00014 {
00015
00027
          class LineStyleComponent : public Component
00028
              public:
00034
                  enum class LineStyle
00035
00036
                      Solid.
00037
                      Dotted,
00038
                      Dashed
00039
00040
00045
                  LineStyleComponent( LineStyle style = LineStyle::Solid );
00046
00047
00048
                   * @brief Virtual destructor.
00050
                  virtual ~LineStyleComponent() = default;
00051
00056
                  [[nodiscard]] LineStyle getStyle() const;
00057
00062
                  void setStyle( LineStyle style );
00063
00068
                  [[nodiscard]] float getDashLength() const;
00069
00075
                  void setDashLength( float dashLength );
00076
00081
                  [[nodiscard]] float getGapLength() const;
00082
00088
                  void setGapLength( float gapLength);
00089
00090
00091
                   \star @brief Outputs the current style and parameters to the console.
00092
00093
                  void debugPrint() const;
00094
              private:
00095
                  LineStyle mStyle;
00096
                  float mDashLength;
00097
                  float mGapLength;
00098
        };
00099
00100 }//End of namespace wEngine
00101
00102 #endif
```

# 8.20 wNotchIntervalComponent.cpp File Reference

Implementation of the wNotchIntervalComponent class.

```
#include "wNotchIntervalComponent.hpp"
#include <iostream>
#include <stdexcept>
```

Include dependency graph for wNotchIntervalComponent.cpp:



### **Namespaces**

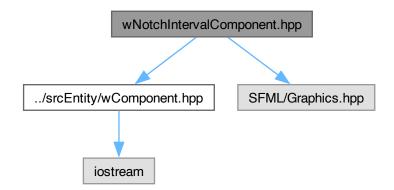
namespace wEngine

### 8.20.1 Detailed Description

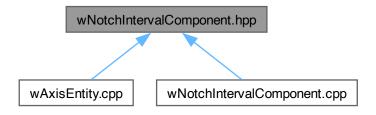
Implementation of the wNotchIntervalComponent class.

# 8.21 wNotchIntervalComponent.hpp File Reference

#include "../srcEntity/wComponent.hpp"
#include <SFML/Graphics.hpp>
Include dependency graph for wNotchIntervalComponent.hpp:



This graph shows which files directly or indirectly include this file:



### Classes

class wEngine::NotchIntervalComponent
 ECS component that defines the interval between notches on an axis.

### **Namespaces**

· namespace wEngine

# 8.22 wNotchIntervalComponent.hpp

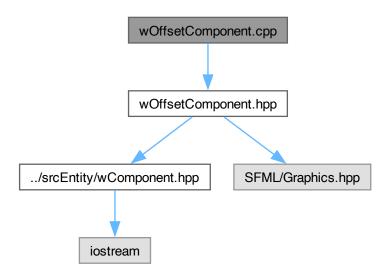
Go to the documentation of this file.

```
00001 /*
00002
00003 Created by Wilfried Koch.
00004 Copyright @ 2025 Wilfried Koch. All rights reserved.
00005
00006 */
00007
00008 #ifndef W_NOTCH_INTERVAL_COMPONENT_HPP
00009 #define W_NOTCH_INTERVAL_COMPONENT_HPP
00010
00011 #include "../srcEntity/wComponent.hpp"
00012
00013 #pragma GCC diagnostic push
00014 #pragma GCC diagnostic ignored "-Wfloat-equal"
00015 #pragma GCC diagnostic ignored "-Wswitch-default"
00016 #include <SFML/Graphics.hpp>
00017 #pragma GCC diagnostic pop
00018
00019 namespace wEngine
00020 {
00021
00035
          class NotchIntervalComponent : public Component
00036
00037
              public:
00043
                  NotchIntervalComponent( float interval = 1.0f );
00044
00045
00046
                  * @brief Virtual destructor.
00047
00048
                  virtual ~NotchIntervalComponent() = default;
00049
                  [[nodiscard]] float getInterval() const;
00054
00055
00061
                  void setInterval( float newInterval );
```

# 8.23 wOffsetComponent.cpp File Reference

Implementation of the OffsetComponent class.

```
#include "wOffsetComponent.hpp"
Include dependency graph for wOffsetComponent.cpp:
```



### **Namespaces**

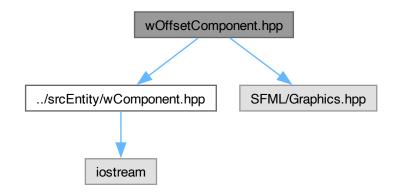
• namespace wEngine

### 8.23.1 Detailed Description

Implementation of the OffsetComponent class.

# 8.24 wOffsetComponent.hpp File Reference

#include "../srcEntity/wComponent.hpp"
#include <SFML/Graphics.hpp>
Include dependency graph for wOffsetComponent.hpp:



This graph shows which files directly or indirectly include this file:



### Classes

class wEngine::OffsetComponent

ECS component that defines a logical coordinate offset.

### **Namespaces**

• namespace wEngine

# 8.25 wOffsetComponent.hpp

Go to the documentation of this file.

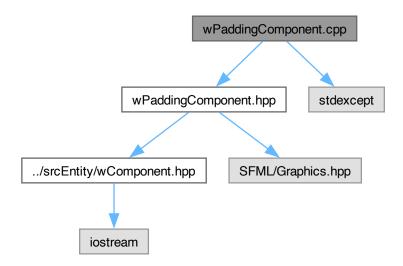
```
00005
00006 */
00007
00008 #ifndef W_OFFSET_COMPONENT_HPP
00009 #define W_OFFSET_COMPONENT_HPP
00011 #include "../srcEntity/wComponent.hpp"
00012
00013 #pragma GCC diagnostic push 00014 #pragma GCC diagnostic ignored "-Wfloat-equal"
00015 #pragma GCC diagnostic ignored "-Wswitch-default"
00016 #include <SFML/Graphics.hpp>
00017 #pragma GCC diagnostic pop
00018
00019 namespace wEngine
00020 {
00021
          class OffsetComponent : public Component
00041
              public:
00042
                  OffsetComponent( sf::Vector2f offset = sf::Vector2f( 0.0f, 0.0f ) );
00047
00048
00049
00050
                   * @brief Virtual destructor.
00051
00052
                  virtual ~OffsetComponent() = default;
00053
                  [[nodiscard]] sf::Vector2f getOffset() const;
00058
00059
00064
                  void setOffset( sf::Vector2f offset );
00065
00070
                  void addOffset( sf::Vector2f delta );
00071
00075
                  void debugPrint() const;
00076
              private:
00077
                  sf::Vector2f mOffset;
         };
00079
00080 }//End of namespace wEngine
00081
00082 #endif
```

# 8.26 wPaddingComponent.cpp File Reference

Implementation of the PaddingComponent class.

```
#include "wPaddingComponent.hpp"
#include <stdexcept>
```

Include dependency graph for wPaddingComponent.cpp:



### **Namespaces**

• namespace wEngine

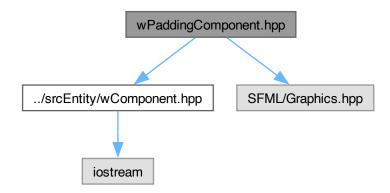
### 8.26.1 Detailed Description

Implementation of the PaddingComponent class.

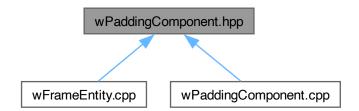
# 8.27 wPaddingComponent.hpp File Reference

#include "../srcEntity/wComponent.hpp"
#include <SFML/Graphics.hpp>

Include dependency graph for wPaddingComponent.hpp:



This graph shows which files directly or indirectly include this file:



### Classes

• class wEngine::PaddingComponent ECS component representing internal padding for UI-like elements.

### **Namespaces**

• namespace wEngine

# 8.28 wPaddingComponent.hpp

### Go to the documentation of this file.

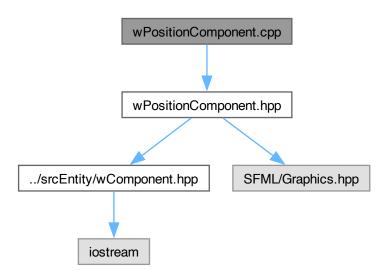
00001 #ifndef W\_PADDING\_COMPONENT\_HPP 00002 #define W\_PADDING\_COMPONENT\_HPP

```
00004 #include "../srcEntity/wComponent.hpp"
00005
00006 #pragma GCC diagnostic push
00007 #pragma GCC diagnostic ignored "-Wfloat-equal"
00008 #pragma GCC diagnostic ignored "-Wswitch-default"
00009 #include <SFML/Graphics.hpp>
00010 #pragma GCC diagnostic pop
00011
00012 namespace wEngine
00013 {
00014
00031
           class PaddingComponent : public Component
00032
00033
               public:
00038
                    PaddingComponent( sf::Vector2f padding = sf::Vector2f( 0.0f, 0.0f ) );
00039
00040
00041
                    * @brief Virtual destructor.
00042
00043
                    virtual ~PaddingComponent() = default;
00044
00049
                    void setPadding( sf::Vector2f padding );
00050
00055
                    [[nodiscard]] sf::Vector2f getPadding() const;
00056
00060
                   void debugPrint() const;
00061
              private:
                   sf::Vector2f mPadding;
00062
00063
00069
                    void validatePositive( const sf::Vector2f& value ) const;
00070
          };
00071
00072 }//End of namespace wEngine
00073
00074 #endif
```

# 8.29 wPositionComponent.cpp File Reference

Implementation of the PositionComponent class.

#include "wPositionComponent.hpp"
Include dependency graph for wPositionComponent.cpp:



### **Namespaces**

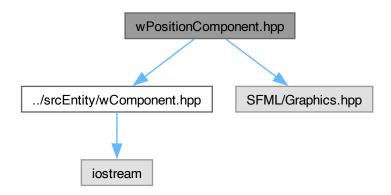
namespace wEngine

### 8.29.1 Detailed Description

Implementation of the PositionComponent class.

# 8.30 wPositionComponent.hpp File Reference

#include "../srcEntity/wComponent.hpp"
#include <SFML/Graphics.hpp>
Include dependency graph for wPositionComponent.hpp:



This graph shows which files directly or indirectly include this file:



#### Classes

class wEngine::PositionComponent
 ECS component storing the position of an entity in 2D space and supports movement tracking.

### **Namespaces**

• namespace wEngine

### 8.31 wPositionComponent.hpp

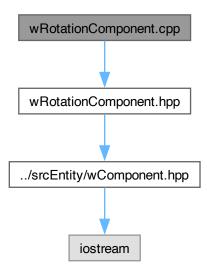
Go to the documentation of this file.

```
00001 /*
00002
00003 Created by Wilfried Koch.
00004 Copyright @ 2025 Wilfried Koch. All rights reserved.
00006 */
00007
00008 #ifndef W_POSITION_COMPONENT_HPP
00009 #define W_POSITION_COMPONENT_HPP
00010
00011 #include "../srcEntity/wComponent.hpp"
00013 #pragma GCC diagnostic push
00014 #pragma GCC diagnostic ignored "-Wfloat-equal"
00015 #pragma GCC diagnostic ignored "-Wswitch-default"
00016 #include <SFML/Graphics.hpp>
00017 #pragma GCC diagnostic pop
00018
00019 namespace wEngine
00020 {
00021
00043
          class PositionComponent : public Component
00044
00045
00050
                   PositionComponent( sf::Vector2f position = sf::Vector2f( 0.0f, 0.0f ) );
00051
00052
                    * @brief Virtual destructor.
00053
00054
00055
                   virtual ~PositionComponent() = default;
00056
00061
                   [[nodiscard]] sf::Vector2f getPosition() const;
00062
00067
                   [[nodiscard]] sf::Vector2f getLastPosition() const;
00068
                   void setPosition( sf::Vector2f newPosition );
00074
00081
                   void move( const sf::Vector2f& offset );
00082
00086
                   void debugPrint() const;
00087
              private:
                  sf::Vector2f mPosition;
00088
00089
                   sf::Vector2f mLastPosition;
00090
00091
00092 }//End of namespace wEngine
00093
00094 #endif
```

# 8.32 wRotationComponent.cpp File Reference

Implementation of the RotationComponent class.

#include "wRotationComponent.hpp"
Include dependency graph for wRotationComponent.cpp:



### Namespaces

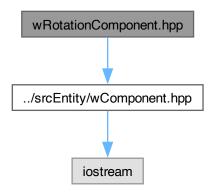
• namespace wEngine

# 8.32.1 Detailed Description

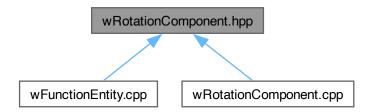
Implementation of the RotationComponent class.

# 8.33 wRotationComponent.hpp File Reference

#include "../srcEntity/wComponent.hpp"
Include dependency graph for wRotationComponent.hpp:



This graph shows which files directly or indirectly include this file:



### Classes

class wEngine::RotationComponent
 ECS component that stores a rotation angle (in degrees).

### Namespaces

• namespace wEngine

### 8.34 wRotationComponent.hpp

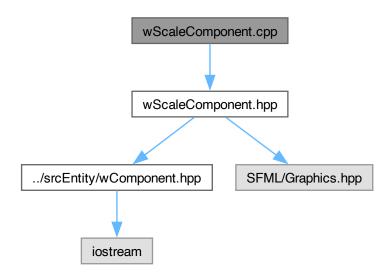
Go to the documentation of this file.

```
00001 /*
00002
00003 Created by Wilfried Koch.
00004 Copyright @ 2025 Wilfried Koch. All rights reserved.
00006 */
00007
00008 #ifndef W_ROTATION_COMPONENT_HPP
00009 #define W_ROTATION_COMPONENT_HPP
00010
00011 #include "../srcEntity/wComponent.hpp"
00013 namespace wEngine
00014 {
00015
00033
         class RotationComponent : public Component
00034
        {
00035
00040
                 RotationComponent( float angle = 0.0f );
00041
00042
                  * @brief Virtual destructor.
00043
00044
00045
                 virtual ~RotationComponent() = default;
00046
00051
                 void setAngle( float angle );
00052
00057
                 [[nodiscard]] float getAngle() const;
00058
00062
                 void debugPrint() const;
00063
00064
           private:
00065
                 float mAngle;
00066
       };
00067
00068 }//End of namespace wEngine
00069
00070 #endif
```

# 8.35 wScaleComponent.cpp File Reference

Implementation of the ScaleComponent class.

#include "wScaleComponent.hpp"
Include dependency graph for wScaleComponent.cpp:



### **Namespaces**

• namespace wEngine

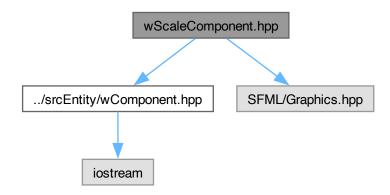
### 8.35.1 Detailed Description

Implementation of the ScaleComponent class.

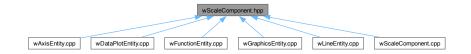
# 8.36 wScaleComponent.hpp File Reference

#include "../srcEntity/wComponent.hpp"
#include <SFML/Graphics.hpp>

Include dependency graph for wScaleComponent.hpp:



This graph shows which files directly or indirectly include this file:



#### **Classes**

• class wEngine::ScaleComponent

ECS component that defines the scaling factor for an entity in 2D space.

### **Namespaces**

• namespace wEngine

# 8.37 wScaleComponent.hpp

### Go to the documentation of this file.

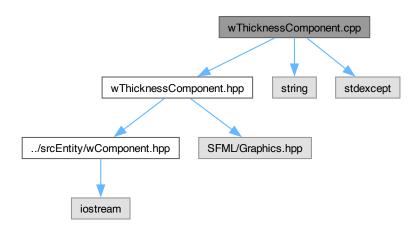
```
00013 #pragma GCC diagnostic push
00014 #pragma GCC diagnostic ignored "-Wfloat-equal" 00015 #pragma GCC diagnostic ignored "-Wswitch-default"
00016 #include <SFML/Graphics.hpp>
00017 #pragma GCC diagnostic pop
00019 namespace wEngine
00020 {
00021
          class ScaleComponent : public Component
00033
00034
00035
              public:
00041
                   ScaleComponent( sf::Vector2f scale = { 1.0f, 1.0f } );
00042
00043
                   * @brief Virtual destructor.
00044
00045
00046
                   virtual ~ScaleComponent() = default;
00047
00052
                   [[nodiscard]] sf::Vector2f getScale() const;
00053
                   void setScale( sf::Vector2f newScale );
00059
00060
00064
                   void debugPrint() const;
00065
              private:
00066
                   sf::Vector2f mScale;
00067
                   void validatePositive( const sf::Vector2f& value ) const;
00073
00074
          };
00075
00076 }//End of namespace wEngine
00077
00078 #endif
```

## 8.38 wThicknessComponent.cpp File Reference

Implementation of the ThicknessComponent class.

```
#include "wThicknessComponent.hpp"
#include <string>
#include <stdexcept>
```

Include dependency graph for wThicknessComponent.cpp:



#### **Namespaces**

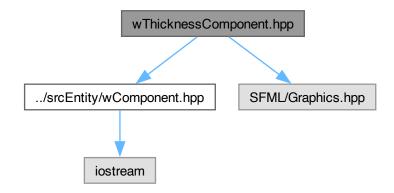
• namespace wEngine

### 8.38.1 Detailed Description

Implementation of the ThicknessComponent class.

# 8.39 wThicknessComponent.hpp File Reference

#include "../srcEntity/wComponent.hpp"
#include <SFML/Graphics.hpp>
Include dependency graph for wThicknessComponent.hpp:



This graph shows which files directly or indirectly include this file:



#### **Classes**

• class wEngine::ThicknessComponent

ECS component that defines the thickness of a drawable object.

### **Namespaces**

· namespace wEngine

## 8.40 wThicknessComponent.hpp

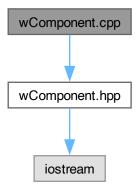
#### Go to the documentation of this file.

```
00001 /*
00002
00003 Created by Wilfried Koch.
00004 Copyright @ 2025 Wilfried Koch. All rights reserved.
00006 */
00007
00008 #ifndef W_THICKNESS_COMPONENT_HPP
00009 #define W_THICKNESS_COMPONENT_HPP
00010
00011 #include "../srcEntity/wComponent.hpp"
00013 #pragma GCC diagnostic push
00014 #pragma GCC diagnostic ignored "-Wfloat-equal"
00014 #pragma GCC diagnostic ignored "-Wswitch-default" 00016 #include <SFML/Graphics.hpp>
00017 #pragma GCC diagnostic pop
00018
00019 namespace wEngine
00020 {
00021
00043
          class ThicknessComponent : public Component
00044
00045
00051
                  ThicknessComponent( float thickness = 2.0f );
00052
00053
                   * @brief Virtual destructor.
00054
00055
                  virtual ~ThicknessComponent() = default;
00057
00062
                 [[nodiscard]] float getThickness() const;
00063
00069
                  void setThickness( float newThickness);
00070
00074
                  void debugPrint() const;
00075
             private:
00076
                  float mThickness;
00077
00078
00079
                   * @brief Validates that the thickness value is strictly positive.
                   * @param value The thickness value to validate.
08000
00081
                   * @throws std::invalid_argument if value <= 0.
00082
00083
                  void validatePositive( float value ) const;
00084
        };
00085
00086 }//End of namespace wEngine
00088 #endif
```

# 8.41 wComponent.cpp File Reference

Implementation of the Component class.

#include "wComponent.hpp"
Include dependency graph for wComponent.cpp:



### **Namespaces**

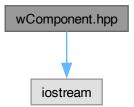
• namespace wEngine

### 8.41.1 Detailed Description

Implementation of the Component class.

# 8.42 wComponent.hpp File Reference

#include <iostream>
Include dependency graph for wComponent.hpp:



This graph shows which files directly or indirectly include this file:



#### Classes

· class wEngine::Component

Abstract base class for all ECS components.

#### **Namespaces**

· namespace wEngine

## 8.43 wComponent.hpp

Go to the documentation of this file.

```
00001 /*
00002
00003 Created by Wilfried Koch.
00004 Copyright @ 2025 Wilfried Koch. All rights reserved.
00005
00006 */
00007
00008 #ifndef W_COMPONENT_HPP
00009 #define W_COMPONENT_HPP
00010
00011 #include <iostream>
00012
00013 namespace wEngine
00014 {
00015
00016
          class Entity;
00017
          class Component
00031
00032
00033
              public:
00034
                   * @brief Virtual destructor.
00035
00036
00037
                  virtual ~Component() = default;
00038
00039
00040
                   * @brief Enables the component (makes it active).
00041
00042
                  virtual void enable();
00043
00044
00045
                   * @brief Disables the component (makes it inactive).
00046
00047
                  virtual void disable();
00048
                  [[nodiscard]] bool isEnabled() const;
00053
00054
00059
                  void setParent( Entity* parent );
00060
00065
                  [[nodiscard]] Entity* getParent() const;
00066
00067
              protected:
00068
00072
                  Component ( );
00073
00074
              private:
```

```
00075 bool mEnabled;

00076 Entity* mParent;

00077 };

00078

00079 }// End of namespace wEngine

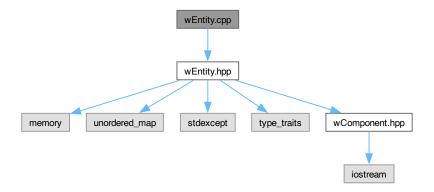
00080

00081 #endif
```

# 8.44 wEntity.cpp File Reference

Implementation of the Entity class.

```
#include "wEntity.hpp"
Include dependency graph for wEntity.cpp:
```



### **Namespaces**

• namespace wEngine

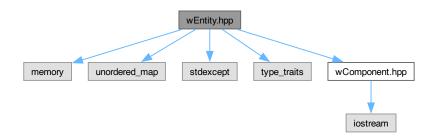
### 8.44.1 Detailed Description

Implementation of the Entity class.

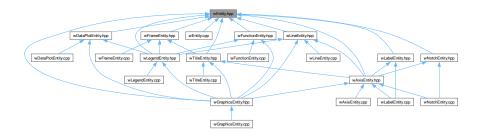
# 8.45 wEntity.hpp File Reference

```
#include <memory>
#include <unordered_map>
#include <stdexcept>
#include <type_traits>
```

#include "wComponent.hpp"
Include dependency graph for wEntity.hpp:



This graph shows which files directly or indirectly include this file:



### Classes

class wEngine::Entity

Represents an entity in the ECS (Entity-Component System) architecture.

### **Namespaces**

• namespace wEngine

#### **Functions**

- std::size\_t wEngine::getNextComponentTypeID ()
- template<typename ComponentType>
   std::size\_t wEngine::getComponentTypeID () noexcept

8.46 wEntity.hpp 183

## 8.46 wEntity.hpp

### Go to the documentation of this file.

```
00001 /*
00002
00003 Created by Wilfried Koch.
00004 Copyright @ 2025 Wilfried Koch. All rights reserved.
00006 */
00007
00008 #ifndef W_ENTITY_HPP
00009 #define W_ENTITY_HPP
00010
00011 #include <memory>
00012 #include <unordered_map>
00013 #include <stdexcept>
00014 #include <type_traits>
00015
00016 #include "wComponent.hpp"
00017
00018 namespace wEngine
00019 {
00020
00021
00022
          * @brief Generates a new unique component type ID.
00023
          * @return A unique integer for component type identification.
00024
00025
          inline std::size_t getNextComponentTypeID()
00026
             static std::size_t componentTypeCounter = 0;
00027
00028
             return componentTypeCounter++;
00029
         }
00030
00031
00032
          \star @brief Returns the unique type ID associated with a specific component type.
00033
          \star @tparam ComponentType The type of the component.
00034
          * @return A unique integer identifying the component type.
00035
00036
          template< typename ComponentType >
00037
          std::size_t getComponentTypeID() noexcept
00038
00039
             static std::size_t componentTypeID = getNextComponentTypeID();
00040
             return componentTypeID;
00041
         }
00042
00059
          class Entity
00060
00061
              public:
00062
00063
                  * @brief Default constructor. Generates a new entity with a unique ID.
00064
00065
                 Entity();
00066
00067
00068
                  \star @brief Destructor. Disables all attached components before destruction.
00069
00070
                 virtual ~Entity();
00071
00076
                 [[nodiscard]] unsigned int getEntityID() const;
00077
00081
                 void clearComponents();
00082
00089
                 static void resetEntityIDCounter();
00090
00103
                 template< typename T, typename... Args >
00104
                  std::shared_ptr< T > addComponent( Args&&... args )
00105
                      static_assert( std::is_base_of< Component, T >::value, "T must be derived from
00106
     Component");
00107
00108
                     auto typeID = getComponentTypeID< T >( );
00109
                      if (mComponents.contains( typeID ))
00110
                         00111
00112
00113
00114
00115
00116
                     auto component = std::make_shared< T >( std::forward< Args >( args )... );
00117
                     component->setParent( this );
00118
                     mComponents[ typeID ] = component;
00119
00120
                     return component;
```

```
}
00122
00130
                  template< typename T >
                  void removeComponent()
00131
00132
00133
                      static assert ( std::is base of < Component, T >::value, "T must be derived from
      Component");
00134
00135
                      auto typeID = getComponentTypeID< T >( );
00136
                      mComponents.erase( typeID );
                  }
00137
00138
00145
                  template< typename T >
00146
                  [[nodiscard]] bool hasComponent() const noexcept
00147
00148
                       static_assert( std::is_base_of< Component, T >::value, "T must be derived from
     Component");
00149
00150
                      return mComponents.contains( getComponentTypeID< T >( ) );
00151
                  }
00152
00159
                  template< typename T >
00160
                  [[nodiscard]] std::shared_ptr< T > getComponent() const
00161
                      static_assert( std::is_base_of< Component, T >::value, "T must be derived from
00162
     Component");
00163
00164
                      auto typeID = getComponentTypeID< T >( );
00165
                      auto it = mComponents.find( typeID );
                      if (it != mComponents.end())
00166
00167
00168
                           return std::dynamic_pointer_cast< T >( it->second );
00169
00170
00171
                      return nullptr;
                  }
00172
00173
00186
                  template< typename T >
00187
                  [[nodiscard]] std::shared_ptr< T > requireComponent( const std::string& context = "")
00188
00189
                      static_assert( std::is_base_of< Component, T >::value, "T must be derived from
     Component");
00190
00191
                       if (!hasComponent< T >( ))
00192
                          std::string msg = "Missing required component: "; msg += typeid( T ).name( );
00193
00194
                           if (!context.empty())
00195
00196
00197
                              msg += " in context: " + context;
00198
00199
                           throw std::runtime_error( msg );
00200
                      }
00201
00202
                      return getComponent< T >( );
00203
                  }
00204
00214
                  template< typename Interface >
                   [[nodiscard]] std::shared_ptr< Interface > getInterfaceComponent() const
00215
00216
00217
                       for (const auto& [ typeID, component ] : mComponents)
00218
00219
                           auto interfaceComponent = std::dynamic_pointer_cast< Interface >( component );
00220
                           if (interfaceComponent)
00221
                           {
00222
                               return interfaceComponent;
00223
                           }
00224
00225
                       return nullptr;
00226
00227
00228
              private:
                  unsigned int mEntityID:
00229
00230
                  std::unordered map< std::size t, std::shared ptr< Component > > mComponents;
00231
00232
                  static unsigned int sEntityIDCounter;
00233
                  static unsigned int generateNextEntityID();
00234
          } ;
00235
00236 }// End of namespace wEngine
00238 #endif
```

## 8.47 wAxisEntity.cpp File Reference

Implementation of the AxisEntity class.

```
#include "wAxisEntity.hpp"
#include "../srcComponents/wPositionComponent.hpp"
#include "../srcComponents/wScaleComponent.hpp"
#include "../srcComponents/wColorComponent.hpp"
#include "../srcComponents/wThicknessComponent.hpp"
#include "../srcComponents/wOffsetComponent.hpp"
#include "../srcComponents/wNotchIntervalComponent.hpp"
#include "../srcComponents/wLengthComponent.hpp"
Include dependency graph for wAxisEntity.cpp:
```



#### **Namespaces**

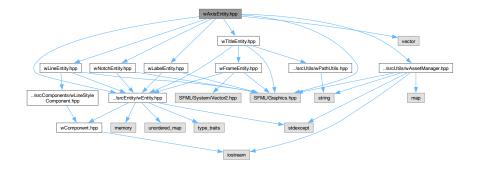
namespace wPlot2D

### 8.47.1 Detailed Description

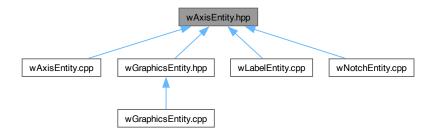
Implementation of the AxisEntity class.

## 8.48 wAxisEntity.hpp File Reference

```
#include "../srcEntity/wEntity.hpp"
#include "../srcUtils/wAssetManager.hpp"
#include "wLineEntity.hpp"
#include "wNotchEntity.hpp"
#include "wTitleEntity.hpp"
#include "wLabelEntity.hpp"
#include <SFML/Graphics.hpp>
#include <vector>
Include dependency graph for wAxisEntity.hpp:
```



This graph shows which files directly or indirectly include this file:



#### **Classes**

· class wPlot2D::AxisEntity

Represents a visual axis (X or Y) in a 2D plot with optional notches and title.

### **Namespaces**

namespace wPlot2D

#### **Enumerations**

- enum class wPlot2D::AxisType { wPlot2D::X\_AXIS , wPlot2D::Y\_AXIS }
   Enum representing the type of axis to render.
- enum class wPlot2D::NotchPosition { wPlot2D::Center , wPlot2D::Above , wPlot2D::Below } Enum controlling the visual placement of notches relative to the axis.

# 8.49 wAxisEntity.hpp

#### Go to the documentation of this file.

00022 #include <SFML/Graphics.hpp>

00021 #pragma GCC diagnostic ignored "-Wswitch-default"

```
00001 /*
00002
00003 Created by Wilfried Koch.
00004 Copyright @ 2025 Wilfried Koch. All rights reserved.
00005
00006 */
00007
00008 #ifndef W_AXIS_ENTITY_HPP
00009 #define W_AXIS_ENTITY_HPP
00010
00011 #include "../srcEntity/wEntity.hpp" 00012 #include "../srcUtils/wAssetManager.hpp"
00013
00014 #include "wLineEntity.hpp"
00015 #include "wNotchEntity.hpp"
00016 #include "wTitleEntity.hpp"
00017 #include "wLabelEntity.hpp"
00018
00019 #pragma GCC diagnostic push
00020 #pragma GCC diagnostic ignored "-Wfloat-equal"
```

8.49 wAxisEntity.hpp 187

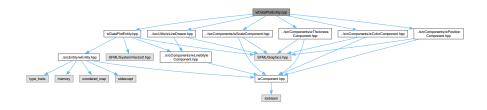
```
00023 #pragma GCC diagnostic pop
00024
00025 #include <vector>
00026
00027 namespace wPlot2D
00028 {
00034
          enum class AxisType
00035
00036
              X AXIS,
00037
              Y_AXIS
00038
         };
00039
00044
          enum class NotchPosition
00045
              Center,
00046
              Above
00047
00048
              Below
00049
          };
00050
00066
          class AxisEntity : public wEngine::Entity
00067
00068
              public:
                  AxisEntity( sf::Font& font, sf::Vector2f origin, sf::Vector2f scale, sf::Vector2f offset,
00078
     AxisType type, sf::Vector2f axisRange );
00079
00083
                  virtual ~AxisEntity() = default;
00084
00090
                  void setColor( sf::Color color );
00091
00098
                  void setThickness( float thickness);
00099
00104
                  void setArrowSize( float arrowSize );
00105
00110
                  void addTitle( const std::string& title );
00111
00116
                  void addTitle( const std::wstring& title );
00117
00122
                  void setTitleFont( const sf::Font& font );
00123
00128
                  void setTitleCharacterSize( unsigned int size );
00129
00134
                  void setTitleColor( sf::Color newColor );
00135
00140
                  void setTitleOffset( sf::Vector2f titleOffset );
00141
00146
                  [[nodiscard]] sf::Vector2f getTitleOffset() const;
00147
00154
                  void addNotches( float interval, NotchPosition position, bool hasLabels = false );
00155
00160
                  void setNotchesColor( const sf::Color& color );
00161
00166
                  void setNotchesThickness( float thickness);
00167
00172
                  void setNotchesLength( float newLength );
00173
00178
                  void setLabelsFont( const sf::Font& font );
00179
00184
                  void setLabelsColor( const sf::Color& color );
00185
00190
                  [[nodiscard]] std::vector< sf::Vector2f > getLabelsOffset() const;
00191
00196
                  void setLabelsOffset( sf::Vector2f offset );
00197
00202
                  void addLabelsOffset( sf::Vector2f delta );
00203
00208
                  void setLabelsCharacterSize( unsigned int newSize );
00209
00214
                  void setLabelsDecimalPlaces( int places );
00215
00220
                  void setCustomLabels( const std::vector< std::string >& labels );
00221
00226
                  void render( sf::RenderWindow& window );
00227
              private:
00228
                  sf::Font& mTitleFont;
00229
                  sf::Font& mLabelsFont;
00230
                  AxisType mAxisType;
00231
                  sf::Vector2f mAxisRange;
00232
                  std::unique_ptr< LineEntity > mAxisLine;
00233
                  float mArrowSize;
00234
00235
                  std::vector< std::unique_ptr< NotchEntity > > mNotches;
00236
                  NotchPosition mNotchPosition;
00237
00238
                  std::unique_ptr< TitleEntity > mTitle;
00239
00240
                  bool mHasLabels:
```

```
std::vector< std::unique_ptr< LabelEntity > > mLabels;
00242
00246
                   void construct();
00247
                   template < typename T >
void initTitle( const T& title );
00251
00252
00257
                   void initNotches();
00258
          };
00259
00260 }//End of namespace wPlot2D
00261
00262 #endif
```

# 8.50 wDataPlotEntity.cpp File Reference

Implementation of the DataPlotEntity class.

```
#include "wDataPlotEntity.hpp"
#include "../srcUtils/wLineDrawer.hpp"
#include "../srcComponents/wColorComponent.hpp"
#include "../srcComponents/wPositionComponent.hpp"
#include "../srcComponents/wScaleComponent.hpp"
#include "../srcComponents/wThicknessComponent.hpp"
Include dependency graph for wDataPlotEntity.cpp:
```



### **Namespaces**

namespace wPlot2D

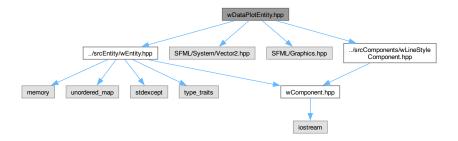
### 8.50.1 Detailed Description

Implementation of the DataPlotEntity class.

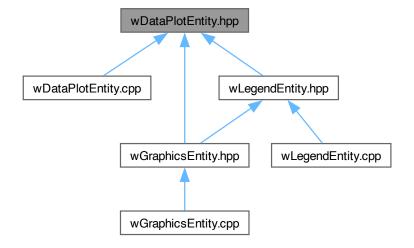
# 8.51 wDataPlotEntity.hpp File Reference

```
#include "../srcEntity/wEntity.hpp"
#include <SFML/System/Vector2.hpp>
#include <SFML/Graphics.hpp>
```

#include "../srcComponents/wLineStyleComponent.hpp"
Include dependency graph for wDataPlotEntity.hpp:



This graph shows which files directly or indirectly include this file:



### Classes

class wPlot2D::DataPlotEntity

Entity for plotting raw data points as a connected polyline.

### **Namespaces**

• namespace wPlot2D

### 8.52 wDataPlotEntity.hpp

#### Go to the documentation of this file.

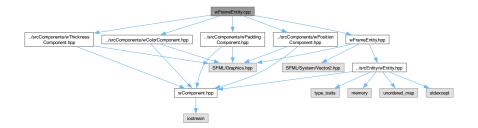
```
00002
00003 Created by Wilfried Koch.
00004 Copyright @ 2025 Wilfried Koch. All rights reserved.
00006 */
00007
00008 #ifndef W_DATA_PLOT_ENTITY_HPP
00009 #define W_DATA_PLOT_ENTITY_HPP
00010
00011 #include "../srcEntity/wEntity.hpp"
00013 #pragma GCC diagnostic push
00014 #pragma GCC diagnostic ignored "-Wfloat-equal"
00015 #pragma GCC diagnostic ignored "-Wswitch-default"
00016 #include <SFML/System/Vector2.hpp>
00017 #include <SFML/Graphics.hpp>
00018 #pragma GCC diagnostic pop
00019
00020 #include "../srcComponents/wLineStyleComponent.hpp"
00021
00022 namespace wPlot2D
00023 {
00024
00041
          class DataPlotEntity : public wEngine::Entity
00042
              public:
00043
                  DataPlotEntity( const sf::Vector2f origin, const sf::Vector2f scale, const std::vector
00051
     sf::Vector2f >& dataPoints );
00052
00056
                  virtual ~DataPlotEntity() = default;
00057
00062
                  [[nodiscard]] sf::Color getColor() const;
00063
00064
00069
                  [[nodiscard]] float getThickness();
00070
00075
                  [[nodiscard]] wEngine::LineStyleComponent::LineStyle getLineStyle();
00076
00077
00082
                  [[nodiscard]] float getDashLength();
00083
00088
                  [[nodiscard]] float getGapLength();
00089
00094
                  void setColor( sf::Color color );
00095
00100
                  void setThickness( float thickness);
00101
00106
                  void setLineStyle( wEngine::LineStyleComponent::LineStyle style );
00107
00112
                  void setDashLength( float dashLength);
00113
00118
                  void setGapLength( float gapLength );
00119
00128
                  void drawDataPlot( sf::RenderWindow &window );
00129
00130
             private:
00131
                  std::vector< sf::Vector2f > mDataPoints;
00132
         };
00133
00134 }//End of namespace wPlot2D
00135
00136 #endif
```

# 8.53 wFrameEntity.cpp File Reference

Implementation of the FrameEntity class.

```
#include "wFrameEntity.hpp"
#include "../srcComponents/wPositionComponent.hpp"
#include "../srcComponents/wThicknessComponent.hpp"
```

#include "../srcComponents/wColorComponent.hpp"
#include "../srcComponents/wPaddingComponent.hpp"
Include dependency graph for wFrameEntity.cpp:



### **Namespaces**

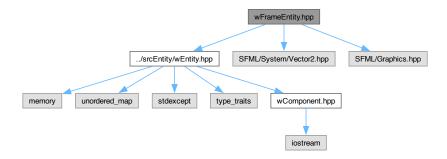
namespace wPlot2D

### 8.53.1 Detailed Description

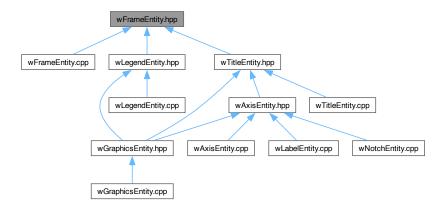
Implementation of the FrameEntity class.

# 8.54 wFrameEntity.hpp File Reference

```
#include "../srcEntity/wEntity.hpp"
#include <SFML/System/Vector2.hpp>
#include <SFML/Graphics.hpp>
Include dependency graph for wFrameEntity.hpp:
```



This graph shows which files directly or indirectly include this file:



#### Classes

· class wPlot2D::FrameEntity

Entity representing a rectangular frame around content.

### **Namespaces**

namespace wPlot2D

# 8.55 wFrameEntity.hpp

Go to the documentation of this file.

```
00001 /*
00002
00003 Created by Wilfried Koch.
00004 Copyright @ 2025 Wilfried Koch. All rights reserved.
00005
00006 */
00007
00008 #ifndef W_FRAME_ENTITY_HPP 00009 #define W_FRAME_ENTITY_HPP
00010
00011 #include "../srcEntity/wEntity.hpp"
00012
00013 #pragma GCC diagnostic push
00014 #pragma GCC diagnostic ignored "-Wfloat-equal"
00015 #pragma GCC diagnostic ignored "-Wswitch-default"
00016 #include <SFML/System/Vector2.hpp>
00017 #include <SFML/Graphics.hpp>
00018 #pragma GCC diagnostic pop
00019
00020 namespace wPlot2D
00021 {
00022
00023
00042
             class FrameEntity : public wEngine::Entity
00043
00044
                  public:
00049
                       FrameEntity( bool enabled = true );
00050
00054
                       virtual ~FrameEntity() = default;
00055
```

```
00060
                  void setEnabled( bool enabled );
00061
00066
                  [[nodiscard]] bool isEnabled() const;
00067
00072
                  [[nodiscard]] sf::Color getFillColor() const;
00073
                  [[nodiscard]] sf::Color getOutlineColor() const;
00079
00084
                  [[nodiscard]] float getThickness() const;
00085
                  [[nodiscard]] sf::Vector2f getPadding() const;
00090
00091
00096
                  void setFillColor( const sf::Color& color );
00097
00102
                  void setOutlineColor( const sf::Color& color );
00103
                  void setThickness( float thickness);
00108
00109
00114
                  void setPadding( const sf::Vector2f& padding );
00115
00121
                  void update( const sf::FloatRect& contentBounds, const sf::Vector2f& position );
00122
00127
                  void render( sf::RenderWindow& window );
00128
              private:
00129
                  bool mEnabled;
00130
                  sf::RectangleShape mFrame;
00131
00132
00133 }//End of namespace wPlot2D
00134
00135 #endif
```

## 8.56 wFunctionEntity.cpp File Reference

Implementation of the FunctionEntity class.

```
#include "wFunctionEntity.hpp"
#include "../srcUtils/wLineDrawer.hpp"
#include "../srcUtils/wMathUtils.hpp"
#include "../srcComponents/wColorComponent.hpp"
#include "../srcComponents/wPositionComponent.hpp"
#include "../srcComponents/wScaleComponent.hpp"
#include "../srcComponents/wOffsetComponent.hpp"
#include "../srcComponents/wThicknessComponent.hpp"
#include "../srcComponents/wFunctionComponent.hpp"
#include "../srcComponents/wDiscontinuityComponent.hpp"
#include "../srcComponents/wRotationComponent.hpp"
#include "../srcComponents/wRotationComponent.hpp"
#include "../srcComponents/wRotationComponent.hpp"
```

Include dependency graph for wFunctionEntity.cpp:



#### **Namespaces**

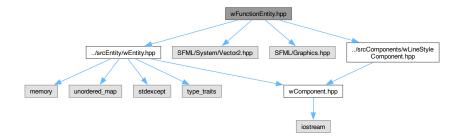
namespace wPlot2D

### 8.56.1 Detailed Description

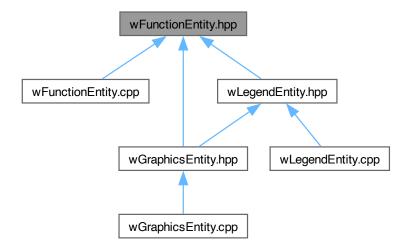
Implementation of the FunctionEntity class.

# 8.57 wFunctionEntity.hpp File Reference

```
#include "../srcEntity/wEntity.hpp"
#include <SFML/System/Vector2.hpp>
#include <SFML/Graphics.hpp>
#include "../srcComponents/wLineStyleComponent.hpp"
Include dependency graph for wFunctionEntity.hpp:
```



This graph shows which files directly or indirectly include this file:



#### Classes

· class wPlot2D::FunctionEntity

Represents a mathematical function as a drawable entity in a 2D plot.

### **Namespaces**

namespace wPlot2D

## 8.58 wFunctionEntity.hpp

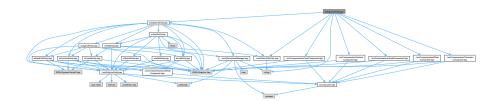
#### Go to the documentation of this file.

```
00001 /*
00002
00003 Created by Wilfried Koch.
00004 Copyright @ 2025 Wilfried Koch. All rights reserved.
00005
00006 */
00007
00008 #ifndef W_FUNCTION_ENTITY_HPP
00009 #define W_FUNCTION_ENTITY_HPP
00010
00011 #include "../srcEntity/wEntity.hpp"
00012
00013 #pragma GCC diagnostic push
00014 #pragma GCC diagnostic ignored "-Wfloat-equal" 00015 #pragma GCC diagnostic ignored "-Wswitch-default"
00016 #include <SFML/System/Vector2.hpp>
00017 #include <SFML/Graphics.hpp>
00018 #pragma GCC diagnostic pop
00019
00020 #include "../srcComponents/wLineStyleComponent.hpp"
00021
00022 namespace wPlot2D
00023 {
00024
00042
          class FunctionEntity : public wEngine::Entity
00043
00044
              public:
00051
                  FunctionEntity( const sf::Vector2f origin, const sf::Vector2f scale, std::function<
      double ( double ) > func );
00052
00056
                  virtual ~FunctionEntity() = default;
00057
00062
                   [[nodiscard]] sf::Vector2f getPosition() const;
00063
00068
                   [[nodiscard]] sf::Color getColor() const;
00069
00074
                   [[nodiscard]] float getThickness() const;
00075
00080
                   [[nodiscard]] wEngine::LineStyleComponent::LineStyle getLineStyle() const;
00081
00087
                   [[nodiscard]] float getDashLength() const;
00088
00094
                   [[nodiscard]] float getGapLength() const;
00095
00100
                   [[nodiscard]] sf::Vector2f getOffset() const;
00101
00106
                   [[nodiscard]] float getRotation() const;
00107
00112
                   void setPosition( sf::Vector2f position );
00113
00118
                  void setColor( sf::Color color );
00119
00124
                  void setThickness( float thickness);
00125
00130
                  void setLineStyle( wEngine::LineStyleComponent::LineStyle style );
00131
00138
                  void setDashLength( float dashLength);
00139
00146
                  void setGapLength( float gapLength);
00147
00158
                  void setOffset( float offsetX, float offsetY );
00159
00169
                  void setRotation( float angleDegrees );
00170
00179
                  void setScale( sf::Vector2f scale );
00180
00189
                  void addExcludedInterval( double min, double max );
00190
00194
                  void clearExcludedIntervals();
00195
                  void alignToYAxis( float normalizedOffsetX = 0.0f, float normalizedOffsetY = 0.0f);
00208
00209
00217
                   void drawFunction( sf::RenderWindow &window, double startX, double endX, size_t nbPoints =
      1000 );
              private:
00218
00219
00220
00221 }//End of namespace wPlot2D
00222
00223 #endif
```

## 8.59 wGraphicsEntity.cpp File Reference

Implementation of the GraphicsEntity class.

```
#include "wGraphicsEntity.hpp"
#include <string>
#include "../srcUtils/wPathUtils.hpp"
#include "../srcComponents/wColorComponent.hpp"
#include "../srcComponents/wPositionComponent.hpp"
#include "../srcComponents/wScaleComponent.hpp"
#include "../srcComponents/wOffsetComponent.hpp"
#include "../srcComponents/wThicknessComponent.hpp"
#include dependency graph for wGraphicsEntity.cpp:
```



#### **Namespaces**

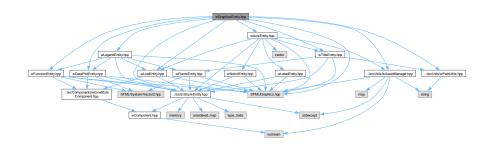
namespace wPlot2D

### 8.59.1 Detailed Description

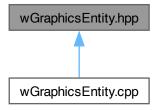
Implementation of the GraphicsEntity class.

## 8.60 wGraphicsEntity.hpp File Reference

```
#include "../srcEntity/wEntity.hpp"
#include "../srcUtils/wAssetManager.hpp"
#include "../srcUtils/wPathUtils.hpp"
#include "wAxisEntity.hpp"
#include "wTitleEntity.hpp"
#include "wFunctionEntity.hpp"
#include "wDataPlotEntity.hpp"
#include "wLegendEntity.hpp"
#include "wLineEntity.hpp"
#include <SFML/Graphics.hpp>
Include dependency graph for wGraphicsEntity.hpp:
```



This graph shows which files directly or indirectly include this file:



#### Classes

class wPlot2D::GraphicsEntity
 Central entity responsible for graphical rendering in wPlot2D.

#### **Namespaces**

namespace wPlot2D

### **Enumerations**

enum class wPlot2D::TitleAlignment { wPlot2D::Top , wPlot2D::Bottom }
 Defines the vertical placement of the main plot title.

# 8.61 wGraphicsEntity.hpp

#### Go to the documentation of this file.

```
00001 /*
00002
00003 Created by Wilfried Koch.
00004 Copyright @ 2025 Wilfried Koch. All rights reserved.
00005
00006 */
00007
00008 #ifndef W_GRAPHICS_ENTITY_HPP 00009 #define W_GRAPHICS_ENTITY_HPP
00010
00011 #include "../srcEntity/wEntity.hpp"
00012
00013 #include "../srcUtils/wAssetManager.hpp" 00014 #include "../srcUtils/wPathUtils.hpp"
00015
00016 #include "wAxisEntity.hpp"
00017 #include "wTitleEntity.hpp"
00018 #include "wFunctionEntity.hpp"
00019 #include "wDataPlotEntity.hpp"
00020 #include "wLegendEntity.hpp"
00021 #include "wLineEntity.hpp"
00022
00023 #pragma GCC diagnostic push
00024 #pragma GCC diagnostic ignored "-Wfloat-equal"
```

```
00025 #pragma GCC diagnostic ignored "-Wswitch-default"
00026 #include <SFML/Graphics.hpp>
00027 #pragma GCC diagnostic pop
00028
00029 namespace wPlot2D
00030 {
00036
          enum class TitleAlignment
00037
00038
              Top,
00039
              Bottom
00040
         };
00041
00064
          class GraphicsEntity : public wEngine::Entity
00065
00066
              public:
                  GraphicsEntity(
00077
00078
                      const std::string& windowTitle = "wPlot2D",
                      const sf::Vector2u& windowSize = { 1600, 1600 },
00079
00080
                      const sf::Vector2f& originFactor = { 0.5f, 0.5f },
00081
                      const sf::Vector2f& scaleFactor = { 0.1f, 0.1f } );
00082
00086
                  virtual ~GraphicsEntity() = default;
00087
00092
                  [[nodiscard]] sf::RenderWindow& getWindow();
00093
00098
                  [[nodiscard]] sf::Vector2u getWindowSize() const;
00099
00104
                  void setWindowSize( const sf::Vector2u& newSize );
00105
00110
                  void setWindowTitle( const std::string& title );
00111
00116
                  void setBackgroundColor( const sf::Color& color );
00117
00123
                  void addFont( const std::string& name, const std::string& fileName );
00124
00131
                  sf::Font& getFont ( const std::string name );
00132
00138
                  [[nodiscard]] sf::Vector2f getOrigin() const;
00139
00145
                  void setOrigin( sf::Vector2f originFactor );
00146
00152
                  [[nodiscard]] sf::Vector2f getScale() const;
00153
00158
                  void setScale( sf::Vector2f scaleFactor );
00159
00165
                  [[nodiscard]] sf::Vector2f getOffset() const;
00166
                  void setOffset( sf::Vector2f offset );
00171
00172
00179
                  [[nodiscard]] AxisEntity* addAxis( AxisType type, sf::Vector2f axisRange );
00180
00187
                  [[nodiscard]] TitleEntity* addTitle( const std::string& title, TitleAlignment alignment =
     TitleAlignment::Bottom );
00188
00195
                  [[nodiscard]] TitleEntity* addTitle( const std::wstring& title, TitleAlignment alignment =
      TitleAlignment::Bottom );
00196
00205
                  [[nodiscard]] FunctionEntity* addFunction( std::function< double( double )> func,
00206
                      double startX, double endX, size_t nbPoints = 1000 );
00207
00213
                  [[nodiscard]] DataPlotEntity* addDataPlot( const std::vector< sf::Vector2f >& dataPoints
     );
00214
00221
                  [[nodiscard]] LegendEntity* addLegend( const sf::Vector2f& position, bool hasFrame = true
00222
00229
                  [[nodiscard]] TitleEntity* addText( const std::string& text, sf::Vector2f position );
00230
00237
                  [[nodiscard]] TitleEntity* addText( const std::wstring& text, sf::Vector2f position );
00238
00246
                  [[nodiscard]] LineEntity* addLine( const sf::Vector2f& start, const sf::Vector2f& end,
     bool withArrow = false );
00247
00253
                  void saveToFile( const std::string& filename );
00254
              private:
00255
                  sf::RenderWindow mWindow;
00256
                  wEngine::AssetManager mAssets;
00257
                  std::unique_ptr< AxisEntity > mAxisX;
                  std::unique_ptr< AxisEntity > mAxisY;
00258
                  std::unique_ptr< TitleEntity > mTitle;
00259
00260
                  TitleAlignment mAlignment;
00261
00262
                  template < typename T >
00263
                  TitleEntity* addTitleImpl( const T& title, TitleAlignment alignment );
00264
00265
                  template < typename T >
```

```
00266
                  TitleEntity* initText( const T& title, sf::Vector2f position );
00267
00268
                  struct FunctionData
00269
                      std::unique_ptr< FunctionEntity > entity;
00270
00271
                      double startX:
00272
                      double endX;
00273
                      size_t nbPoints;
00274
00275
                  std::vector< FunctionData > mFunctions;
00276
00277
                  struct DataPlotData
00278
                  {
00279
                       std::unique_ptr< DataPlotEntity > entity;
00280
00281
                  std::vector< DataPlotData > mDataPlots;
00282
00283
                  std::unique_ptr< LegendEntity > mLegend;
00284
00285
                  struct TextData
00286
00287
                       std::unique_ptr< TitleEntity > entity;
00288
                  std::vector< TextData > mTexts:
00289
00290
00291
                  struct LineData
00292
00293
                       std::unique_ptr< LineEntity > entity;
00294
00295
                  std::vector< LineData > mLines;
00296
00313
                  void render();
00314
00325
                  void validateNormalizedFactor( const sf::Vector2f& factor ) const;
00326
                  [[nodiscard]] sf::Vector2f convertNormalizedToPixels( const sf::Vector2f& factor ) const;
00338
00339
          };
00340
00341 }//End of namespace wPlot2D
00342
00343 #endif
```

# 8.62 wLabelEntity.cpp File Reference

Implementation of the LabelEntity class.

```
#include "wLabelEntity.hpp"
#include "wAxisEntity.hpp"
#include "../srcComponents/wPositionComponent.hpp"
#include "../srcComponents/wOffsetComponent.hpp"
#include "../srcComponents/wColorComponent.hpp"
#include "../srcComponents/wFontComponent.hpp"
Include dependency graph for wLabelEntity.cpp:
```



#### **Namespaces**

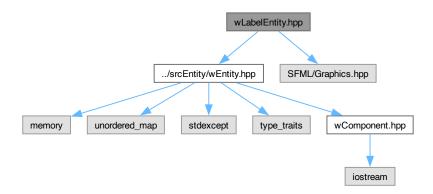
namespace wPlot2D

### 8.62.1 Detailed Description

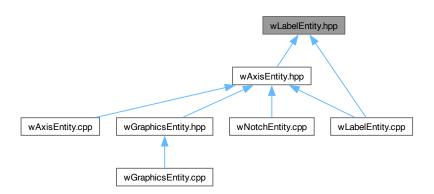
Implementation of the LabelEntity class.

# 8.63 wLabelEntity.hpp File Reference

#include "../srcEntity/wEntity.hpp"
#include <SFML/Graphics.hpp>
Include dependency graph for wLabelEntity.hpp:



This graph shows which files directly or indirectly include this file:



#### **Classes**

class wPlot2D::LabelEntity

Represents a textual label or a collection of axis labels.

8.64 wLabelEntity.hpp 201

#### **Namespaces**

• namespace wPlot2D

### 8.64 wLabelEntity.hpp

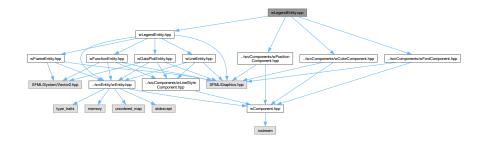
#### Go to the documentation of this file.

```
00001 /*
00002
00003 Created by Wilfried Koch.
00004 Copyright @ 2025 Wilfried Koch. All rights reserved.
00006 */
00007
00008 #ifndef W_LABEL_ENTITY_HPP
00009 #define W_LABEL_ENTITY_HPP
00011 #include "../srcEntity/wEntity.hpp"
00012
00013 #pragma GCC diagnostic push
00014 #pragma GCC diagnostic ignored "-Wfloat-equal"
00015 #pragma GCC diagnostic ignored "-Wswitch-default"
00016 #include <SFML/Graphics.hpp>
00017 #pragma GCC diagnostic pop
00018
00019 namespace wPlot2D
00020 {
00022
          enum class AxisType;
00023
00043
          class LabelEntity : public wEngine::Entity
00044
              public:
00045
00052
                  LabelEntity ( const sf::Font& font, AxisType type, sf::Vector2f initialPosition );
00053
00057
                  virtual ~LabelEntity() = default;
00058
                  [[nodiscard]] float getValue() const;
00063
00064
00069
                  [[nodiscard]] unsigned int getCharacterSize() const;
00075
                  [[nodiscard]] int getDecimalPlaces() const;
00076
00081
                  void setFont( const sf::Font& font );
00082
00091
                  void setLabelText( std::string text );
00092
00097
                  void setCharacterSize( unsigned int newSize );
00098
00103
                  void setDecimalPlaces( int places );
00104
00113
                  void setCustomLabels( const std::string& labels );
00114
00119
                  [[nodiscard]] bool usesCustomLabels() const;
00120
00130
                  std::string formatLabel( float value );
00131
00136
                  void render( sf::RenderWindow& window );
00137
             private:
00138
                 AxisType mAlignment;
00139
                  unsigned int mCharacterSize;
00140
                  float mValue;
00141
                  int mDecimalPlaces;
00142
                 sf::Vector2f mOffset;
00143
00144
                 std::string mCustomLabels;
00145
                  bool mUseCustomLabels;
00146
00147
                  sf::Text mLabel;
00148
         };
00149
00150 } // namespace wPlot2D
00151
00152 #endif
```

### 8.65 wLegendEntity.cpp File Reference

Implementation of the LegendEntity class.

```
#include "wLegendEntity.hpp"
#include "../srcComponents/wPositionComponent.hpp"
#include "../srcComponents/wColorComponent.hpp"
#include "../srcComponents/wFontComponent.hpp"
Include dependency graph for wLegendEntity.cpp:
```



#### **Namespaces**

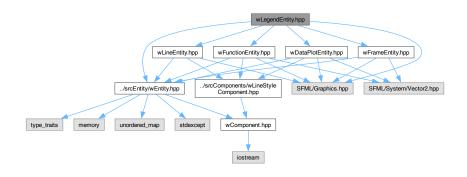
namespace wPlot2D

#### 8.65.1 Detailed Description

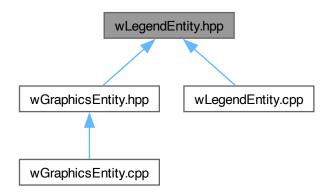
Implementation of the LegendEntity class.

### 8.66 wLegendEntity.hpp File Reference

```
#include "../srcEntity/wEntity.hpp"
#include "wLineEntity.hpp"
#include "wFrameEntity.hpp"
#include "wFunctionEntity.hpp"
#include "wDataPlotEntity.hpp"
#include <SFML/Graphics.hpp>
Include dependency graph for wLegendEntity.hpp:
```



This graph shows which files directly or indirectly include this file:



#### **Classes**

· class wPlot2D::LegendEntity

Represents a legend box that describes functions and data plots.

#### **Namespaces**

namespace wPlot2D

# 8.67 wLegendEntity.hpp

#### Go to the documentation of this file.

```
00001 /*
00002
00003 Created by Wilfried Koch.
00004 Copyright @ 2025 Wilfried Koch. All rights reserved.
00005
00006 */
00007
00008 #ifndef W_LEGEND_ENTITY_HPP
00009 #define W_LEGEND_ENTITY_HPP
00010
00011 #include "../srcEntity/wEntity.hpp"
00012 #include "wLineEntity.hpp"
00013 #include "wFrameEntity.hpp"
00014
00015 #include "wFunctionEntity.hpp"
00016 #include "wDataPlotEntity.hpp"
00017
00018 #pragma GCC diagnostic push
00010 #pragma GCC diagnostic ignored "-Wfloat-equal"
00020 #pragma GCC diagnostic ignored "-Wswitch-default"
00021 #include <SFML/Graphics.hpp>
00022 #pragma GCC diagnostic pop
00023
00024 namespace wPlot2D
00025 {
00026
00027
```

```
class LegendEntity : public wEngine::Entity
00056
              public:
00057
00067
                  LegendEntity( const sf::Font& font, const sf::Vector2f& position, bool hasFrame = true );
00068
00072
                  virtual ~LegendEntity() = default;
00079
                  void addItem( const std::string& label, FunctionEntity* function );
08000
00086
                  void addItem( const std::wstring& label, FunctionEntity* function );
00087
                  void addItem( const std::string& label, DataPlotEntity* plot );
00093
00094
00100
                  void addItem( const std::wstring& label, DataPlotEntity* plot );
00101
00106
                  void setFrameEnabled( bool enabled );
00107
                  void setFrameFillColor( const sf::Color& color );
00112
00113
00118
                  void setFrameOutlineColor( const sf::Color& color );
00119
00124
                  void setFrameThickness( float thickness );
00125
00135
                  void setPadding( const sf::Vector2f& padding );
00136
                  void setFont( const sf::Font& font );
00144
00149
                  void setCharacterSize( unsigned int size );
00150
00155
                  void setTextColor( const sf::Color& color );
00156
00165
                  void render( sf::RenderWindow& window );
00166
              private:
00171
                  struct LegendItem
00172
00173
                      std::string label;
00174
                      std::unique_ptr< LineEntity > line;
00175
                      sf::Text labelText;
00176
00177
                      LegendItem( std::unique_ptr< LineEntity > line, sf::Text&& txt )
00178
                      : line{ std::move( line ) },
00179
                          labelText{ std::move( txt ) }
00180
00181
00182
00183
00184
00197
                  template < typename LabelT, typename SourceT >
00198
                  void addItemGeneric( const LabelT& label, SourceT* source );
00199
00211
                  template< typename T >
00212
                  void createItem( const T& label, std::unique_ptr< LineEntity > line );
00213
00214
                  std::vector< LegendItem > mItems;
00215
                  const sf::Font& mFont;
00216
                  unsigned int mCharacterSize = 30;
                  FrameEntity mFrame;
00218
          };
00219
00220 } // namespace wPlot2D
00221
00222 #endif
```

# 8.68 wLineEntity.cpp File Reference

Implementation of the LineEntity class.

```
#include "wLineEntity.hpp"
#include "../srcComponents/wPositionComponent.hpp"
#include "../srcComponents/wScaleComponent.hpp"
#include "../srcComponents/wColorComponent.hpp"
#include "../srcComponents/wThicknessComponent.hpp"
#include "../srcComponents/wLineStyleComponent.hpp"
```

#include "../srcUtils/wLineDrawer.hpp"
Include dependency graph for wLineEntity.cpp:



#### **Namespaces**

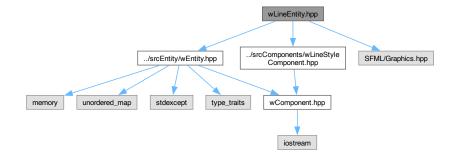
namespace wPlot2D

### 8.68.1 Detailed Description

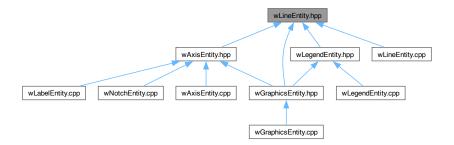
Implementation of the LineEntity class.

### 8.69 wLineEntity.hpp File Reference

```
#include "../srcEntity/wEntity.hpp"
#include "../srcComponents/wLineStyleComponent.hpp"
#include <SFML/Graphics.hpp>
Include dependency graph for wLineEntity.hpp:
```



This graph shows which files directly or indirectly include this file:



#### Classes

· class wPlot2D::LineEntity

Entity representing a straight line segment with optional arrowhead.

#### **Namespaces**

• namespace wPlot2D

# 8.70 wLineEntity.hpp

Go to the documentation of this file.

```
00001 /*
00002
00003 Created by Wilfried Koch.
00004 Copyright @ 2025 Wilfried Koch. All rights reserved.
00005
00006 */
00007
00008 #ifndef W_LINE_ENTITY_HPP
00009 #define W_LINE_ENTITY_HPP
00010
00011 #include "../srcEntity/wEntity.hpp"
00012 #include "../srcComponents/wLineStyleComponent.hpp"
00013
00014 #pragma GCC diagnostic push
00015 #pragma GCC diagnostic ignored "-Wfloat-equal"
00016 #pragma GCC diagnostic ignored "-Wswitch-default"
00017 #include <SFML/Graphics.hpp>
00018 #pragma GCC diagnostic pop
00019
00020 namespace wPlot2D
00021 {
00022
00042
          class LineEntity : public wEngine::Entity
00043
00044
              public:
                 LineEntity( const sf::Vector2f& origin, const sf::Vector2f& scale, const sf::Vector2f&
00054
      start, const sf::Vector2f& end,
00055
                     bool withArrow = false );
00056
00060
                  virtual ~LineEntity() = default;
00061
00066
                  void setColor( sf::Color color );
00067
00072
                  void setThickness( float thickness );
00073
00078
                  [[nodiscard]] float getThickness() const;
```

```
00079
00084
                  void setLineStyle( wEngine::LineStyleComponent::LineStyle style );
00085
00090
                  void setDashLength( float dashLength);
00091
00096
                  void setGapLength( float gapLength );
00097
00102
                  [[nodiscard]] sf::Vector2f getStartPoint() const;
00103
00108
                  [[nodiscard]] sf::Vector2f getEndPoint() const;
00109
00114
                  [[nodiscard]] bool hasArrow() const;
00115
00120
                  [[nodiscard]] float getArrowSize() const;
00121
00126
                  void setArrowSize( float arrowSize );
00127
00132
                  void render( sf::RenderWindow& window );
00133
00134
              private:
00135
                  sf::Vector2f mStart;
00136
                  sf::Vector2f mEnd;
00137
                  bool mWithArrow;
00138
                  sf::ConvexShape mArrowHead;
00139
                  float mArrowSize;
00140
00152
                  void initArrowHead( const sf::Vector2f& lineEnd, const sf::Vector2f& dir, float arrowSize,
      sf::Color color );
00153
00154
00155 }//End of namespace wPlot2D
00156
00157 #endif
```

### 8.71 wNotchEntity.cpp File Reference

Implementation of the NotchEntity class.

```
#include "wNotchEntity.hpp"
#include "wAxisEntity.hpp"
#include "../srcComponents/wPositionComponent.hpp"
#include "../srcComponents/wColorComponent.hpp"
#include "../srcComponents/wThicknessComponent.hpp"
#include "../srcComponents/wLengthComponent.hpp"
Include dependency graph for wNotchEntity.cpp:
```



#### **Namespaces**

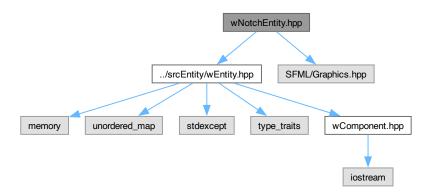
namespace wPlot2D

#### 8.71.1 Detailed Description

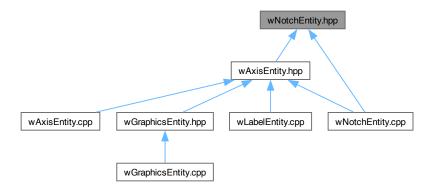
Implementation of the NotchEntity class.

# 8.72 wNotchEntity.hpp File Reference

#include "../srcEntity/wEntity.hpp"
#include <SFML/Graphics.hpp>
Include dependency graph for wNotchEntity.hpp:



This graph shows which files directly or indirectly include this file:



#### Classes

class wPlot2D::NotchEntity

Represents a single tick mark ("notch") on a 2D axis.

### Namespaces

namespace wPlot2D

8.73 wNotchEntity.hpp 209

### 8.73 wNotchEntity.hpp

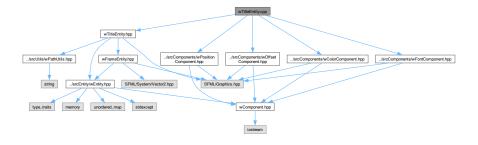
#### Go to the documentation of this file.

```
00001 /*
00002
00003 Created by Wilfried Koch.
00004 Copyright @ 2025 Wilfried Koch. All rights reserved.
00006 */
00007
00008 #ifndef W_NOTCH_ENTITY_HPP
00009 #define W NOTCH ENTITY HPP
00010
00011 #include "../srcEntity/wEntity.hpp"
00013 #pragma GCC diagnostic push
00014 #pragma GCC diagnostic ignored "-Wfloat-equal"
00015 #pragma GCC diagnostic ignored "-Wswitch-default"
00016 #include <SFML/Graphics.hpp>
00017 #pragma GCC diagnostic pop
00018
00019 namespace wPlot2D
00020 {
00021
00022
          enum class AxisType;
00023
00051
          class NotchEntity : public wEngine::Entity
00052
00053
              public:
00058
                  NotchEntity( AxisType type );
00059
00063
                  virtual ~NotchEntity() = default;
00064
00074
                  void render( sf::RenderWindow& window );
00075
00076
              private:
00077
                  AxisType mAlignment;
00078
         };
00080 } // namespace wPlot2D
00081
00082 #endif
```

# 8.74 wTitleEntity.cpp File Reference

Implementation of the TitleEntity class.

```
#include "wTitleEntity.hpp"
#include "../srcComponents/wPositionComponent.hpp"
#include "../srcComponents/wOffsetComponent.hpp"
#include "../srcComponents/wColorComponent.hpp"
#include "../srcComponents/wFontComponent.hpp"
Include dependency graph for wTitleEntity.cpp:
```



#### **Namespaces**

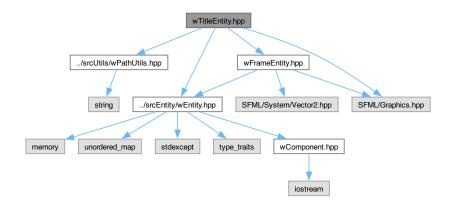
namespace wPlot2D

### 8.74.1 Detailed Description

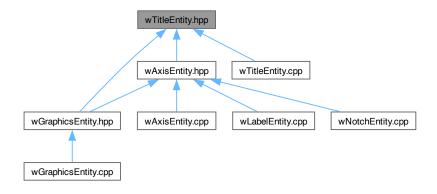
Implementation of the TitleEntity class.

### 8.75 wTitleEntity.hpp File Reference

```
#include "../srcEntity/wEntity.hpp"
#include "../srcUtils/wPathUtils.hpp"
#include "wFrameEntity.hpp"
#include <SFML/Graphics.hpp>
Include dependency graph for wTitleEntity.hpp:
```



This graph shows which files directly or indirectly include this file:



8.76 wTitleEntity.hpp 211

#### **Classes**

· class wPlot2D::TitleEntity

Represents a textual label (typically an axis title or main plot title) in a 2D plot.

#### **Namespaces**

namespace wPlot2D

### 8.76 wTitleEntity.hpp

Go to the documentation of this file.

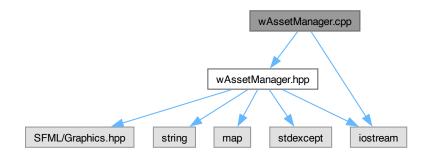
```
00001 /*
00002
00003 Created by Wilfried Koch.
00004 Copyright @ 2025 Wilfried Koch. All rights reserved.
00005
00006 */
00007
00008 #ifndef W_TITLE_ENTITY_HPP
00009 #define W_TITLE_ENTITY_HPP
00010
00011 #include "../srcEntity/wEntity.hpp"
00012 #include "../srcUtils/wPathUtils.hpp"
00013 #include "wFrameEntity.hpp"
00014
00015 #pragma GCC diagnostic push
00016 #pragma GCC diagnostic ignored "-Wfloat-equal"
00017 #pragma GCC diagnostic ignored "-Wswitch-default"
00018 #include <SFML/Graphics.hpp>
00019 #pragma GCC diagnostic pop
00020
00021 namespace wPlot2D
00022 {
00023
00044
          class TitleEntity : public wEngine::Entity
00045
00046
              public:
00053
                  TitleEntity( const sf::Font& font, const std::string& title, bool hasFrame = false );
00054
00061
                  TitleEntity( const sf::Font& font, const std::wstring& title, bool hasFrame = false );
00062
00066
                   virtual ~TitleEntity() = default;
00067
00072
                   [[nodiscard]] unsigned int getCharacterSize( ) const;
00073
00078
                   [[nodiscard]] sf::FloatRect getTextSize() const;
00079
00084
                   void setTextColor( sf::Color textColor );
00085
00090
                   void setOffset( sf::Vector2f offset );
00091
00096
                   void setCharacterSize( unsigned int size );
00097
00103
                   void setFont( const sf::Font& font );
00104
00109
                   [[nodiscard]] sf::Color getFrameOutlineColor() const;
00110
00115
                   [[nodiscard]] sf::Color getFrameFillColor() const;
00116
00121
                   [[nodiscard]] float getFrameThickness() const;
00122
00127
                   [[nodiscard]] sf::Vector2f getPadding() const;
00128
                   [[nodiscard]] bool isFrameEnabled() const;
00133
00134
00139
                   void setFrameEnabled( bool enabled );
00140
00145
                   void setFrameOutlineColor( const sf::Color& color );
00146
00151
                   void setFrameFillColor( const sf::Color& color );
00152
00157
                   void setFrameThickness( float thickness);
```

```
00167
                   void setPadding( sf::Vector2f padding );
00168
                   void render( sf::RenderWindow& window );
00182
00183
              private:
00184
00185
                  sf::Text mTitleText;
00186
                  FrameEntity mFrame;
00187
                  template < typename T >
void init( const T& title );
00193
00194
00195
          };
00196
00197 } // namespace wPlot2D
00198
00199 #endif
```

# 8.77 wAssetManager.cpp File Reference

Implementation of the AssetManager class.

```
#include "wAssetManager.hpp"
#include <iostream>
Include dependency graph for wAssetManager.cpp:
```



#### **Namespaces**

• namespace wEngine

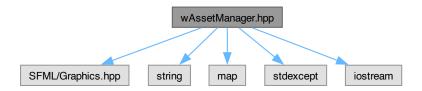
#### 8.77.1 Detailed Description

Implementation of the AssetManager class.

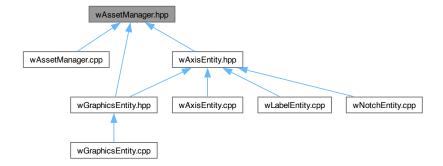
# 8.78 wAssetManager.hpp File Reference

```
#include <SFML/Graphics.hpp>
#include <string>
#include <map>
#include <stdexcept>
#include <iostream>
```

Include dependency graph for wAssetManager.hpp:



This graph shows which files directly or indirectly include this file:



#### Classes

• class wEngine::AssetManager

Manages graphical assets such as fonts for reuse across the application.

#### **Namespaces**

· namespace wEngine

### 8.79 wAssetManager.hpp

#### Go to the documentation of this file.

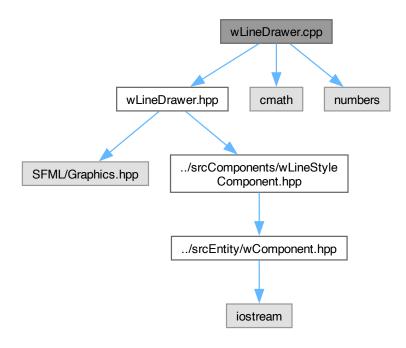
```
00002
00003 Created by Wilfried Koch.
00004 Copyright @ 2025 Wilfried Koch. All rights reserved.
00006 */
00007
00008 #ifndef W_ASSET_MANAGER_HPP
00009 #define W ASSET MANAGER HPP
00010
00011 #pragma GCC diagnostic push
00012 #pragma GCC diagnostic ignored "-Wfloat-equal"
00013 #pragma GCC diagnostic ignored "-Wswitch-default"
00014 #include <SFML/Graphics.hpp>
00015 #pragma GCC diagnostic pop
00016
00017 #include <string>
00018 #include <map>
00019 #include <stdexcept>
00020 #include <iostream>
00021
00022 namespace wEngine
00023 {
00024
00049
         class AssetManager
00050
00051
              public:
                 AssetManager() = default;
00052
00053
                  AssetManager( const AssetManager& ) = delete;
                 AssetManager& operator=( const AssetManager& ) = delete;
00055
                  ~AssetManager() = default;
00056
00068
                 void LoadFont( const std::string& name, const std::string& fileName );
00069
00076
                 sf::Font& getFont( const std::string& name );
00077
00083
                 void RemoveFont( const std::string& name );
00084
00088
                 void debugPrintFonts() const;
00089
             private:
00090
                 std::map< std::string, sf::Font > mFont;
00091
00105
                  template < typename Map >
00106
                  static void EnsureExists (const Map& map, const std::string& name, const std::string& type
);
00107
         };
00108
00109 }//End of namespace wEngine
00110
00111 #endif
```

# 8.80 wLineDrawer.cpp File Reference

Implementation of the LineDrawer class.

```
#include "wLineDrawer.hpp"
#include <cmath>
#include <numbers>
```

Include dependency graph for wLineDrawer.cpp:



#### **Namespaces**

• namespace wEngine

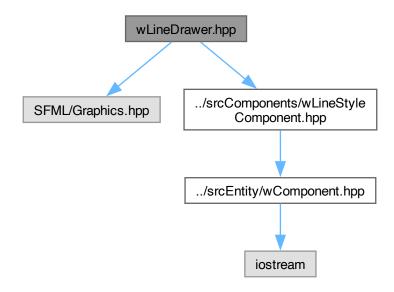
### 8.80.1 Detailed Description

Implementation of the LineDrawer class.

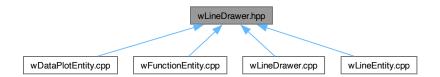
# 8.81 wLineDrawer.hpp File Reference

```
#include <SFML/Graphics.hpp>
#include "../srcComponents/wLineStyleComponent.hpp"
```

Include dependency graph for wLineDrawer.hpp:



This graph shows which files directly or indirectly include this file:



#### Classes

• class wEngine::LineDrawer

Utility class for rendering thick lines and polylines with style support.

#### **Namespaces**

• namespace wEngine

8.82 wLineDrawer.hpp 217

### 8.82 wLineDrawer.hpp

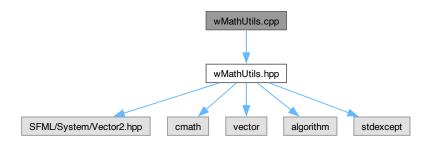
#### Go to the documentation of this file.

```
00002
00003 Created by Wilfried Koch.
00004 Copyright @ 2025 Wilfried Koch. All rights reserved.
00006 */
00007
00008 #ifndef W_LINE_DRAWER_HPP
00009 #define W LINE DRAWER HPP
00010
00011 #pragma GCC diagnostic push
00012 #pragma GCC diagnostic ignored "-Wfloat-equal"
00013 #pragma GCC diagnostic ignored "-Wswitch-default"
00014 #include <SFML/Graphics.hpp>
00015 #pragma GCC diagnostic pop
00016
00017 #include "../srcComponents/wLineStyleComponent.hpp"
00019 namespace wEngine
00020 {
00021
00051
          class LineDrawer
00052
              public:
00053
00054
00081
                   static float drawLine( sf::RenderWindow& window, const sf::Vector2f& point1, const
      sf::Vector2f& point2,
00082
                      const sf::Color& color, float thickness, LineStyleComponent::LineStyle style =
      LineStyleComponent::LineStyle::Solid,
00083
                      float dashLength = 20.0f, float gapLength = 5.0f, float patternOffset = 0.0f);
00084
00102
                  static void drawPolylineRound( sf::RenderWindow& window, const std::vector<sf::Vector2f>&
      points, const sf::Color& color,
00103
      float thickness, LineStyleComponent::LineStyle style =
LineStyleComponent::LineStyle::Solid, float dashLength = 20.0f,
00104
                      float gapLength = 5.0f, unsigned int arcResolution = 12 );
00105
00106
00107 }//End of namespace wEngine
00108
00109 #endif
```

# 8.83 wMathUtils.cpp File Reference

Implementation of the MathUtils class.

```
#include "wMathUtils.hpp"
Include dependency graph for wMathUtils.cpp:
```



#### **Namespaces**

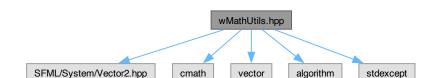
• namespace wEngine

### 8.83.1 Detailed Description

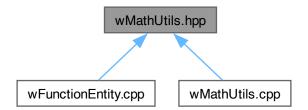
Implementation of the MathUtils class.

### 8.84 wMathUtils.hpp File Reference

```
#include <SFML/System/Vector2.hpp>
#include <cmath>
#include <vector>
#include <algorithm>
#include <stdexcept>
Include dependency graph for wMathUtils.hpp:
```



This graph shows which files directly or indirectly include this file:



#### Classes

• class wEngine::MathUtils

Provides common mathematical helper functions for plotting and geometry.

8.85 wMathUtils.hpp 219

#### **Namespaces**

• namespace wEngine

### 8.85 wMathUtils.hpp

Go to the documentation of this file.

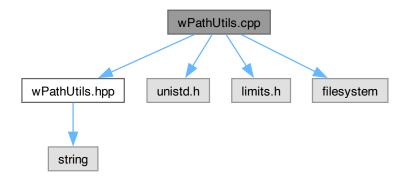
```
00001 /*
00002
00003 Created by Wilfried Koch.
00004 Copyright @ 2025 Wilfried Koch. All rights reserved.
00006 */
00007
00008 #ifndef W_MATH_UTILS_HPP
00009 #define W_MATH_UTILS_HPP
00011 #pragma GCC diagnostic push
00012 *pragma GCC diagnostic ignored "-Wfloat-equal" 00013 #include <SFML/System/Vector2.hpp>
00014 #pragma GCC diagnostic pop
00015
00016 #include <cmath>
00017 #include <vector>
00018 #include <algorithm>
00019 #include <stdexcept>
00020
00021 namespace wEngine
00022 {
00023
00037
          class MathUtils
00038
               public:
00039
                   [[nodiscard]] static std::vector< double > linspace( double start, double end, size_t
00057
      nbPoints );
00058
          };
00059
00060 }//End of namespace wEngine
00061
00062 #endif
```

# 8.86 wPathUtils.cpp File Reference

Implementation of the PathUtils class.

```
#include "wPathUtils.hpp"
#include <unistd.h>
#include <limits.h>
#include <filesystem>
```

Include dependency graph for wPathUtils.cpp:



#### Namespaces

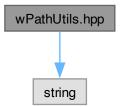
• namespace wEngine

### 8.86.1 Detailed Description

Implementation of the PathUtils class.

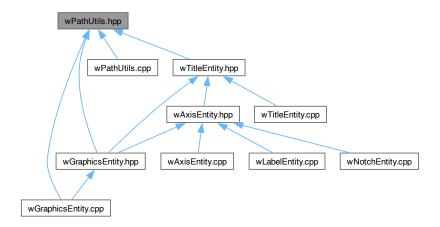
# 8.87 wPathUtils.hpp File Reference

#include <string>
Include dependency graph for wPathUtils.hpp:



8.88 wPathUtils.hpp 221

This graph shows which files directly or indirectly include this file:



#### Classes

· class wEngine::PathUtils

Utility class providing static functions for managing executable and resource paths across platforms.

#### **Namespaces**

namespace wEngine

# 8.88 wPathUtils.hpp

Go to the documentation of this file.

```
00001 /*
00002
00003 Created by Wilfried Koch.
00004 Copyright @ 2025 Wilfried Koch. All rights reserved.
00005
00006 */
00007
00008 #ifndef W_PATH_UTILS_HPP
00009 #define W_PATH_UTILS_HPP
00011 #include <string>
00012
00013 namespace wEngine
00014 {
00015
00044
           class PathUtils
00045
00046
                public:
                    [[nodiscard]] static std::string getExecutablePath();
00059
00060
00070
                     [[nodiscard]] static std::string getExecutableDir();
00071
           };
00072
00073 }//End of namespace wEngine 00074
00075 #endif
```

# Index

$\sim$ AssetManager	wEngine::ThicknessComponent, 136	
wEngine::AssetManager, 16	$\sim$ TitleEntity	
~AxisEntity	wPlot2D::TitleEntity, 140	
wPlot2D::AxisEntity, 22		
~ColorComponent	Above	
wEngine::ColorComponent, 30	wPlot2D, 13	
~Component	addAxis	
wEngine::Component, 31	wPlot2D::GraphicsEntity, 76	
~DataPlotEntity	addComponent	
wPlot2D::DataPlotEntity, 35	wEngine::Entity, 44	
~DiscontinuityComponent	addDataPlot	
wEngine::DiscontinuityComponent, 40	wPlot2D::GraphicsEntity, 77	
~Entity	addExcludedInterval	
wEngine::Entity, 43	wEngine::DiscontinuityComponent, 4	
~FontComponent	wPlot2D::FunctionEntity, 67	
wEngine::FontComponent, 49	addFont	
~FrameEntity	wPlot2D::GraphicsEntity, 73	
wPlot2D::FrameEntity, 53	addFunction	
~FunctionComponent	wPlot2D::GraphicsEntity, 77	
wEngine::FunctionComponent, 58	addItem	
~FunctionEntity	wPlot2D::LegendEntity, 89, 90	
wPlot2D::FunctionEntity, 62	addLabelsOffset	
~GraphicsEntity	wPlot2D::AxisEntity, 26	
wPlot2D::GraphicsEntity, 72	addLegend	
~LabelEntity	wPlot2D::GraphicsEntity, 77	
wPlot2D::LabelEntity, 82	addLine	
	wPlot2D::GraphicsEntity, 78	
~LegendEntity	addNotches	
wPlot2D::LegendEntity, 89	wPlot2D::AxisEntity, 24	
~LengthComponent	addOffset	
wEngine::LengthComponent, 95	wEngine::OffsetComponent, 119	
~LineEntity	addText	
wPlot2D::LineEntity, 102		
~LineStyleComponent	wPlot2D::GraphicsEntity, 78 addTitle	
wEngine::LineStyleComponent, 107		
~NotchEntity	wPlot2D::AxisEntity, 23 wPlot2D::GraphicsEntity, 76	
wPlot2D::NotchEntity, 113	alignToYAxis	
~NotchIntervalComponent	•	
wEngine::NotchIntervalComponent, 116	wPlot2D::FunctionEntity, 67	
~OffsetComponent	AssetManager	
wEngine::OffsetComponent, 119	wEngine::AssetManager, 16	
$\sim$ PaddingComponent	AxisEntity	
wEngine::PaddingComponent, 122	wPlot2D::AxisEntity, 22	
$\sim$ PositionComponent	AxisType	
wEngine::PositionComponent, 127	wPlot2D, 13	
$\sim$ RotationComponent	Dolous	
wEngine::RotationComponent, 130	Below	
$\sim$ ScaleComponent	wPlot2D, 13	
wEngine::ScaleComponent, 133	Bottom	
$\sim$ ThicknessComponent	wPlot2D, 13	

calculate	wPlot2D::FrameEntity, 52
wEngine::FunctionComponent, 58	FunctionComponent
Center	wEngine::FunctionComponent, 57
wPlot2D, 13	FunctionEntity
clearComponents	wPlot2D::FunctionEntity, 62
wEngine::Entity, 43	W ISLES III and ISLES III III III III III III III III III I
clearExcludedIntervals	getAngle
wEngine::DiscontinuityComponent, 41	wEngine::RotationComponent, 130
	getArrowSize
wPlot2D::FunctionEntity, 67	wPlot2D::LineEntity, 104
ColorComponent	getCharacterSize
wEngine::ColorComponent, 29	wPlot2D::LabelEntity, 83
Component	The state of the s
wEngine::Component, 31	wPlot2D::TitleEntity, 140
	getColor
Dashed	wEngine::ColorComponent, 30
wEngine::LineStyleComponent, 107	wPlot2D::DataPlotEntity, 36
DataPlotEntity	wPlot2D::FunctionEntity, 62
wPlot2D::DataPlotEntity, 35	getComponent
debugPrint	wEngine::Entity, 45
wEngine::ColorComponent, 30	getComponentTypeID
wEngine::DiscontinuityComponent, 41	wEngine, 12
wEngine::FontComponent, 49	getDashLength
wEngine::FunctionComponent, 58	wEngine::LineStyleComponent, 107
wEngine::LengthComponent, 95	wPlot2D::DataPlotEntity, 36
wEngine::LineStyleComponent, 109	wPlot2D::FunctionEntity, 63
wEngine::NotchIntervalComponent, 116	getDecimalPlaces
•	wPlot2D::LabelEntity, 83
wEngine::OffsetComponent, 119	getEndPoint
wEngine::PaddingComponent, 122	•
wEngine::PositionComponent, 128	wPlot2D::LineEntity, 103
wEngine::RotationComponent, 130	getEntityID
wEngine::ScaleComponent, 133	wEngine::Entity, 43
wEngine::ThicknessComponent, 136	getExcludedIntervals
debugPrintFonts	wEngine::DiscontinuityComponent, 41
wEngine::AssetManager, 17	getExecutableDir
disable	wEngine::PathUtils, 124
wEngine::Component, 32	getExecutablePath
DiscontinuityComponent	wEngine::PathUtils, 124
wEngine::DiscontinuityComponent, 40	getFillColor
Dotted	wPlot2D::FrameEntity, 53
wEngine::LineStyleComponent, 107	getFont
drawDataPlot	wEngine::AssetManager, 17
wPlot2D::DataPlotEntity, 38	wEngine::FontComponent, 49
drawFunction	wPlot2D::GraphicsEntity, 74
wPlot2D::FunctionEntity, 67	getFrameFillColor
	wPlot2D::TitleEntity, 142
drawLine	getFrameOutlineColor
wEngine::LineDrawer, 97	•
drawPolylineRound	wPlot2D::TitleEntity, 142
wEngine::LineDrawer, 97	getFrameThickness
	wPlot2D::TitleEntity, 142
enable	getGapLength
wEngine::Component, 32	wEngine::LineStyleComponent, 108
Entity	wPlot2D::DataPlotEntity, 36
wEngine::Entity, 43	wPlot2D::FunctionEntity, 63
	getInterfaceComponent
FontComponent	wEngine::Entity, 46
wEngine::FontComponent, 48	getInterval
formatLabel	wEngine::NotchIntervalComponent, 116
wPlot2D::LabelEntity, 85	getLabelsOffset
FrameEntity	wPlot2D::AxisEntity, 26
<del>-</del>	ICLD MICHILLY, 20

and and Desiries	(afficiently)
getLastPosition	isEnabled
wEngine::PositionComponent, 127	wEngine::Component, 32
getLength	wPlot2D::FrameEntity, 53
wEngine::LengthComponent, 95	isFrameEnabled
getLineStyle	wPlot2D::TitleEntity, 142
wPlot2D::DataPlotEntity, 36	isInExcludedInterval
wPlot2D::FunctionEntity, 63	wEngine::DiscontinuityComponent, 41
getNextComponentTypeID	
wEngine, 12	LabelEntity
getOffset	wPlot2D::LabelEntity, 82
wEngine::OffsetComponent, 119	LegendEntity
wPlot2D::FunctionEntity, 63	wPlot2D::LegendEntity, 89
wPlot2D::GraphicsEntity, 75	LengthComponent
•	wEngine::LengthComponent, 94
getOrigin	LineEntity
wPlot2D::GraphicsEntity, 74	wPlot2D::LineEntity, 101
getOutlineColor	LineStyle
wPlot2D::FrameEntity, 53	-
getPadding	wEngine::LineStyleComponent, 106
wEngine::PaddingComponent, 122	LineStyleComponent
wPlot2D::FrameEntity, 54	wEngine::LineStyleComponent, 107
wPlot2D::TitleEntity, 142	linspace
getParent	wEngine::MathUtils, 109
wEngine::Component, 32	LoadFont
getPosition	wEngine::AssetManager, 16
wEngine::PositionComponent, 127	
wPlot2D::FunctionEntity, 62	main
getRotation	main.cpp, 145
	main.cpp, 145
wPlot2D::FunctionEntity, 64	main, 145
getScale	move
wEngine::ScaleComponent, 133	wEngine::PositionComponent, 127
wPlot2D::GraphicsEntity, 75	WENGING COMOTICOMPONENT, 127
getStartPoint	NotchEntity
wPlot2D::LineEntity, 103	wPlot2D::NotchEntity, 113
getStyle	NotchIntervalComponent
wEngine::LineStyleComponent, 107	wEngine::NotchIntervalComponent, 115
getTextSize	NotchPosition
wPlot2D::TitleEntity, 140	
getThickness	wPlot2D, 13
wEngine::ThicknessComponent, 136	OffactComponent
wPlot2D::DataPlotEntity, 36	OffsetComponent
wPlot2D::FrameEntity, 54	wEngine::OffsetComponent, 118
<del>_</del>	operator=
wPlot2D::FunctionEntity, 62	wEngine::AssetManager, 16
wPlot2D::LineEntity, 102	D 11: 0
getTitleOffset	PaddingComponent
wPlot2D::AxisEntity, 24	wEngine::PaddingComponent, 122
getValue	PositionComponent
wPlot2D::LabelEntity, 83	wEngine::PositionComponent, 126
getWindow	
wPlot2D::GraphicsEntity, 72	removeComponent
getWindowSize	wEngine::Entity, 44
wPlot2D::GraphicsEntity, 72	RemoveFont
GraphicsEntity	wEngine::AssetManager, 17
wPlot2D::GraphicsEntity, 72	render
wi lotzbGraphicsEntity, 72	wPlot2D::AxisEntity, 27
hasArrow	wPlot2D::FrameEntity, 55
	wPlot2D::LabelEntity, 85
wPlot2D::LineEntity, 103	wPlot2D::LegendEntity, 92
hasComponent	
	- · · · · · · · · · · · · · · · · · · ·
wEngine::Entity, 45	wPlot2D::LineEntity, 104 wPlot2D::NotchEntity, 113

wPlot2D::TitleEntity, 144	wPlot2D::TitleEntity, 143	
requireComponent	setGapLength	
·	· -	
wEngine::Entity, 45	wEngine::LineStyleComponent, 108	
resetEntityIDCounter	wPlot2D::DataPlotEntity, 38	
wEngine::Entity, 44	wPlot2D::FunctionEntity, 65	
RotationComponent	wPlot2D::LineEntity, 103	
wEngine::RotationComponent, 129	setInterval	
	wEngine::NotchIntervalComponent, 116	
saveToFile	setLabelsCharacterSize	
wPlot2D::GraphicsEntity, 79	wPlot2D::AxisEntity, 26	
ScaleComponent	setLabelsColor	
wEngine::ScaleComponent, 132	wPlot2D::AxisEntity, 26	
setAngle	setLabelsDecimalPlaces	
wEngine::RotationComponent, 130	wPlot2D::AxisEntity, 27	
setArrowSize		
wPlot2D::AxisEntity, 23	setLabelsFont	
wPlot2D::LineEntity, 104	wPlot2D::AxisEntity, 25	
	setLabelsOffset	
setBackgroundColor	wPlot2D::AxisEntity, 26	
wPlot2D::GraphicsEntity, 73	setLabelText	
setCharacterSize	wPlot2D::LabelEntity, 84	
wPlot2D::LabelEntity, 84	setLength	
wPlot2D::LegendEntity, 92	wEngine::LengthComponent, 95	
wPlot2D::TitleEntity, 141	setLineStyle	
setColor	wPlot2D::DataPlotEntity, 37	
wEngine::ColorComponent, 30	wPlot2D::FunctionEntity, 65	
wPlot2D::AxisEntity, 22		
wPlot2D::DataPlotEntity, 37	wPlot2D::LineEntity, 102	
wPlot2D::FunctionEntity, 64	setNotchesColor	
wPlot2D::LineEntity, 102	wPlot2D::AxisEntity, 25	
	setNotchesLength	
setCustomLabels	wPlot2D::AxisEntity, 25	
wPlot2D::AxisEntity, 27	setNotchesThickness	
wPlot2D::LabelEntity, 84	wPlot2D::AxisEntity, 25	
setDashLength	setOffset	
wEngine::LineStyleComponent, 108	wEngine::OffsetComponent, 119	
wPlot2D::DataPlotEntity, 37	WEIIgilioOnootoomponont, 110	
wPlot2D::FunctionEntity, 65	wPlot2D::GraphicsEntity, 76	
wPlot2D::LineEntity, 103	wPlot2D::TitleEntity, 141	
setDecimalPlaces	• •	
wPlot2D::LabelEntity, 84	setOrigin	
setEnabled	wPlot2D::GraphicsEntity, 74	
wPlot2D::FrameEntity, 53	setOutlineColor	
•	wPlot2D::FrameEntity, 54	
setFillColor	setPadding	
wPlot2D::FrameEntity, 54	wEngine::PaddingComponent, 122	
setFont	wPlot2D::FrameEntity, 55	
wEngine::FontComponent, 49	wPlot2D::LegendEntity, 91	
wPlot2D::LabelEntity, 83	wPlot2D::TitleEntity, 144	
wPlot2D::LegendEntity, 91	setParent	
wPlot2D::TitleEntity, 141	wEngine::Component, 32	
setFrameEnabled	setPosition	
wPlot2D::LegendEntity, 90		
wPlot2D::TitleEntity, 143	wEngine::PositionComponent, 127	
setFrameFillColor	wPlot2D::FunctionEntity, 64	
	setRotation	
wPlot2D::LegendEntity, 90	wPlot2D::FunctionEntity, 66	
wPlot2D::TitleEntity, 143	setScale	
setFrameOutlineColor	wEngine::ScaleComponent, 133	
wPlot2D::LegendEntity, 91	wPlot2D::FunctionEntity, 66	
wPlot2D::TitleEntity, 143	wPlot2D::GraphicsEntity, 75	
setFrameThickness	setStyle	
wPlot2D::LegendEntity, 91		

wEngine::LineStyleComponent, 107	getFont, 17		
setTextColor	LoadFont, 16		
wPlot2D::LegendEntity, 92	operator=, 16		
wPlot2D::TitleEntity, 141	RemoveFont, 17		
setThickness	wEngine::ColorComponent, 28		
wEngine::ThicknessComponent, 136	~ColorComponent, 30		
wPlot2D::AxisEntity, 22	ColorComponent, 29		
wPlot2D::DataPlotEntity, 37	debugPrint, 30		
wPlot2D::FrameEntity, 55	getColor, 30		
wPlot2D::FunctionEntity, 64	setColor, 30		
wPlot2D::LineEntity, 102	wEngine::Component, 31		
setTitleCharacterSize	$\sim$ Component, 31		
wPlot2D::AxisEntity, 24	Component, 31		
setTitleColor	disable, 32		
wPlot2D::AxisEntity, 24	enable, 32		
setTitleFont	getParent, 32		
wPlot2D::AxisEntity, 23	isEnabled, 32		
setTitleOffset	setParent, 32		
wPlot2D::AxisEntity, 24	wEngine::DiscontinuityComponent, 38		
setWindowSize	~DiscontinuityComponent, 40		
wPlot2D::GraphicsEntity, 73	addExcludedInterval, 41		
setWindowTitle	clearExcludedIntervals, 41		
wPlot2D::GraphicsEntity, 73	debugPrint, 41		
· · · · · · · · · · · · · · · · · · ·	_		
Solid	DiscontinuityComponent, 40		
wEngine::LineStyleComponent, 107	getExcludedIntervals, 41		
ThicknessComponent	isInExcludedInterval, 41		
ThicknessComponent	wEngine::Entity, 42		
wEngine::ThicknessComponent, 135	∼Entity, <mark>43</mark>		
TitleAlignment	addComponent, 44		
wPlot2D, 13	clearComponents, 43		
TitleEntity	Entity, 43		
wPlot2D::TitleEntity, 140	getComponent, 45		
Тор	getEntityID, 43		
wPlot2D, 13	getInterfaceComponent, 46		
	hasComponent, 45		
update	removeComponent, 44		
wPlot2D::FrameEntity, 55	requireComponent, 45		
usesCustomLabels	resetEntityIDCounter, 44		
wPlot2D::LabelEntity, 85	wEngine::FontComponent, 47		
	~FontComponent, 49		
wAssetManager.cpp, 212	debugPrint, 49		
wAssetManager.hpp, 213, 214	FontComponent, 48		
wAxisEntity.cpp, 185	getFont, 49		
wAxisEntity.hpp, 185, 186	setFont, 49		
wColorComponent.cpp, 146	wEngine::FunctionComponent, 56		
wColorComponent.hpp, 146, 147	~FunctionComponent, 58		
wComponent.cpp, 178	calculate, 58		
wComponent.hpp, 179, 180			
wDataPlotEntity.cpp, 188	debugPrint, 58		
wDataPlotEntity.hpp, 188, 190	FunctionComponent, 57		
wDiscontinuityComponent.cpp, 148	wEngine::LengthComponent, 93		
wDiscontinuityComponent.hpp, 149, 150	~LengthComponent, 95		
wEngine, 11	debugPrint, 95		
getComponentTypeID, 12	getLength, 95		
getNextComponentTypeID, 12	LengthComponent, 94		
wEngine::AssetManager, 15	setLength, 95		
~AssetManager, 16	wEngine::LineDrawer, 96		
AssetManager, 16	drawLine, 97		
——————————————————————————————————————	drawPolylineRound, 97		
debugPrintFonts, 17			

wEngine::LineStyleComponent, 105	wEngine::ThicknessComponent, 134		
~LineStyleComponent, 107	~ThicknessComponent, 136		
Dashed, 107	debugPrint, 136		
debugPrint, 109	getThickness, 136		
Dotted, 107	setThickness, 136		
getDashLength, 107	ThicknessComponent, 135		
getGapLength, 108	wEntity.cpp, 181		
getStyle, 107	wEntity.hpp, 181, 183		
LineStyle, 106	wFontComponent.cpp, 150		
LineStyleComponent, 107	wFontComponent.hpp, 151, 152		
setDashLength, 108	wFrameEntity.cpp, 190		
setGapLength, 108	wFrameEntity.hpp, 191, 192		
setStyle, 107	wFunctionComponent.cpp, 153		
Solid, 107	wFunctionComponent.hpp, 154, 155		
wEngine::MathUtils, 109	wFunctionEntity.cpp, 193		
linspace, 109	wFunctionEntity.hpp, 194, 195		
wEngine::NotchIntervalComponent, 114	wGraphicsEntity.cpp, 196		
~NotchIntervalComponent, 116	wGraphicsEntity.hpp, 196, 197		
debugPrint, 116	wLabelEntity.cpp, 199		
getInterval, 116	wLabelEntity.hpp, 200, 201		
NotchIntervalComponent, 115	wLegendEntity.cpp, 202		
setInterval, 116	wLegendEntity.hpp, 202, 203		
wEngine::OffsetComponent, 117	wLengthComponent.cpp, 155		
~OffsetComponent, 119	wLengthComponent.hpp, 156, 157		
addOffset, 119	wLineDrawer.cpp, 214		
debugPrint, 119	wLineDrawer.hpp, 215, 217		
getOffset, 119	wLineEntity.cpp, 204		
OffsetComponent, 118	wLineEntity.hpp, 205, 206		
setOffset, 119	wLineStyleComponent.cpp, 158		
wEngine::PaddingComponent, 120	wLineStyleComponent.hpp, 159, 160		
~PaddingComponent, 122	wMathUtils.cpp, 217		
debugPrint, 122	wMathUtils.hpp, 218, 219		
getPadding, 122	wNotchEntity.cpp, 207		
PaddingComponent, 122	wNotchEntity.hpp, 208, 209		
setPadding, 122	wNotchIntervalComponent.cpp, 160		
wEngine::PathUtils, 123	wNotchIntervalComponent.hpp, 161, 162		
getExecutableDir, 124	wOffsetComponent.cpp, 163		
getExecutablePath, 124	wOffsetComponent.hpp, 164		
wEngine::PositionComponent, 125	wPaddingComponent.cpp, 165		
~PositionComponent, 127	wPaddingComponent.hpp, 166, 167		
debugPrint, 128	wPathUtils.cpp, 219		
getLastPosition, 127	wPathUtils.hpp, 220, 221		
getPosition, 127	wPlot2D, 12		
move, 127	Above, 13		
PositionComponent, 126	AxisType, 13		
setPosition, 127	Below, 13		
wEngine::RotationComponent, 128	Bottom, 13		
~RotationComponent, 130	Center, 13		
debugPrint, 130	NotchPosition, 13		
getAngle, 130	TitleAlignment, 13		
RotationComponent, 129	Top, 13		
setAngle, 130	X_AXIS, 13		
wEngine::ScaleComponent, 131	Y AXIS, 13		
~ScaleComponent, 133	wPlot2D - ECS-Based 2D Plotting Engine,		
debugPrint, 133	wPlot2D::AxisEntity, 18		
getScale, 133	~AxisEntity, 22		
ScaleComponent, 132	addLabelsOffset, 26		
setScale, 133	addNotches, 24		
, <del></del>			

addTitle, 23	getDashLength, 63	
AxisEntity, 22	getGapLength, 63	
getLabelsOffset, 26	getLineStyle, 63	
getTitleOffset, 24	getOffset, 63	
render, 27	getPosition, 62	
setArrowSize, 23	getRotation, 64	
setColor, 22	getThickness, 62	
setCustomLabels, 27	setColor, 64	
setLabelsCharacterSize, 26	setDashLength, 65	
setLabelsColor, 26	setGapLength, 65	
setLabelsDecimalPlaces, 27	setLineStyle, 65	
setLabelsFont, 25	setOffset, 66	
setLabelsOffset, 26	setPosition, 64	
setNotchesColor, 25	setRotation, 66	
setNotchesLength, 25	setScale, 66	
setNotchesThickness, 25	setThickness, 64	
setThickness, 22	wPlot2D::GraphicsEntity, 68	
setTitleCharacterSize, 24	$\sim$ GraphicsEntity, 72	
setTitleColor, 24	addAxis, 76	
setTitleFont, 23	addDataPlot, 77	
setTitleOffset, 24	addFont, 73	
wPlot2D::DataPlotEntity, 33	addFunction, 77	
~DataPlotEntity, 35	addLegend, 77	
DataPlotEntity, 35	addLine, 78	
drawDataPlot, 38	addText, 78	
getColor, 36	addTitle, 76	
getDashLength, 36	getFont, 74	
getGapLength, 36	get ont, 74 getOffset, 75	
getLineStyle, 36	getOrigin, 74	
getThickness, 36	getScale, 75	
setColor, 37	getWindow, 72	
setDashLength, 37	getWindowSize, 72	
setGapLength, 38	GraphicsEntity, 72	
setLineStyle, 37	saveToFile, 79	
setThickness, 37	setBackgroundColor, 73	
wPlot2D::FrameEntity, 50	setOffset, 76	
~FrameEntity, 53	setOrigin, 74	
FrameEntity, 52 getFillColor, 53	setScale, 75 setWindowSize, 73	
getOutlineColor, 53		
getPadding, 54	setWindowTitle, 73	
getThickness, 54	wPlot2D::LabelEntity, 79	
isEnabled, 53	~LabelEntity, 82	
•	formatLabel, 85	
render, 55	getCharacterSize, 83	
setEnabled, 53	getDecimalPlaces, 83	
setFillColor, 54	getValue, 83	
setOutlineColor, 54	LabelEntity, 82	
setPadding, 55	render, 85	
setThickness, 55	setCharacterSize, 84	
update, 55	setCustomLabels, 84	
wPlot2D::FunctionEntity, 59	setDecimalPlaces, 84	
~FunctionEntity, 62	setFont, 83	
addExcludedInterval, 67	setLabelText, 84	
alignToYAxis, 67	usesCustomLabels, 85	
clearExcludedIntervals, 67	wPlot2D::LegendEntity, 86	
drawFunction, 67	$\sim$ LegendEntity, 89	
FunctionEntity, 62	addItem, 89, 90	
getColor, 62	LegendEntity, 89	

	render, 92	X_AXIS
	setCharacterSize, 92	wPlot2D, 13
	setFont, 91	•
	setFrameEnabled, 90	Y_AXIS
	setFrameFillColor, 90	wPlot2D, 13
	setFrameOutlineColor, 91	
	setFrameThickness, 91	
	setPadding, 91	
	setTextColor, 92	
wPlc	ot2D::LineEntity, 98	
vvi ic	~LineEntity, 102	
	getArrowSize, 104	
	getEndPoint, 103	
	-	
	getStartPoint, 103	
	getThickness, 102	
	hasArrow, 103	
	LineEntity, 101	
	render, 104	
	setArrowSize, 104	
	setColor, 102	
	setDashLength, 103	
	setGapLength, 103	
	setLineStyle, 102	
	setThickness, 102	
wPlc	ot2D::NotchEntity, 110	
	∼NotchEntity, 113	
	NotchEntity, 113	
	render, 113	
wPlc	ot2D::TitleEntity, 137	
	∼TitleEntity, 140	
	getCharacterSize, 140	
	getFrameFillColor, 142	
	getFrameOutlineColor, 142	
	getFrameThickness, 142	
	getPadding, 142	
	getTextSize, 140	
	isFrameEnabled, 142	
	render, 144	
	setCharacterSize, 141	
	setFont, 141	
	setFrameEnabled, 143	
	setFrameFillColor, 143	
	setFrameOutlineColor, 143	
	setFrameThickness, 143	
	setOffset, 141	
	setPadding, 144	
	setTextColor, 141	
wD-	TitleEntity, 140	
	sitionComponent.cpp, 168	
	sitionComponent.hpp, 169, 170	
	tationComponent.cpp, 170	
	tationComponent.hpp, 172, 173	
	aleComponent.cpp, 173	
	aleComponent.hpp, 174, 175	
	icknessComponent.cpp, 176	
	icknessComponent.hpp, 177, 178	
wTitl	leEntity.cpp, 209	
wTitl	leEntity.hop 210 211	