

Visualizer

1.0.0

Generated by Doxygen 1.9.1

1 VISUALIZER	1
1.1 Dependency	1
1.2 Demonstrate	1
1.3 Features	1
1.3.1 Main features	1
1.3.2 Data structures	2
1.4 Documentation	2
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	13
6.1 DISPLAY Namespace Reference	13
6.1.1 Detailed Description	13
6.1.2 Variable Documentation	13
6.1.2.1 HOME_	13
6.1.2.2 WORKING_	13
6.2 FILEE Namespace Reference	14
6.2.1 Detailed Description	14
6.2.2 Function Documentation	14
6.2.2.1 readFile()	14
6.3 JSON Namespace Reference	14
6.3.1 Detailed Description	14
6.3.2 Function Documentation	15
6.3.2.1 readFile()	15
6.3.2.2 saveFile()	15
6.4 NUMBER Namespace Reference	15
6.4.1 Detailed Description	16
6.4.2 Function Documentation	16
6.4.2.1 intToString()	16
6.4.2.2 isDigit()	16
6.4.2.3 isInterval()	17
6.4.2.4 isLetter()	17
6.4.2.5 isNumber()	17
6.4.2.6 isOperator()	17

6.4.2.7 isSign()	18
6.4.2.8 isSymbol()	18
6.4.2.9 removeLeadingZero()	18
6.4.2.10 stringToArray()	18
6.4.2.11 stringToInt()	19
6.4.3 Variable Documentation	19
6.4.3.1 INF	19
6.5 PATH Namespace Reference	19
6.5.1 Detailed Description	20
6.5.2 Variable Documentation	20
6.5.2.1 ASSETS_	20
6.5.2.2 ATTRIBUTE_	20
6.5.2.3 SAVING_	20
6.6 PATH::ASSETS Namespace Reference	20
6.6.1 Variable Documentation	20
6.6.1.1 FONTS_	21
6.6.1.2 GRAPHICS_	21
6.6.1.3 SCRIPT_	21
6.7 PATH::ATB Namespace Reference	21
6.7.1 Variable Documentation	21
6.7.1.1 BUTTON_	21
6.7.1.2 DATA_STRUCTURES_	22
6.7.1.3 DISPLAY_	22
6.7.1.4 INPUTBOX_	22
6.7.1.5 OBJECT_	22
6.7.1.6 SCRIPT_	22
6.7.1.7 SPRITE_	22
6.8 PATH::SAVING Namespace Reference	23
6.8.1 Variable Documentation	23
6.8.1.1 AVL_	23
6.8.1.2 BTREE_4TH_	23
6.8.1.3 GRAPH_	23
6.8.1.4 HASH_TABLE_	23
6.8.1.5 MAX_HEAP_	24
6.8.1.6 MIN_HEAP_	24
6.8.1.7 TRIE_	24
6.9 RANDOM Namespace Reference	24
6.9.1 Detailed Description	24
6.9.2 Function Documentation	25
6.9.2.1 flipCoin()	25
6.9.2.2 getChar() [1/2]	25
6.9.2.3 getChar() [2/2]	25

6.9.2.4 <code>getDouble()</code>	25
6.9.2.5 <code>getFloat()</code>	26
6.9.2.6 <code>getInt()</code> [1/2]	26
6.9.2.7 <code>getInt()</code> [2/2]	26
6.9.2.8 <code>getLongLong()</code>	26
6.9.2.9 <code>getString()</code> [1/2]	27
6.9.2.10 <code>getString()</code> [2/2]	27
6.9.3 Variable Documentation	27
6.9.3.1 <code>rng</code>	27
6.10 SIUSTRING Namespace Reference	27
6.10.1 Detailed Description	28
6.10.2 Function Documentation	28
6.10.2.1 <code>isSeparator()</code>	28
6.10.2.2 <code>split()</code>	28
7 Class Documentation	29
7.1 AVL Class Reference	29
7.1.1 Detailed Description	30
7.1.2 Constructor & Destructor Documentation	30
7.1.2.1 <code>AVL()</code>	30
7.1.2.2 <code>~AVL()</code>	31
7.1.3 Member Function Documentation	31
7.1.3.1 <code>balanceFactor()</code>	31
7.1.3.2 <code>balancing()</code>	32
7.1.3.3 <code>closeScript()</code>	32
7.1.3.4 <code>getHigh()</code>	32
7.1.3.5 <code>goBack()</code>	33
7.1.3.6 <code>goNext()</code>	33
7.1.3.7 <code>goOff()</code>	33
7.1.3.8 <code>goOn()</code>	33
7.1.3.9 <code>highlight()</code>	34
7.1.3.10 <code>init()</code>	34
7.1.3.11 <code>insert()</code> [1/2]	34
7.1.3.12 <code>insert()</code> [2/2]	35
7.1.3.13 <code>isReceiveEvent()</code>	36
7.1.3.14 <code>locating()</code>	36
7.1.3.15 <code>maxDepth()</code> [1/2]	37
7.1.3.16 <code>maxDepth()</code> [2/2]	37
7.1.3.17 <code>react()</code>	37
7.1.3.18 <code>remove()</code> [1/2]	38
7.1.3.19 <code>remove()</code> [2/2]	38
7.1.3.20 <code>rendering()</code>	40

7.1.3.21 renderLine()	40
7.1.3.22 rotateLeft()	41
7.1.3.23 rotateRight()	41
7.1.3.24 search() [1/2]	42
7.1.3.25 search() [2/2]	42
7.1.3.26 setEdgesColor()	43
7.1.3.27 setNodeColor()	43
7.1.3.28 setting()	43
7.1.3.29 slowDown()	44
7.1.3.30 speedUp()	44
7.1.3.31 unhighlight()	44
7.1.3.32 unplugSmallest()	44
7.1.3.33 waitForStep()	45
7.2 BTree4th Class Reference	45
7.2.1 Detailed Description	46
7.2.2 Constructor & Destructor Documentation	46
7.2.2.1 BTree4th()	46
7.2.2.2 ~BTree4th()	46
7.2.3 Member Function Documentation	46
7.2.3.1 addRecordToLeaf()	46
7.2.3.2 init()	47
7.2.3.3 insert() [1/2]	47
7.2.3.4 insert() [2/2]	48
7.2.3.5 leftMost()	48
7.2.3.6 mergeChild() [1/2]	48
7.2.3.7 mergeChild() [2/2]	49
7.2.3.8 remove()	49
7.2.3.9 rightMost()	49
7.2.3.10 search() [1/2]	49
7.2.3.11 search() [2/2]	50
7.2.3.12 split()	50
7.3 Button Class Reference	51
7.3.1 Detailed Description	52
7.3.2 Constructor & Destructor Documentation	53
7.3.2.1 Button()	53
7.3.2.2 ~Button()	53
7.3.3 Member Function Documentation	53
7.3.3.1 addCharacter()	53
7.3.3.2 addText()	54
7.3.3.3 changeToCircle() [1/5]	54
7.3.3.4 changeToCircle() [2/5]	54
7.3.3.5 changeToCircle() [3/5]	54

7.3.3.6 changeToCircle() [4/5]	55
7.3.3.7 changeToCircle() [5/5]	55
7.3.3.8 changeToRectangle()	55
7.3.3.9 coloring() [1/3]	55
7.3.3.10 coloring() [2/3]	56
7.3.3.11 coloring() [3/3]	56
7.3.3.12 cropping() [1/3]	56
7.3.3.13 cropping() [2/3]	56
7.3.3.14 cropping() [3/3]	57
7.3.3.15 fillCircleByColor()	57
7.3.3.16 fillRectangleByColor()	58
7.3.3.17 fillWithColor()	58
7.3.3.18 fitTheTexture()	58
7.3.3.19 getAction()	58
7.3.3.20 getColor()	59
7.3.3.21 getCrop()	59
7.3.3.22 getDataType()	59
7.3.3.23 getH()	59
7.3.3.24 getInputType()	60
7.3.3.25 getLocation()	60
7.3.3.26 getNextScreen()	60
7.3.3.27 getSize()	60
7.3.3.28 getText()	60
7.3.3.29 getW()	61
7.3.3.30 getX()	61
7.3.3.31 getY()	61
7.3.3.32 hide()	61
7.3.3.33 importFromJson() [1/2]	61
7.3.3.34 importFromJson() [2/2]	62
7.3.3.35 initAction()	62
7.3.3.36 initBackground()	63
7.3.3.37 initSprites()	63
7.3.3.38 isChooosed()	64
7.3.3.39 isClicked()	64
7.3.3.40 isHover()	64
7.3.3.41 isLielInside() [1/4]	65
7.3.3.42 isLielInside() [2/4]	65
7.3.3.43 isLielInside() [3/4]	65
7.3.3.44 isLielInside() [4/4]	65
7.3.3.45 isReceiveEvent()	66
7.3.3.46 isVisible()	66
7.3.3.47 linking()	66

7.3.3.48 locating() [1/3]	66
7.3.3.49 locating() [2/3]	67
7.3.3.50 locating() [3/3]	67
7.3.3.51 locatingH()	67
7.3.3.52 locatingW()	67
7.3.3.53 locatingX()	68
7.3.3.54 locatingY()	68
7.3.3.55 move()	68
7.3.3.56 moveX()	68
7.3.3.57 moveY()	69
7.3.3.58 noCropping()	69
7.3.3.59 removeCharacter() [1/2]	69
7.3.3.60 removeCharacter() [2/2]	69
7.3.3.61 rendering()	70
7.3.3.62 setFont()	70
7.3.3.63 setShape()	70
7.3.3.64 setText()	71
7.3.3.65 show()	71
7.3.3.66 textToTexture()	71
7.3.3.67 textureFromFile()	71
7.3.3.68 zoom()	72
7.3.3.69 zoomH()	72
7.3.3.70 zoomInMiddle()	72
7.3.3.71 zoomW()	72
7.4 DataStructures Class Reference	73
7.4.1 Detailed Description	75
7.4.2 Constructor & Destructor Documentation	75
7.4.2.1 DataStructures()	76
7.4.2.2 ~DataStructures()	76
7.4.3 Member Function Documentation	76
7.4.3.1 addCharacter()	76
7.4.3.2 addText()	77
7.4.3.3 changeToCircle() [1/5]	77
7.4.3.4 changeToCircle() [2/5]	77
7.4.3.5 changeToCircle() [3/5]	77
7.4.3.6 changeToCircle() [4/5]	78
7.4.3.7 changeToCircle() [5/5]	78
7.4.3.8 changeToRectangle()	78
7.4.3.9 closeScript()	78
7.4.3.10 coloring() [1/3]	79
7.4.3.11 coloring() [2/3]	79
7.4.3.12 coloring() [3/3]	79

7.4.3.13 cropping() [1/3]	80
7.4.3.14 cropping() [2/3]	80
7.4.3.15 cropping() [3/3]	80
7.4.3.16 Dijkstra()	80
7.4.3.17 dijkstra()	81
7.4.3.18 fillCircleByColor()	81
7.4.3.19 fillRectangleByColor()	82
7.4.3.20 fillWithColor()	82
7.4.3.21 fitTheTexture()	82
7.4.3.22 getColor()	82
7.4.3.23 getCrop()	83
7.4.3.24 getData Type()	83
7.4.3.25 getH()	83
7.4.3.26 getLocation()	83
7.4.3.27 getName()	83
7.4.3.28 getSize()	84
7.4.3.29 getText()	84
7.4.3.30 getW()	84
7.4.3.31 getX()	84
7.4.3.32 getY()	84
7.4.3.33 goBack()	85
7.4.3.34 goNext()	85
7.4.3.35 goOff()	85
7.4.3.36 goOn()	85
7.4.3.37 hide()	86
7.4.3.38 importFromJson() [1/2]	86
7.4.3.39 importFromJson() [2/2]	86
7.4.3.40 init()	87
7.4.3.41 initAVL()	87
7.4.3.42 initBackground()	87
7.4.3.43 initDisplay()	88
7.4.3.44 initGraph()	88
7.4.3.45 initHashTable()	88
7.4.3.46 initLinker()	89
7.4.3.47 initMinHeap()	89
7.4.3.48 initTrie()	89
7.4.3.49 insert()	89
7.4.3.50 insertAVL()	90
7.4.3.51 insertHashTable()	90
7.4.3.52 insertMinHeap()	90
7.4.3.53 insertTrie()	90
7.4.3.54 isLiInside() [1/4]	91

7.4.3.55 isLieInside() [2/4]	91
7.4.3.56 isLieInside() [3/4]	91
7.4.3.57 isLieInside() [4/4]	91
7.4.3.58 isReceiveEvent()	92
7.4.3.59 isVisible()	92
7.4.3.60 linking()	92
7.4.3.61 locating() [1/3]	92
7.4.3.62 locating() [2/3]	93
7.4.3.63 locating() [3/3]	93
7.4.3.64 locatingH()	93
7.4.3.65 locatingW()	93
7.4.3.66 locatingX()	94
7.4.3.67 locatingY()	94
7.4.3.68 moveX()	94
7.4.3.69 moveY()	94
7.4.3.70 MST()	95
7.4.3.71 mst()	95
7.4.3.72 noCropping()	95
7.4.3.73 react()	95
7.4.3.74 remove()	96
7.4.3.75 removeAVL()	96
7.4.3.76 removeCharacter() [1/2]	96
7.4.3.77 removeCharacter() [2/2]	97
7.4.3.78 removeHashTable()	97
7.4.3.79 removeMinHeap()	97
7.4.3.80 removeTrie()	97
7.4.3.81 rendering()	98
7.4.3.82 SCC()	98
7.4.3.83 scc()	98
7.4.3.84 search()	98
7.4.3.85 searchAVL()	99
7.4.3.86 searchHashTable()	99
7.4.3.87 searchMinHeap()	99
7.4.3.88 searchTrie()	99
7.4.3.89 setDataType()	100
7.4.3.90 setFont()	100
7.4.3.91 setShape()	100
7.4.3.92 setText()	101
7.4.3.93 setting()	101
7.4.3.94 settingAVL()	102
7.4.3.95 settingGraph()	102
7.4.3.96 settingHashTable()	103

7.4.3.97 settingMinHeap()	103
7.4.3.98 settingTrie()	104
7.4.3.99 show()	105
7.4.3.100 size()	105
7.4.3.101 slowDown()	105
7.4.3.102 speedUp()	106
7.4.3.103 textToTexture()	106
7.4.3.104 textureFromFile()	106
7.4.3.105 top()	107
7.4.3.106 zoom()	107
7.4.3.107 zoomH()	107
7.4.3.108 zoomInMiddle()	107
7.4.3.109 zoomW()	108
7.4.4 Member Data Documentation	108
7.4.4.1 capacity	108
7.5 Display Class Reference	108
7.5.1 Detailed Description	110
7.5.2 Constructor & Destructor Documentation	110
7.5.2.1 Display()	110
7.5.2.2 ~Display()	110
7.5.3 Member Function Documentation	111
7.5.3.1 addCharacter()	111
7.5.3.2 addText()	111
7.5.3.3 changeToCircle() [1/5]	111
7.5.3.4 changeToCircle() [2/5]	111
7.5.3.5 changeToCircle() [3/5]	112
7.5.3.6 changeToCircle() [4/5]	112
7.5.3.7 changeToCircle() [5/5]	112
7.5.3.8 changeToRectangle()	112
7.5.3.9 coloring() [1/3]	113
7.5.3.10 coloring() [2/3]	113
7.5.3.11 coloring() [3/3]	113
7.5.3.12 cropping() [1/3]	113
7.5.3.13 cropping() [2/3]	114
7.5.3.14 cropping() [3/3]	114
7.5.3.15 fillCircleByColor()	114
7.5.3.16 fillRectangleByColor()	115
7.5.3.17 fillWithColor()	115
7.5.3.18 fitTheTexture()	115
7.5.3.19 getColor()	116
7.5.3.20 getCrop()	116
7.5.3.21 getH()	116

7.5.3.22 getLocation()	116
7.5.3.23 getSize()	116
7.5.3.24 getText()	117
7.5.3.25 getW()	117
7.5.3.26 getX()	117
7.5.3.27 getY()	117
7.5.3.28 hide()	117
7.5.3.29 importFromJson() [1/2]	118
7.5.3.30 importFromJson() [2/2]	118
7.5.3.31 initBackground()	118
7.5.3.32 initButtons()	119
7.5.3.33 isButtonReceiveEvent()	119
7.5.3.34 isLieInside() [1/4]	119
7.5.3.35 isLieInside() [2/4]	119
7.5.3.36 isLieInside() [3/4]	120
7.5.3.37 isLieInside() [4/4]	120
7.5.3.38 isReceiveEvent()	120
7.5.3.39 isVisible()	120
7.5.3.40 linking()	121
7.5.3.41 locating() [1/3]	121
7.5.3.42 locating() [2/3]	121
7.5.3.43 locating() [3/3]	121
7.5.3.44 locatingH()	122
7.5.3.45 locatingW()	122
7.5.3.46 locatingX()	122
7.5.3.47 locatingY()	122
7.5.3.48 moveX()	123
7.5.3.49 moveY()	123
7.5.3.50 noCropping()	123
7.5.3.51 react()	123
7.5.3.52 removeCharacter() [1/2]	124
7.5.3.53 removeCharacter() [2/2]	124
7.5.3.54 rendering()	124
7.5.3.55 setFont()	124
7.5.3.56 setShape()	125
7.5.3.57 setText()	125
7.5.3.58 show()	125
7.5.3.59 textToTexture()	126
7.5.3.60 textureFromFile()	126
7.5.3.61 zoom()	126
7.5.3.62 zoomH()	126
7.5.3.63 zoomInMiddle()	127

7.5.3.64 zoomW()	127
7.6 distanceHeap Struct Reference	127
7.6.1 Detailed Description	128
7.6.2 Constructor & Destructor Documentation	128
7.6.2.1 distanceHeap()	128
7.6.2.2 ~distanceHeap()	128
7.6.3 Member Function Documentation	128
7.6.3.1 empty()	128
7.6.3.2 insert()	129
7.6.3.3 pop()	129
7.6.3.4 swap()	129
7.6.3.5 swappable()	130
7.6.4 Member Data Documentation	130
7.6.4.1 g	130
7.6.4.2 value	130
7.7 DSU Struct Reference	130
7.7.1 Detailed Description	131
7.7.2 Constructor & Destructor Documentation	131
7.7.2.1 DSU()	131
7.7.2.2 ~DSU()	131
7.7.3 Member Function Documentation	131
7.7.3.1 find() [1/2]	131
7.7.3.2 find() [2/2]	132
7.7.3.3 isUnionized() [1/3]	132
7.7.3.4 isUnionized() [2/3]	132
7.7.3.5 isUnionized() [3/3]	132
7.7.3.6 unionEdge() [1/3]	133
7.7.3.7 unionEdge() [2/3]	133
7.7.3.8 unionEdge() [3/3]	133
7.7.4 Member Data Documentation	133
7.7.4.1 parent	133
7.8 Graph Class Reference	134
7.8.1 Detailed Description	134
7.8.2 Constructor & Destructor Documentation	134
7.8.2.1 Graph()	135
7.8.2.2 ~Graph()	135
7.8.3 Member Function Documentation	135
7.8.3.1 Dijkstra()	135
7.8.3.2 init()	136
7.8.3.3 isReceiveEvent()	137
7.8.3.4 MST()	137
7.8.3.5 react()	138

7.8.3.6 renderEdge()	138
7.8.3.7 rendering()	139
7.8.3.8 repair()	139
7.8.3.9 SCC()	140
7.8.3.10 setting()	140
7.8.3.11 Tarjan()	141
7.8.3.12 unionEdges()	141
7.8.3.13 waitForStep()	141
7.8.4 Friends And Related Function Documentation	142
7.8.4.1 distanceHeap	142
7.8.4.2 DSU	142
7.9 HashTable Class Reference	142
7.9.1 Detailed Description	143
7.9.2 Constructor & Destructor Documentation	143
7.9.2.1 HashTable()	143
7.9.2.2 ~HashTable()	144
7.9.3 Member Function Documentation	144
7.9.3.1 closeScript()	144
7.9.3.2 defaultSetting()	145
7.9.3.3 drawEdge()	145
7.9.3.4 goBack()	145
7.9.3.5 goNext()	145
7.9.3.6 goOff()	146
7.9.3.7 goOn()	146
7.9.3.8 highlight()	146
7.9.3.9 init()	146
7.9.3.10 insert() [1/2]	147
7.9.3.11 insert() [2/2]	147
7.9.3.12 isReceiveEvent()	148
7.9.3.13 locating() [1/2]	149
7.9.3.14 locating() [2/2]	149
7.9.3.15 react()	149
7.9.3.16 remove() [1/2]	150
7.9.3.17 remove() [2/2]	151
7.9.3.18 renderEdges()	151
7.9.3.19 rendering()	152
7.9.3.20 search() [1/2]	152
7.9.3.21 search() [2/2]	153
7.9.3.22 setEdgesColor()	153
7.9.3.23 setNodeColor()	153
7.9.3.24 setting()	153
7.9.3.25 slowDown()	154

7.9.3.26 speedUp()	154
7.9.3.27 unhighlight()	154
7.9.3.28 waitForStep()	155
7.10 InputBox Class Reference	155
7.10.1 Detailed Description	156
7.10.2 Constructor & Destructor Documentation	156
7.10.2.1 InputBox()	156
7.10.2.2 ~InputBox()	157
7.10.3 Member Function Documentation	157
7.10.3.1 getOperator()	157
7.10.3.2 getText()	157
7.10.3.3 importFromJson()	157
7.10.3.4 initBackground()	158
7.10.3.5 initButtons()	158
7.10.3.6 initOperator()	158
7.10.3.7 initSprites()	159
7.10.3.8 isButtonReceiveEvent()	159
7.10.3.9 isInputReceiveEvent()	159
7.10.3.10 isReceiveEvent()	160
7.10.3.11 linking()	160
7.10.3.12 react()	160
7.10.3.13 rendering()	161
7.10.3.14 setDuplicate()	161
7.10.3.15 setText() [1/2]	162
7.10.3.16 setText() [2/2]	162
7.10.4 Member Data Documentation	162
7.10.4.1 m	162
7.10.4.2 n	162
7.11 maxHeap Class Reference	163
7.11.1 Detailed Description	163
7.11.2 Constructor & Destructor Documentation	163
7.11.2.1 maxHeap()	163
7.11.2.2 ~maxHeap()	163
7.11.3 Member Function Documentation	164
7.11.3.1 heapify()	164
7.11.3.2 init()	164
7.11.3.3 insert()	164
7.11.3.4 pop()	165
7.11.3.5 size()	165
7.11.3.6 swap()	165
7.11.3.7 swapable()	166
7.11.3.8 top()	166

7.12 minHeap Class Reference	166
7.12.1 Detailed Description	167
7.12.2 Constructor & Destructor Documentation	167
7.12.2.1 minHeap()	167
7.12.2.2 ~minHeap()	168
7.12.3 Member Function Documentation	168
7.12.3.1 closeScript()	168
7.12.3.2 heapify()	169
7.12.3.3 highlight()	169
7.12.3.4 init()	170
7.12.3.5 insert()	170
7.12.3.6 isReceiveEvent()	170
7.12.3.7 locating()	171
7.12.3.8 pop()	171
7.12.3.9 react()	172
7.12.3.10 realInsert()	172
7.12.3.11 rendering()	173
7.12.3.12 renderLine()	174
7.12.3.13 setmax()	174
7.12.3.14 setmin()	174
7.12.3.15 setting()	175
7.12.3.16 size()	175
7.12.3.17 swap()	175
7.12.3.18 swapable()	175
7.12.3.19 top()	176
7.12.3.20 unhighlight()	176
7.12.3.21 waitForStep()	176
7.13 MyWindow Class Reference	177
7.13.1 Detailed Description	177
7.13.2 Constructor & Destructor Documentation	178
7.13.2.1 MyWindow()	178
7.13.2.2 ~MyWindow()	178
7.13.3 Member Function Documentation	179
7.13.3.1 getDataFromFile()	179
7.13.3.2 getDataFromRandom()	180
7.13.3.3 getEvent()	181
7.13.3.4 initSDL2()	181
7.13.3.5 interacting()	182
7.13.3.6 isClosed()	182
7.13.3.7 isOpen()	183
7.13.3.8 react()	183
7.13.3.9 rendering()	185

7.13.3.10 run()	186
7.13.3.11 runOperator()	186
7.13.3.12 setDataType()	187
7.13.3.13 setDisplay()	187
7.13.3.14 setInputBox()	188
7.14 Object Class Reference	188
7.14.1 Detailed Description	190
7.14.2 Constructor & Destructor Documentation	190
7.14.2.1 Object()	190
7.14.2.2 ~Object()	190
7.14.3 Member Function Documentation	191
7.14.3.1 addCharacter()	191
7.14.3.2 addText()	191
7.14.3.3 changeToCircle() [1/5]	191
7.14.3.4 changeToCircle() [2/5]	191
7.14.3.5 changeToCircle() [3/5]	192
7.14.3.6 changeToCircle() [4/5]	192
7.14.3.7 changeToCircle() [5/5]	192
7.14.3.8 changeToRectangle()	192
7.14.3.9 coloring() [1/3]	193
7.14.3.10 coloring() [2/3]	193
7.14.3.11 coloring() [3/3]	193
7.14.3.12 cropping() [1/3]	193
7.14.3.13 cropping() [2/3]	194
7.14.3.14 cropping() [3/3]	194
7.14.3.15 fillCircleByColor()	194
7.14.3.16 fillRectangleByColor()	195
7.14.3.17 fillWithColor()	195
7.14.3.18 fitTheTexture()	195
7.14.3.19 getColor()	196
7.14.3.20 getCrop()	196
7.14.3.21 getH()	196
7.14.3.22 getLocation()	196
7.14.3.23 getSize()	196
7.14.3.24 getText()	197
7.14.3.25 getW()	197
7.14.3.26 getX()	197
7.14.3.27 getY()	197
7.14.3.28 hide()	197
7.14.3.29 importFromJson()	198
7.14.3.30 isLieInside() [1/4]	198
7.14.3.31 isLieInside() [2/4]	198

7.14.3.32 isLieInside() [3/4]	199
7.14.3.33 isLieInside() [4/4]	199
7.14.3.34 isVisible()	199
7.14.3.35 linking()	199
7.14.3.36 locating() [1/3]	200
7.14.3.37 locating() [2/3]	200
7.14.3.38 locating() [3/3]	200
7.14.3.39 locatingH()	200
7.14.3.40 locatingW()	201
7.14.3.41 locatingX()	201
7.14.3.42 locatingY()	201
7.14.3.43 moveX()	201
7.14.3.44 moveY()	202
7.14.3.45 noCropping()	202
7.14.3.46 removeCharacter() [1/2]	202
7.14.3.47 removeCharacter() [2/2]	202
7.14.3.48 rendering()	203
7.14.3.49 setFont()	203
7.14.3.50 setShape()	203
7.14.3.51 setText()	203
7.14.3.52 show()	204
7.14.3.53 textToTexture()	204
7.14.3.54 textureFromFile()	204
7.14.3.55 zoom()	204
7.14.3.56 zoomH()	205
7.14.3.57 zoomInMiddle()	205
7.14.3.58 zoomW()	205
7.15 Point Struct Reference	206
7.15.1 Detailed Description	206
7.15.2 Member Data Documentation	206
7.15.2.1 x	206
7.15.2.2 y	206
7.16 Position Struct Reference	206
7.16.1 Detailed Description	207
7.16.2 Member Data Documentation	207
7.16.2.1 center	207
7.16.2.2 crop	207
7.16.2.3 location	207
7.16.2.4 radius	207
7.16.2.5 visible	208
7.17 Script Class Reference	208
7.17.1 Detailed Description	209

7.17.2 Constructor & Destructor Documentation	209
7.17.2.1 Script()	209
7.17.2.2 ~Script()	209
7.17.3 Member Function Documentation	209
7.17.3.1 highlight()	209
7.17.3.2 importFromJson()	210
7.17.3.3 initBackground()	210
7.17.3.4 initButtons()	210
7.17.3.5 isReceiveEvent()	211
7.17.3.6 linking()	211
7.17.3.7 react()	211
7.17.3.8 rendering()	212
7.17.3.9 unhighlight()	213
7.18 Sprite Class Reference	213
7.18.1 Detailed Description	215
7.18.2 Constructor & Destructor Documentation	215
7.18.2.1 Sprite()	215
7.18.2.2 ~Sprite()	215
7.18.3 Member Function Documentation	216
7.18.3.1 addCharacter()	216
7.18.3.2 addText()	216
7.18.3.3 aligning() [1/4]	216
7.18.3.4 aligning() [2/4]	217
7.18.3.5 aligning() [3/4]	217
7.18.3.6 aligning() [4/4]	217
7.18.3.7 backspace() [1/2]	217
7.18.3.8 backspace() [2/2]	218
7.18.3.9 changeToCircle() [1/5]	218
7.18.3.10 changeToCircle() [2/5]	218
7.18.3.11 changeToCircle() [3/5]	218
7.18.3.12 changeToCircle() [4/5]	219
7.18.3.13 changeToCircle() [5/5]	219
7.18.3.14 changeToRectangle()	219
7.18.3.15 coloring() [1/3]	219
7.18.3.16 coloring() [2/3]	220
7.18.3.17 coloring() [3/3]	220
7.18.3.18 cropping() [1/3]	220
7.18.3.19 cropping() [2/3]	220
7.18.3.20 cropping() [3/3]	221
7.18.3.21 fillCircleByColor()	221
7.18.3.22 fillRectangleByColor()	222
7.18.3.23 fillWithColor()	222

7.18.3.24 fitTheTexture()	222
7.18.3.25 getColor()	222
7.18.3.26 getCrop()	223
7.18.3.27 getH()	223
7.18.3.28 getLocation()	223
7.18.3.29 getSize()	223
7.18.3.30 getText()	223
7.18.3.31 getW()	224
7.18.3.32 getX()	224
7.18.3.33 getY()	224
7.18.3.34 hide()	224
7.18.3.35 highlight()	224
7.18.3.36 importFromJson() [1/2]	225
7.18.3.37 importFromJson() [2/2]	225
7.18.3.38 initBackground()	225
7.18.3.39 initInput()	226
7.18.3.40 initObjects()	226
7.18.3.41 initTextBox()	226
7.18.3.42 isLieInside() [1/4]	227
7.18.3.43 isLieInside() [2/4]	227
7.18.3.44 isLieInside() [3/4]	227
7.18.3.45 isLieInside() [4/4]	227
7.18.3.46 isVisible()	228
7.18.3.47 linking()	228
7.18.3.48 locating() [1/3]	228
7.18.3.49 locating() [2/3]	228
7.18.3.50 locating() [3/3]	229
7.18.3.51 locatingH()	229
7.18.3.52 locatingW()	229
7.18.3.53 locatingX()	229
7.18.3.54 locatingY()	230
7.18.3.55 moveX()	230
7.18.3.56 moveY()	230
7.18.3.57 noCropping()	230
7.18.3.58 removeCharacter() [1/2]	231
7.18.3.59 removeCharacter() [2/2]	231
7.18.3.60 rendering()	231
7.18.3.61 setFont()	231
7.18.3.62 setFontColor()	232
7.18.3.63 setShape()	232
7.18.3.64 setText()	232
7.18.3.65 setTextBox()	233

7.18.3.66 setTextBoxTransparent()	233
7.18.3.67 show()	233
7.18.3.68 textToTexture()	233
7.18.3.69 textureFromFile()	234
7.18.3.70 typing() [1/2]	234
7.18.3.71 typing() [2/2]	234
7.18.3.72 unhighlight()	234
7.18.3.73 zoom()	235
7.18.3.74 zoomH()	235
7.18.3.75 zoomInMiddle()	235
7.18.3.76 zoomW()	235
7.19 Trie Class Reference	236
7.19.1 Detailed Description	236
7.19.2 Constructor & Destructor Documentation	236
7.19.2.1 Trie()	237
7.19.2.2 ~Trie()	237
7.19.3 Member Function Documentation	237
7.19.3.1 closeScript()	238
7.19.3.2 drawEdges()	238
7.19.3.3 highlight()	238
7.19.3.4 init()	238
7.19.3.5 insert() [1/2]	239
7.19.3.6 insert() [2/2]	240
7.19.3.7 isReceiveEvent()	240
7.19.3.8 locating()	241
7.19.3.9 react()	241
7.19.3.10 remove() [1/2]	242
7.19.3.11 remove() [2/2]	243
7.19.3.12 rendering()	243
7.19.3.13 search() [1/2]	244
7.19.3.14 search() [2/2]	244
7.19.3.15 setEdgesColor()	245
7.19.3.16 setNodeColor()	245
7.19.3.17 setting()	245
7.19.3.18 unhighlight()	246
7.19.3.19 waitForStep()	246
8 File Documentation	247
8.1 include/button.hpp File Reference	247
8.2 include/data_structures.hpp File Reference	247
8.3 include/data_structures/AVL.hpp File Reference	248
8.4 include/data_structures/btree4th.hpp File Reference	248

8.5 include/data_structures/graph.hpp File Reference	248
8.6 include/data_structures/hash_table.hpp File Reference	249
8.7 include/data_structures/maxheap.hpp File Reference	249
8.8 include/data_structures/minheap.hpp File Reference	249
8.9 include/data_structures/trie.hpp File Reference	250
8.10 include/display.hpp File Reference	250
8.11 include/GLOBAL.hpp File Reference	250
8.11.1 Typedef Documentation	252
8.11.1.1 json	252
8.11.2 Enumeration Type Documentation	252
8.11.2.1 BUTTON_ACTION	253
8.11.2.2 BUTTON_STATUS	254
8.11.2.3 DATA_STRUCTURES_OPERATOR	254
8.11.2.4 DATA_STRUCTURES_TYPE	256
8.11.2.5 HORIZONTAL_ALIGN	257
8.11.2.6 INPUT_TYPE	257
8.11.2.7 SHAPE	257
8.11.2.8 VERTICAL_ALIGN	258
8.11.2.9 WINDOW_STATUS	258
8.12 include/inputbox.hpp File Reference	259
8.13 include/object.hpp File Reference	259
8.13.1 Typedef Documentation	259
8.13.1.1 json	260
8.14 include/script.hpp File Reference	260
8.15 include/services.hpp File Reference	260
8.15.1 Typedef Documentation	261
8.15.1.1 json	261
8.16 include/sprite.hpp File Reference	262
8.16.1 Typedef Documentation	262
8.16.1.1 json	262
8.17 include/window.hpp File Reference	262
8.18 README.md File Reference	263
8.19 src/AVL/constructor.cpp File Reference	263
8.20 src/btree4th/constructor.cpp File Reference	263
8.21 src/button/constructor.cpp File Reference	263
8.22 src/data_structures/constructor.cpp File Reference	263
8.23 src/display/constructor.cpp File Reference	263
8.24 src/graph/constructor.cpp File Reference	263
8.25 src/hash_table/constructor.cpp File Reference	264
8.26 src/inputbox/constructor.cpp File Reference	264
8.27 src/maxheap/constructor.cpp File Reference	264
8.28 src/minheap/constructor.cpp File Reference	264

8.29 src/object/constructor.cpp File Reference	264
8.30 src/script/constructor.cpp File Reference	264
8.31 src/sprite/constructor.cpp File Reference	265
8.32 src/trie/constructor.cpp File Reference	265
8.33 src/window/constructor.cpp File Reference	265
8.34 src/AVL/destructor.cpp File Reference	265
8.35 src/btree4th/destructor.cpp File Reference	265
8.36 src/button/destructor.cpp File Reference	265
8.37 src/data_structures/destructor.cpp File Reference	265
8.38 src/display/destructor.cpp File Reference	266
8.39 src/graph/destructor.cpp File Reference	266
8.40 src/hash_table/destructor.cpp File Reference	266
8.41 src/inputbox/destructor.cpp File Reference	266
8.42 src/maxheap/destructor.cpp File Reference	266
8.43 src/minheap/destructor.cpp File Reference	266
8.44 src/object/destructor.cpp File Reference	266
8.45 src/script/destructor.cpp File Reference	266
8.46 src/sprite/destructor.cpp File Reference	267
8.47 src/trie/destructor.cpp File Reference	267
8.48 src/window/destructor.cpp File Reference	267
8.49 src/AVL/event.cpp File Reference	267
8.50 src/button/event.cpp File Reference	267
8.51 src/data_structures/event.cpp File Reference	267
8.52 src/display/event.cpp File Reference	267
8.53 src/graph/event.cpp File Reference	268
8.54 src/hash_table/event.cpp File Reference	268
8.55 src/inputbox/event.cpp File Reference	268
8.56 src/minheap/event.cpp File Reference	268
8.57 src/script/event.cpp File Reference	268
8.58 src/trie/event.cpp File Reference	268
8.59 src/window/event.cpp File Reference	268
8.60 src/AVL/operator/delete.cpp File Reference	269
8.61 src/AVL/operator/init.cpp File Reference	269
8.62 src/btree4th/operator/init.cpp File Reference	269
8.63 src/data_structures/operator/AVL/init.cpp File Reference	269
8.64 src/data_structures/operator/graph/init.cpp File Reference	269
8.65 src/data_structures/operator/hash_table/init.cpp File Reference	269
8.66 src/data_structures/operator/minheap/init.cpp File Reference	269
8.67 src/data_structures/operator/trie/init.cpp File Reference	269
8.68 src/graph/operator/init.cpp File Reference	270
8.68.1 Function Documentation	270
8.68.1.1 sqr()	270

8.69 src/hash_table/operator/init.cpp File Reference	270
8.70 src/maxheap/operator/init.cpp File Reference	270
8.71 src/minheap/operator/init.cpp File Reference	270
8.72 src/trie/operator/init.cpp File Reference	270
8.73 src/AVL/operator/insert.cpp File Reference	271
8.74 src/btree4th/operator/insert.cpp File Reference	271
8.75 src/data_structures/operator/AVL/insert.cpp File Reference	271
8.76 src/data_structures/operator/hash_table/insert.cpp File Reference	271
8.77 src/data_structures/operator/minheap/insert.cpp File Reference	271
8.78 src/data_structures/operator/trie/insert.cpp File Reference	271
8.79 src/hash_table/operator/insert.cpp File Reference	271
8.80 src/maxheap/operator/insert.cpp File Reference	271
8.81 src/minheap/operator/insert.cpp File Reference	272
8.82 src/trie/operator/insert.cpp File Reference	272
8.83 src/AVL/operator/search.cpp File Reference	272
8.84 src/btree4th/operator/search.cpp File Reference	272
8.85 src/data_structures/operator/AVL/search.cpp File Reference	272
8.86 src/data_structures/operator/hash_table/search.cpp File Reference	272
8.87 src/data_structures/operator/trie/search.cpp File Reference	272
8.88 src/hash_table/operator/search.cpp File Reference	272
8.89 src/trie/operator/search.cpp File Reference	273
8.90 src/AVL/rendering.cpp File Reference	273
8.91 src/button/rendering.cpp File Reference	273
8.92 src/data_structures/rendering.cpp File Reference	273
8.93 src/display/rendering.cpp File Reference	273
8.94 src/graph/rendering.cpp File Reference	273
8.95 src/hash_table/rendering.cpp File Reference	273
8.96 src/inputbox/rendering.cpp File Reference	274
8.97 src/maxheap/rendering.cpp File Reference	274
8.98 src/minheap/rendering.cpp File Reference	274
8.99 src/object/rendering.cpp File Reference	274
8.100 src/script/rendering.cpp File Reference	274
8.101 src/sprite/rendering.cpp File Reference	274
8.102 src/trie/rendering.cpp File Reference	274
8.103 src/window/rendering.cpp File Reference	274
8.104 src/AVL/rotate.cpp File Reference	275
8.105 src/AVL/step.cpp File Reference	275
8.106 src/data_structures/step.cpp File Reference	275
8.107 src/hash_table/step.cpp File Reference	275
8.108 src/minheap/step.cpp File Reference	275
8.109 src/trie/step.cpp File Reference	275
8.110 src/btree4th/node.cpp File Reference	275

8.111 src/btree4th/operator/addRecord.cpp File Reference	276
8.112 src/btree4th/operator/split.cpp File Reference	276
8.113 src/button/action.cpp File Reference	276
8.114 src/button/mouse_action.cpp File Reference	276
8.115 src/data_structures/AVL.cpp File Reference	276
8.116 src/data_structures/btree4th.cpp File Reference	276
8.117 src/data_structures/graph.cpp File Reference	276
8.118 src/data_structures/hash_table.cpp File Reference	276
8.119 src/data_structures/heap.cpp File Reference	276
8.120 src/data_structures/operator.cpp File Reference	276
8.121 src/inputbox/operator.cpp File Reference	277
8.122 src/data_structures/operator/AVL/remove.cpp File Reference	277
8.123 src/data_structures/operator/hash_table/remove.cpp File Reference	277
8.124 src/data_structures/operator/trie/remove.cpp File Reference	277
8.125 src/hash_table/operator/remove.cpp File Reference	277
8.126 src/maxheap/operator/remove.cpp File Reference	277
8.127 src/minheap/operator/remove.cpp File Reference	277
8.127.1 Variable Documentation	278
8.127.1.1 valuesReachedZero	278
8.128 src/trie/operator/remove.cpp File Reference	278
8.129 src/data_structures/operator/AVL/setting.cpp File Reference	278
8.130 src/data_structures/operator/graph/setting.cpp File Reference	278
8.131 src/data_structures/operator/hash_table/setting.cpp File Reference	278
8.132 src/data_structures/operator/minheap/setting.cpp File Reference	278
8.133 src/data_structures/operator/trie/setting.cpp File Reference	279
8.134 src/data_structures/operator/graph/dijkstra.cpp File Reference	279
8.135 src/graph/operator/dijkstra.cpp File Reference	279
8.136 src/data_structures/operator/graph/mst.cpp File Reference	279
8.137 src/graph/operator/mst.cpp File Reference	279
8.138 src/data_structures/operator/graph/scc.cpp File Reference	279
8.139 src/graph/operator/scc.cpp File Reference	280
8.140 src/data_structures/operator/minheap/pop.cpp File Reference	280
8.141 src/data_structures/operator/minheap/size.cpp File Reference	280
8.142 src/maxheap/operator/size.cpp File Reference	280
8.143 src/minheap/operator/size.cpp File Reference	280
8.144 src/data_structures/operator/minheap/top.cpp File Reference	280
8.145 src/data_structures/trie.cpp File Reference	280
8.146 src/graph/operator/repair.cpp File Reference	280
8.147 src/inputbox/typing.cpp File Reference	280
8.148 src/main.cpp File Reference	281
8.148.1 Function Documentation	281
8.148.1.1 main()	281

8.149 src/maxheap/operator/getmax.cpp File Reference	281
8.150 src/maxheap/operator/heapify.cpp File Reference	281
8.151 src/minheap/operator/heapify.cpp File Reference	281
8.152 src/minheap/operator/getmin.cpp File Reference	281
8.153 src/object/coloring.cpp File Reference	282
8.154 src/sprite/coloring.cpp File Reference	282
8.155 src/object/cropping.cpp File Reference	282
8.156 src/object/external_storage.cpp File Reference	282
8.157 src/object/font.cpp File Reference	282
8.158 src/object/locating.cpp File Reference	282
8.159 src/sprite/locating.cpp File Reference	282
8.160 src/object/shape.cpp File Reference	283
8.161 src/object/visible.cpp File Reference	283
8.162 src/script/highlight.cpp File Reference	283
8.163 src/services/file.cpp File Reference	283
8.164 src/services/json.cpp File Reference	283
8.165 src/services/number.cpp File Reference	283
8.165.1 Function Documentation	284
8.165.1.1 removeLeadingZero()	284
8.166 src/services/random.cpp File Reference	284
8.167 src/services/string.cpp File Reference	284
8.168 src/sprite/aligning.cpp File Reference	284
8.169 src/sprite/textbox.cpp File Reference	284
8.170 src/window/running.cpp File Reference	285
8.171 src/window/status.cpp File Reference	285
8.172 src/window/updating.cpp File Reference	285
Index	287

Chapter 1

VISUALIZER

- This is my CS163 solo project that is used to visualize data structures.

1.1 Dependency

- C++17
- SDL2 (SDL2_image, SDL2_ttf)
- <https://github.com/nlohmann/json>
- CMake
- Makefile

1.2 Demonstrate

1.3 Features

1.3.1 Main features

- [x] Main menu
- [x] Option menu
- [x] Settings
- [] Code highlight
 - [x] Hash table
 - [x] AVL
 - [] 234 tree
 - [x] Heap
 - [x] Trie
 - [] graph

1.3.2 Data structures

- [x] [AVL](#)
 - [x] Initial, init from file
 - [x] Insert a value
 - [x] Delete a value
 - [x] Find a value
- [x] [Trie](#)
 - [x] Initial, init from file
 - [x] Insert a value
 - [x] Delete a value
 - [x] Find a value
- [x] Hash table
 - [x] Initial, init from file
 - [x] Insert a value
 - [x] Delete a value
 - [x] Find a value
- [x] Heap
 - [x] Initial, init from file
 - [x] Insert a value
 - [x] Remove the largest/smallest value
 - [x] Get the largest/smallest value
 - [x] Get the size of heap
- [x] [Graph](#)
 - [x] Init from matrix, file
 - [x] Connected components
 - [x] Minimum spanning tree
 - [x] Dijkstra
- [] 234 tree
 - [] Initial, init from file
 - [] Insert a value
 - [] Delete a value
 - [] Find a value

1.4 Documentation

- [Report](#)
- [Demonstrate video](#)

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

DISPLAY		
	Name of display	13
FILEE		
	Interact with text files	14
JSON		
	Interact with *.json files	14
NUMBER		
	Convert between string and interger	15
PATH		
	Path to assets, atributes, and saving files	19
PATH::ASSETS		20
PATH::ATB		21
PATH::SAVING		23
RANDOM		
	Random intergers, doubles, strings generator	24
SIUSTRING		
	Features for std::string	27

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

AVL	29
BTree4th	45
distanceHeap	127
DSU	130
Graph	134
HashTable	142
maxHeap	163
minHeap	166
MyWindow	177
Point	206
Position	206
Object	188
Button	51
DataStructures	73
Display	108
InputBox	155
Script	208
Sprite	213
Trie	236

Chapter 4

Class Index

4.1 Class List

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

AVL		
	AVL class	29
BTree4th		45
Button		
	Button class that interact with user input	51
DataStructures		
	Container that contains all data structures	73
Display		
	Container of button intermediate between button and user input, window	108
distanceHeap		127
DSU		130
Graph		
	Graph class	134
HashTable		
	HashTable class	142
InputBox		
	Register for user keyboard input	155
maxHeap		163
minHeap		
	Heap class	166
MyWindow		
	Window class class that create a window and manage it	177
Object		
	Class that represent shape, image from files, text. Smallest drawable unit	188
Point		206
Position		
	Location of an object in 2D coordinate	206
Script		
	Container that contains a pseudo-code	208
Sprite		
	Object container Drawable	213
Trie		
	Trie data structure	236

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

include/button.hpp	247
include/data_structures.hpp	247
include/display.hpp	250
include/GLOBAL.hpp	250
include/inputbox.hpp	259
include/object.hpp	259
include/script.hpp	260
include/services.hpp	260
include/sprite.hpp	262
include/window.hpp	262
include/data_structures/AVL.hpp	248
include/data_structures/btree4th.hpp	248
include/data_structures/graph.hpp	248
include/data_structures/hash_table.hpp	249
include/data_structures/maxheap.hpp	249
include/data_structures/minheap.hpp	249
include/data_structures/trie.hpp	250
src/main.cpp	281
src/AVL/constructor.cpp	263
src/AVL/destructor.cpp	265
src/AVL/event.cpp	267
src/AVL/rendering.cpp	273
src/AVL/rotate.cpp	275
src/AVL/step.cpp	275
src/AVL/operator/delete.cpp	269
src/AVL/operator/init.cpp	269
src/AVL/operator/insert.cpp	271
src/AVL/operator/search.cpp	272
src/btree4th/constructor.cpp	263
src/btree4th/destructor.cpp	265
src/btree4th/node.cpp	275
src/btree4th/operator/addRecord.cpp	276
src/btree4th/operator/init.cpp	269
src/btree4th/operator/insert.cpp	271
src/btree4th/operator/search.cpp	272

src/btree4th/operator/split.cpp	276
src/button/action.cpp	276
src/button/constructor.cpp	263
src/button/destructor.cpp	265
src/button/event.cpp	267
src/button/mouse_action.cpp	276
src/button/rendering.cpp	273
src/data_structures/AVL.cpp	276
src/data_structures/btree4th.cpp	276
src/data_structures/constructor.cpp	263
src/data_structures/destructor.cpp	265
src/data_structures/event.cpp	267
src/data_structures/graph.cpp	276
src/data_structures/hash_table.cpp	276
src/data_structures/heap.cpp	276
src/data_structures/operator.cpp	276
src/data_structures/rendering.cpp	273
src/data_structures/step.cpp	275
src/data_structures/trie.cpp	280
src/data_structures/operator/AVL/init.cpp	269
src/data_structures/operator/AVL/insert.cpp	271
src/data_structures/operator/AVL/remove.cpp	277
src/data_structures/operator/AVL/search.cpp	272
src/data_structures/operator/AVL/setting.cpp	278
src/data_structures/operator/graph/dijkstra.cpp	279
src/data_structures/operator/graph/init.cpp	269
src/data_structures/operator/graph/mst.cpp	279
src/data_structures/operator/graph/scc.cpp	279
src/data_structures/operator/graph/setting.cpp	278
src/data_structures/operator/hash_table/init.cpp	269
src/data_structures/operator/hash_table/insert.cpp	271
src/data_structures/operator/hash_table/remove.cpp	277
src/data_structures/operator/hash_table/search.cpp	272
src/data_structures/operator/hash_table/setting.cpp	278
src/data_structures/operator/minheap/init.cpp	269
src/data_structures/operator/minheap/insert.cpp	271
src/data_structures/operator/minheap/pop.cpp	280
src/data_structures/operator/minheap/setting.cpp	278
src/data_structures/operator/minheap/size.cpp	280
src/data_structures/operator/minheap/top.cpp	280
src/data_structures/operator/trie/init.cpp	269
src/data_structures/operator/trie/insert.cpp	271
src/data_structures/operator/trie/remove.cpp	277
src/data_structures/operator/trie/search.cpp	272
src/data_structures/operator/trie/setting.cpp	279
src/display/constructor.cpp	263
src/display/destructor.cpp	266
src/display/event.cpp	267
src/display/rendering.cpp	273
src/graph/constructor.cpp	263
src/graph/destructor.cpp	266
src/graph/event.cpp	268
src/graph/rendering.cpp	273
src/graph/operator/dijkstra.cpp	279
src/graph/operator/init.cpp	270
src/graph/operator/mst.cpp	279
src/graph/operator/repair.cpp	280
src/graph/operator/scc.cpp	280

src/hash_table/constructor.cpp	264
src/hash_table/destructor.cpp	266
src/hash_table/event.cpp	268
src/hash_table/rendering.cpp	273
src/hash_table/step.cpp	275
src/hash_table/operator/init.cpp	270
src/hash_table/operator/insert.cpp	271
src/hash_table/operator/remove.cpp	277
src/hash_table/operator/search.cpp	272
src/inputbox/constructor.cpp	264
src/inputbox/destructor.cpp	266
src/inputbox/event.cpp	268
src/inputbox/operator.cpp	277
src/inputbox/rendering.cpp	274
src/inputbox/typing.cpp	280
src/maxheap/constructor.cpp	264
src/maxheap/destructor.cpp	266
src/maxheap/rendering.cpp	274
src/maxheap/operator/getmax.cpp	281
src/maxheap/operator/heapify.cpp	281
src/maxheap/operator/init.cpp	270
src/maxheap/operator/insert.cpp	271
src/maxheap/operator/remove.cpp	277
src/maxheap/operator/size.cpp	280
src/minheap/constructor.cpp	264
src/minheap/destructor.cpp	266
src/minheap/event.cpp	268
src/minheap/rendering.cpp	274
src/minheap/step.cpp	275
src/minheap/operator/getmin.cpp	281
src/minheap/operator/heapify.cpp	281
src/minheap/operator/init.cpp	270
src/minheap/operator/insert.cpp	272
src/minheap/operator/remove.cpp	277
src/minheap/operator/size.cpp	280
src/object/coloring.cpp	282
src/object/constructor.cpp	264
src/object/cropping.cpp	282
src/object/destructor.cpp	266
src/object/external_storage.cpp	282
src/object/font.cpp	282
src/object/locating.cpp	282
src/object/rendering.cpp	274
src/object/shape.cpp	283
src/object/visible.cpp	283
src/script/constructor.cpp	264
src/script/destructor.cpp	266
src/script/event.cpp	268
src/script/highlight.cpp	283
src/script/rendering.cpp	274
src/services/file.cpp	283
src/services/json.cpp	283
src/services/number.cpp	283
src/services/random.cpp	284
src/services/string.cpp	284
src/sprite/aligning.cpp	284
src/sprite/coloring.cpp	282
src/sprite/constructor.cpp	265

src/sprite/destructor.cpp	267
src/sprite/locating.cpp	282
src/sprite/rendering.cpp	274
src/sprite/textbox.cpp	284
src/trie/constructor.cpp	265
src/trie/destructor.cpp	267
src/trie/event.cpp	268
src/trie/rendering.cpp	274
src/trie/step.cpp	275
src/trie/operator/init.cpp	270
src/trie/operator/insert.cpp	272
src/trie/operator/remove.cpp	278
src/trie/operator/search.cpp	273
src/window/constructor.cpp	265
src/window/destructor.cpp	267
src/window/event.cpp	268
src/window/rendering.cpp	274
src/window/running.cpp	285
src/window/status.cpp	285
src/window/updating.cpp	285

Chapter 6

Namespace Documentation

6.1 DISPLAY Namespace Reference

Name of display.

Variables

- `const std::string HOME_ = "home"`
- `const std::string WORKING_ = "working"`

6.1.1 Detailed Description

Name of display.

6.1.2 Variable Documentation

6.1.2.1 HOME_

```
const std::string DISPLAY::HOME_ = "home"
```

Definition at line 152 of file GLOBAL.hpp.

6.1.2.2 WORKING_

```
const std::string DISPLAY::WORKING_ = "working"
```

Definition at line 153 of file GLOBAL.hpp.

6.2 FILEE Namespace Reference

Interact with text files.

Functions

- `std::vector< std::string > readFile (std::string path)`

6.2.1 Detailed Description

Interact with text files.

6.2.2 Function Documentation

6.2.2.1 readFile()

```
std::vector< std::string > FILEE::readFile (  
    std::string path )
```

Definition at line 5 of file file.cpp.

```
6 {  
7     std::vector<std::string> result;  
8  
9     std::ifstream fin(path);  
10  
11     std::string line;  
12  
13     while(std::getline(fin, line))  
14     {  
15         result.push_back(line);  
16     }  
17  
18     fin.close();  
19  
20     return result;  
21 }
```

6.3 JSON Namespace Reference

Interact with *.json files.

Functions

- `json * readFile (std::string path)`
- `void saveFile (std::string path, json *data)`

6.3.1 Detailed Description

Interact with *.json files.

6.3.2 Function Documentation

6.3.2.1 readFile()

```
json * JSON::readFile (
    std::string path )
```

Definition at line 5 of file json.cpp.

```
6 {
7     json* mem = new json();
8     std::ifstream fin(path);
9
10    fin >> *mem;
11
12    fin.close();
13
14    return mem;
15 }
```

6.3.2.2 saveFile()

```
void JSON::saveFile (
    std::string path,
    json * data )
```

Definition at line 17 of file json.cpp.

```
18 {
19     std::ofstream fout(path);
20
21     fout << data->dump(4);
22
23     fout.close();
24
25     return ;
26 }
```

6.4 NUMBER Namespace Reference

Convert between string and interger.

Functions

- int64_t [stringToInt](#) (std::string s)
- std::string [intToString](#) (int64_t n)
- std::vector< int > [stringToArray](#) (std::string s)
- bool [isDigit](#) (char c)
- bool [isLetter](#) (char c)
- bool [isSymbol](#) (char c)
- bool [isSign](#) (char c)
- bool [isOperator](#) (char c)
- std::string [removeLeadingZero](#) (std::string s)
- bool [isNumber](#) (std::string s)
- bool [isInInterval](#) (std::string s, int64_t a, int64_t b)

Variables

- `const int64_t INF = LLONG_MAX`

6.4.1 Detailed Description

Convert between string and interger.

6.4.2 Function Documentation

6.4.2.1 `intToString()`

```
std::string NUMBER::intToString (
    int64_t n )
```

Definition at line 4 of file number.cpp.

```
5 {
6     if(n == 0) return "0";
7     std::string result = "";
8     bool negative = false;
9     if(n < 0)
10    {
11        negative = true;
12        n *= -1;
13    }
14
15    while(n)
16    {
17        result = (char) (n % 10 + '0') + result;
18        n /= 10;
19    }
20
21    return result;
22 }
```

6.4.2.2 `isDigit()`

```
bool NUMBER::isDigit (
    char c )
```

Definition at line 67 of file number.cpp.

```
68 {
69     if((int) c < 48 || (int) c > 57) return false;
70     return true;
71 }
```

6.4.2.3 isInterval()

```
bool NUMBER::isInterval (
    std::string s,
    int64_t a,
    int64_t b )
```

Definition at line 106 of file number.cpp.

```
107 {
108     if (!NUMBER::isNumber(s)) return false;
109
110     int64_t n = stringToInt(s);
111
112     return (n >= a && n <= b);
113 }
```

6.4.2.4 isLetter()

```
bool NUMBER::isLetter (
    char c )
```

Definition at line 73 of file number.cpp.

```
74 {
75     return ((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z'));
76 }
```

6.4.2.5 isNumber()

```
bool NUMBER::isNumber (
    std::string s )
```

Definition at line 91 of file number.cpp.

```
92 {
93     if ((int) s.size() == 0)
94         return false;
95     if ((int) s.size() == 1)
96         return isDigit(s[0]);
97     if (!NUMBER::isSign(s[0]) && !NUMBER::isDigit(s[0]))
98         return false;
99
100     for (int i = 1; i < (int) s.size(); i++)
101         if (!NUMBER::isDigit(s[i])) return false;
102
103     return true;
104 }
```

6.4.2.6 isOperator()

```
bool NUMBER::isOperator (
    char c )
```

Definition at line 82 of file number.cpp.

```
83 {
84     return (c == '+' || c == '-' || c == '*' || c == '/');
85 }
```

6.4.2.7 isSign()

```
bool NUMBER::isSign (
    char c )
```

Definition at line 86 of file number.cpp.

```
87 {
88     return (c == '+' || c == '-');
89 }
```

6.4.2.8 isSymbol()

```
bool NUMBER::isSymbol (
    char c )
```

Definition at line 77 of file number.cpp.

```
78 {
79     return (c == '.' || c == ',' || c == '!' || c == '?' || c == ':' || c == ';' || c == '(' || c == ')'
    || c == '[' || c == ']' || c == '{' || c == '}' || c == '"' || c == '\');
80 }
```

6.4.2.9 removeLeadingZero()

```
std::string NUMBER::removeLeadingZero (
    std::string s )
```

6.4.2.10 stringToArray()

```
std::vector< int > NUMBER::stringToArray (
    std::string s )
```

Definition at line 24 of file number.cpp.

```
25 {
26     std::vector<int> result;
27     int i = 0;
28
29     while(i != (int) s.size())
30     {
31         while(i != (int) s.size() && !NUMBER::isDigit(s[i])) i++;
32         if(i == (int) s.size()) break;
33
34         int n = 0;
35         while(i != (int) s.size() && NUMBER::isDigit(s[i])) n = n * 10 + s[i++] - '0';
36         result.push_back(n);
37     }
38     return result;
39 }
```

6.4.2.11 stringToInt()

```
int64_t NUMBER::stringToInt (
    std::string s )
```

Definition at line 41 of file number.cpp.

```
42 {
43     if(!NUMBER::isNumber(s)) return NUMBER::INF;
44     if((int) s.size() > 18) return NUMBER::INF;
45     if((int) s.size() == 1) return (int64_t) (s[0] - '0');
46
47     int64_t n = 0;
48     bool negative = false;
49     bool sign = false;
50     if(NUMBER::isSign(s[0]))
51     {
52         sign = true;
53         negative = (s[0] == '-');
54     }
55
56     for(int i = sign; i < (int) s.size(); i++)
57     {
58         n *= 10;
59         n += (int64_t) (s[i] - '0');
60     }
61
62     if(negative) n *= -1;
63
64     return n;
65 }
```

6.4.3 Variable Documentation

6.4.3.1 INF

```
const int64_t NUMBER::INF = LLONG_MAX
```

Definition at line 36 of file services.hpp.

6.5 PATH Namespace Reference

Path to assets, attributes, and saving files.

Namespaces

- [ASSETS](#)
- [ATB](#)
- [SAVING](#)

Variables

- const std::string [ASSETS_](#) = "assets/"
- const std::string [ATTRIBUTE_](#) = "atb/"
- const std::string [SAVING_](#) = "saving/"

6.5.1 Detailed Description

Path to assets, attributes, and saving files.

6.5.2 Variable Documentation

6.5.2.1 ASSETS_

```
const std::string PATH::ASSETS_ = "assets/"
```

Definition at line 161 of file GLOBAL.hpp.

6.5.2.2 ATTRIBUTE_

```
const std::string PATH::ATTRIBUTE_ = "atb/"
```

Definition at line 168 of file GLOBAL.hpp.

6.5.2.3 SAVING_

```
const std::string PATH::SAVING_ = "saving/"
```

Definition at line 180 of file GLOBAL.hpp.

6.6 PATH::ASSETS Namespace Reference

Variables

- const std::string [GRAPHICS_](#) = "assets/graphics/"
- const std::string [FONTS_](#) = "assets/fonts/"
- const std::string [SCRIPT_](#) = "assets/script/"

6.6.1 Variable Documentation

6.6.1.1 FONTS_

```
const std::string PATH::ASSETS::FONTS_ = "assets/fonts/"
```

Definition at line 165 of file GLOBAL.hpp.

6.6.1.2 GRAPHICS_

```
const std::string PATH::ASSETS::GRAPHICS_ = "assets/graphics/"
```

Definition at line 164 of file GLOBAL.hpp.

6.6.1.3 SCRIPT_

```
const std::string PATH::ASSETS::SCRIPT_ = "assets/script/"
```

Definition at line 166 of file GLOBAL.hpp.

6.7 PATH::ATB Namespace Reference

Variables

- const std::string [SPRITE_](#) = "atb/sprite/"
- const std::string [OBJECT_](#) = "atb/object/"
- const std::string [DISPLAY_](#) = "atb/display/"
- const std::string [BUTTON_](#) = "atb/button/"
- const std::string [DATA_STRUCTURES_](#) = "atb/data_structures/"
- const std::string [INPUTBOX_](#) = "atb/input/"
- const std::string [SCRIPT_](#) = "atb/script/"

6.7.1 Variable Documentation

6.7.1.1 BUTTON_

```
const std::string PATH::ATB::BUTTON_ = "atb/button/"
```

Definition at line 174 of file GLOBAL.hpp.

6.7.1.2 DATA_STRUCTURES_

```
const std::string PATH::ATB::DATA_STRUCTURES_ = "atb/data_structures/"
```

Definition at line 175 of file GLOBAL.hpp.

6.7.1.3 DISPLAY_

```
const std::string PATH::ATB::DISPLAY_ = "atb/display/"
```

Definition at line 173 of file GLOBAL.hpp.

6.7.1.4 INPUTBOX_

```
const std::string PATH::ATB::INPUTBOX_ = "atb/input/"
```

Definition at line 176 of file GLOBAL.hpp.

6.7.1.5 OBJECT_

```
const std::string PATH::ATB::OBJECT_ = "atb/object/"
```

Definition at line 172 of file GLOBAL.hpp.

6.7.1.6 SCRIPT_

```
const std::string PATH::ATB::SCRIPT_ = "atb/script/"
```

Definition at line 177 of file GLOBAL.hpp.

6.7.1.7 SPRITE_

```
const std::string PATH::ATB::SPRITE_ = "atb/sprite/"
```

Definition at line 171 of file GLOBAL.hpp.

6.8 PATH::SAVING Namespace Reference

Variables

- `const std::string AVL_ = "saving/AVL.txt"`
- `const std::string HASH_TABLE_ = "saving/HASH_TABLE.txt"`
- `const std::string GRAPH_ = "saving/GRAPH.txt"`
- `const std::string TRIE_ = "saving/TRIE.txt"`
- `const std::string MIN_HEAP_ = "saving/MIN_HEAP.txt"`
- `const std::string MAX_HEAP_ = "saving/MAX_HEAP.txt"`
- `const std::string BTREE_4TH_ = "saving/BTREE_4TH.txt"`

6.8.1 Variable Documentation

6.8.1.1 AVL_

```
const std::string PATH::SAVING::AVL_ = "saving/AVL.txt"
```

Definition at line 183 of file GLOBAL.hpp.

6.8.1.2 BTREE_4TH_

```
const std::string PATH::SAVING::BTREE_4TH_ = "saving/BTREE_4TH.txt"
```

Definition at line 189 of file GLOBAL.hpp.

6.8.1.3 GRAPH_

```
const std::string PATH::SAVING::GRAPH_ = "saving/GRAPH.txt"
```

Definition at line 185 of file GLOBAL.hpp.

6.8.1.4 HASH_TABLE_

```
const std::string PATH::SAVING::HASH_TABLE_ = "saving/HASH_TABLE.txt"
```

Definition at line 184 of file GLOBAL.hpp.

6.8.1.5 MAX_HEAP_

```
const std::string PATH::SAVING::MAX_HEAP_ = "saving/MAX_HEAP.txt"
```

Definition at line 188 of file GLOBAL.hpp.

6.8.1.6 MIN_HEAP_

```
const std::string PATH::SAVING::MIN_HEAP_ = "saving/MIN_HEAP.txt"
```

Definition at line 187 of file GLOBAL.hpp.

6.8.1.7 TRIE_

```
const std::string PATH::SAVING::TRIE_ = "saving/TRIE.txt"
```

Definition at line 186 of file GLOBAL.hpp.

6.9 RANDOM Namespace Reference

Random intergers, doubles, strings generator.

Functions

- int [getInt](#) (int a, int b)
- std::string [getInt](#) (int length, int a, int b)
- long long [getLongLong](#) (long long a, long long b)
- float [getFloat](#) (float a, float b)
- double [getDouble](#) (double a, double b)
- char [getChar](#) (char a, char b)
- char [getChar](#) ()
- std::string [getString](#) (int length)
- std::string [getString](#) (int length, char a, char b)
- bool [flipCoin](#) ()

Variables

- std::mt19937 [rng](#) = std::mt19937(std::chrono::steady_clock::now().time_since_epoch().count())

6.9.1 Detailed Description

Random intergers, doubles, strings generator.

6.9.2 Function Documentation

6.9.2.1 flipCoin()

```
bool RANDOM::flipCoin ( )
```

Definition at line 60 of file random.cpp.

```
61 {  
62     return getInt(1, 2) - 1;  
63 }
```

6.9.2.2 getChar() [1/2]

```
char RANDOM::getChar ( )
```

Definition at line 39 of file random.cpp.

```
40 {  
41     return getChar('a', 'z');  
42 }
```

6.9.2.3 getChar() [2/2]

```
char RANDOM::getChar (  
    char a,  
    char b )
```

Definition at line 34 of file random.cpp.

```
35 {  
36     return std::uniform_int_distribution<char> (a, b)(rng);  
37 }
```

6.9.2.4 getDouble()

```
double RANDOM::getDouble (  
    double a,  
    double b )
```

Definition at line 29 of file random.cpp.

```
30 {  
31     return std::uniform_real_distribution<double> (a, b)(rng);  
32 }
```

6.9.2.5 getFloat()

```
float RANDOM::getFloat (
    float a,
    float b )
```

Definition at line 24 of file random.cpp.

```
25 {
26     return std::uniform_real_distribution<float> (a, b) (rng);
27 }
```

6.9.2.6 getInt() [1/2]

```
int RANDOM::getInt (
    int a,
    int b )
```

Definition at line 5 of file random.cpp.

```
6 {
7     return std::uniform_int_distribution<int> (a, b) (rng);
8 }
```

6.9.2.7 getInt() [2/2]

```
std::string RANDOM::getInt (
    int length,
    int a,
    int b )
```

Definition at line 10 of file random.cpp.

```
11 {
12     if (length == 0) return "";
13     std::string result = NUMBER::intToString(RANDOM::getInt(a, b));
14
15     while (--length) result = result + " " + NUMBER::intToString(RANDOM::getInt(a, b));
16     return result;
17 }
```

6.9.2.8 getLongLong()

```
long long RANDOM::getLongLong (
    long long a,
    long long b )
```

Definition at line 19 of file random.cpp.

```
20 {
21     return std::uniform_int_distribution<long long> (a, b) (rng);
22 }
```

6.9.2.9 getString() [1/2]

```
std::string RANDOM::getString (
    int length )
```

Definition at line 44 of file random.cpp.

```
45 {
46     std::string s = "";
47     for (int i = 0; i < length; i++)
48         s += getChar();
49     return s;
50 }
```

6.9.2.10 getString() [2/2]

```
std::string RANDOM::getString (
    int length,
    char a,
    char b )
```

Definition at line 52 of file random.cpp.

```
53 {
54     std::string s = "";
55     for (int i = 0; i < length; i++)
56         s += getChar(a, b);
57     return s;
58 }
```

6.9.3 Variable Documentation

6.9.3.1 rng

```
std::mt19937 RANDOM::rng = std::mt19937(std::chrono::steady_clock::now().time_since_epoch().count())
[extern]
```

Definition at line 3 of file random.cpp.

6.10 SIUSTRING Namespace Reference

Features for std::string.

Functions

- bool [isSeparator](#) (char c)
- std::vector< std::string > [split](#) (std::string s)

6.10.1 Detailed Description

Features for `std::string`.

6.10.2 Function Documentation

6.10.2.1 `isSeparator()`

```
bool SIUSTRING::isSeparator (  
    char c )
```

Definition at line 3 of file `string.cpp`.

```
3      {  
4      return (c == ' ' || c == '\t' || c == '\n' || c == '\r' || c == ',');  
5  }
```

6.10.2.2 `split()`

```
std::vector< std::string > SIUSTRING::split (  
    std::string s )
```

Definition at line 7 of file `string.cpp`.

```
8  {  
9      std::vector<std::string> result;  
10  
11      int i = 0;  
12      while (i < s.length())  
13      {  
14          while(i < s.length() && isSeparator(s[i])) i++;  
15          if (i >= s.length()) break;  
16          result.push_back("");  
17          while(i < s.length() && !isSeparator(s[i])) result.back() += s[i++];  
18      }  
19  
20      return result;  
21 }
```

Chapter 7

Class Documentation

7.1 AVL Class Reference

AVL class.

```
#include <AVL.hpp>
```

Public Member Functions

- [AVL](#) (SDL_Renderer *render, std::mutex &m, TTF_Font *f, SDL_Rect v, int cap)
- [~AVL](#) ()
- void [init](#) (std::vector< int > v)
- bool [insert](#) (int key)
- bool [remove](#) (int key)
- bool [search](#) (int key)
- int [maxDepth](#) ()
- void [setEdgesColor](#) (SDL_Color c)
- void [setNodeColor](#) (SDL_Color bg, SDL_Color fg)
- void [goOff](#) ()
- void [goOn](#) ()
- void [goNext](#) ()
- void [goBack](#) ()
- void [speedUp](#) ()
- void [slowDown](#) ()
- void [closeScript](#) ()
- bool [isReceiveEvent](#) (SDL_Event &e)
- [Button](#) * [react](#) (SDL_Event &e)
- void [rendering](#) ()
- void [setting](#) (SDL_Color c1, SDL_Color c2, SDL_Color c3, SDL_Color c4)

Protected Member Functions

- Node * [rotateLeft](#) (Node *node)
- Node * [rotateRight](#) (Node *node)
- int [getHigh](#) (Node *node)
- int [balanceFactor](#) (Node *node)
- Node * [balancing](#) (Node *node)
- int [maxDepth](#) (Node *node)
- Node * [insert](#) (Node *node, Node *newNode)
- Node * [unplugSmallest](#) (Node *Node, struct Node *&n2)
- Node * [remove](#) (Node *node, int key)
- void [search](#) (Node *node, int key)
- int [locating](#) (Node *node, int shiftDown, int shiftRight)
- void [renderLine](#) (Node *src, Node *dst)
- void [waitForStep](#) ()
- void [highlight](#) (std::vector< int > l)
- void [unhighlight](#) (std::vector< int > l)

7.1.1 Detailed Description

[AVL](#) class.

Drawable [AVL](#) tree.

Definition at line 18 of file AVL.hpp.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 AVL()

```
AVL::AVL (
    SDL_Renderer * render,
    std::mutex & m,
    TTF_Font * f,
    SDL_Rect v,
    int cap )
```

Definition at line 23 of file constructor.cpp.

```
23                                     : ds_mutex(m)
24 {
25     root = nullptr;
26     render = rend;
27     font = f;
28     viewport = vp;
29     capacity = cap;
30     sizeOfTree = 0;
31     isQueue = false;
32     isPause = false;
33     edgesColor = {255, 255, 255, 255};
34     fontColor = {255, 255, 255, 255};
35     nodeColor = {20, 85, 185, 255};
36     shiftX = 20;
37     shiftY = 20;
38     distanceX = 40;
39     distanceY = 100;
40     isMoving = false;
```



```

41     stepWait = 600;
42     isAnimate = false;
43
44     nodeColor = {20, 85, 185, 255};
45     fontColor = {255, 255, 255, 255};
46     bgColor = {0, 0, 0, 255};
47
48     std::string fontpath = PATH::ASSETS::FONTS_ + "nimbus-sans-l/regular.otf";
49     scriptFont = TTF_OpenFont(fontpath.c_str(), 18);
50
51     currentScript = nullptr;
52     Script* insert = new Script(render, scriptFont);
53     insert->linking("AVL/insert");
54     scripts[DATA_STRUCTURES_OPERATOR::INSERT] = insert;
55
56     Script* remove = new Script(render, scriptFont);
57     remove->linking("AVL/remove");
58     scripts[DATA_STRUCTURES_OPERATOR::DELETE] = remove;
59
60     Script* search = new Script(render, scriptFont);
61     search->linking("AVL/search");
62     scripts[DATA_STRUCTURES_OPERATOR::SEARCH] = search;
63
64     Script* init = new Script(render, scriptFont);
65     init->linking("AVL/init");
66     scripts[DATA_STRUCTURES_OPERATOR::INIT] = init;
67
68     currentScript = insert;
69 }

```

7.1.2.2 ~AVL()

AVL::~~AVL ()

Definition at line 9 of file destructor.cpp.

```

10 {
11     if (root != nullptr) delete root;
12     if (scriptFont != nullptr) TTF_CloseFont (scriptFont);
13     for (auto& script : scripts)
14     {
15         if (script.second != nullptr) delete script.second;
16     }
17     scripts.clear();
18 }

```

7.1.3 Member Function Documentation

7.1.3.1 balanceFactor()

```

int AVL::balanceFactor (
    Node * node ) [protected]

```

Definition at line 9 of file rotate.cpp.

```

10 {
11     if (node == nullptr) return 0;
12     return getHigh (node->lson) - getHigh (node->rson);
13 }

```

7.1.3.2 balancing()

```
AVL::Node * AVL::balancing (
    Node * node ) [protected]
```

Definition at line 79 of file rotate.cpp.

```
80 {
81     if (node == nullptr)
82     {
83         return nullptr;
84     }
85     if (node->lson != nullptr) node->lson->repair();
86     if (node->rson != nullptr) node->rson->repair();
87     node->repair();
88     int bf = balanceFactor(node);
89
90     if (bf >= -1 && bf <= 1) return node;
91     if (bf > 1)
92     {
93         if (balanceFactor(node->lson) < 0)
94         {
95             node->lson = rotateLeft(node->lson);
96             node->repair();
97         }
98         node = rotateRight(node);
99     }
100     else if (bf < -1)
101     {
102         if (balanceFactor(node->rson) > 0)
103         {
104             node->rson = rotateRight(node->rson);
105             node->repair();
106         }
107         node = rotateLeft(node);
108         node->repair();
109     }
110     return node;
111 }
112 }
```

7.1.3.3 closeScript()

```
void AVL::closeScript ( )
```

Definition at line 79 of file event.cpp.

```
80 {
81     currentScript = nullptr;
82 }
```

7.1.3.4 getHigh()

```
int AVL::getHigh (
    Node * node ) [protected]
```

Definition at line 3 of file rotate.cpp.

```
4 {
5     if (node == nullptr) return 0;
6     return node->high;
7 }
```

7.1.3.5 goBack()

```
void AVL::goBack ( )
```

Definition at line 52 of file step.cpp.

```
53 {  
54 }
```

7.1.3.6 goNext()

```
void AVL::goNext ( )
```

Definition at line 56 of file step.cpp.

```
57 {  
58     pause_mutex.lock();  
59     isQueue = true;  
60     pause_mutex.unlock();  
61     step_cv.notify_one();  
62 }
```

7.1.3.7 goOff()

```
void AVL::goOff ( )
```

Definition at line 72 of file step.cpp.

```
73 {  
74     pause_mutex.lock();  
75     isPause = true;  
76     pause_mutex.unlock();  
77 }
```

7.1.3.8 goOn()

```
void AVL::goOn ( )
```

Definition at line 64 of file step.cpp.

```
65 {  
66     pause_mutex.lock();  
67     isPause = false;  
68     pause_mutex.unlock();  
69     step_cv.notify_one();  
70 }
```

7.1.3.9 highlight()

```
void AVL::highlight (
    std::vector< int > l ) [protected]
```

Definition at line 5 of file step.cpp.

```
6 {
7     if(isAnimate)
8     {
9         animate_mutex.lock();
10        for(int i = 0; i < l.size(); i++)
11        {
12            currentScript->highlight(l[i]);
13        }
14        animate_mutex.unlock();
15    }
16 }
```

7.1.3.10 init()

```
void AVL::init (
    std::vector< int > v )
```

Definition at line 4 of file init.cpp.

```
5 {
6     if(root != nullptr)
7         delete root;
8     root = nullptr;
9     cache = nullptr;
10    sizeOfTree = 0;
11    shiftX = 20;
12    shiftY = 20;
13    isAnimate = false;
14    currentScript = scripts[DATA_STRUCTURES_OPERATOR::INIT];
15    for(int i = 0; i < values.size() && sizeOfTree < capacity; i++)
16    {
17        Sprite* sprite = new Sprite(render);
18        sprite->setFont(font);
19        sprite->linking("AVL/node");
20        sprite->setText(NUMBER::intToString(values[i]));
21
22        root = insert(root, new Node(values[i], sprite));
23    }
24    maxHigh = maxDepth(root);
25 }
```

7.1.3.11 insert() [1/2]

```
bool AVL::insert (
    int key )
```

Definition at line 62 of file insert.cpp.

```
63 {
64     if(sizeOfTree == capacity)
65     {
66         return false;
67     }
68     currentScript = scripts[DATA_STRUCTURES_OPERATOR::INSERT];
69     cache = nullptr;
70
71     Sprite* sprite = new Sprite(render);
72     sprite->setFont(font);
73     sprite->linking("AVL/node");
74
75     sprite->setFontColor(fontColor);
76     sprite->Object::coloring(nodeColor);
```

```

77
78     sprite->setText(NUMBER::intToString(key));
79
80     cache = new Node(key, sprite);
81
82     isPause = false;
83     isQueue = false;
84     isAnimate = true;
85
86     highlight({0});
87     waitForStep();
88     unhighlight({0});
89
90     root = insert(root, cache);
91
92     maxHigh = maxDepth(root);
93
94
95     cache = nullptr;
96
97     return true;
98 }

```

7.1.3.12 insert() [2/2]

```

AVL::Node * AVL::insert (
    Node * node,
    Node * newNode ) [protected]

```

Definition at line 5 of file insert.cpp.

```

6 {
7     if (node == nullptr)
8     {
9         node = newNode;
10        if (isAnimate)
11        {
12            highlight({1, 2, 3});
13            animate_mutex.lock();
14            node->sprite->highlight();
15            animate_mutex.unlock();
16
17            waitForStep();
18
19            animate_mutex.lock();
20            node->sprite->unhighlight();
21            animate_mutex.unlock();
22            unhighlight({1, 2, 3});
23        }
24        sizeOfTree++;
25        return newNode;
26    }
27    if (isAnimate)
28    {
29        animate_mutex.lock();
30        node->sprite->highlight();
31        animate_mutex.unlock();
32
33        waitForStep();
34
35        animate_mutex.lock();
36        node->sprite->unhighlight();
37        animate_mutex.unlock();
38    }
39    if (compare(newNode, node) == -1)
40    {
41        highlight({4, 5});
42        if (isAnimate) waitForStep();
43        unhighlight({4, 5});
44
45        node->lson = insert(node->lson, newNode);
46    }
47    else if (compare(newNode, node) == 1)
48    {
49        highlight({6, 7});
50        if (isAnimate) waitForStep();
51        unhighlight({6, 7});
52        node->rson = insert(node->rson, newNode);
53    }

```

```

54
55     highlight({9});
56     node = balancing(node);
57     unhighlight({9});
58
59     return node;
60 }

```

7.1.3.13 isReceiveEvent()

```

bool AVL::isReceiveEvent (
    SDL_Event & e )

```

Definition at line 3 of file event.cpp.

```

4 {
5     std::lock_guard<std::mutex> lk(animate_mutex);
6     switch(e.type)
7     {
8         case SDL_MOUSEBUTTONDOWN:
9             if(currentScript != nullptr && currentScript->isReceiveEvent(e)) return true;
10            if(e.motion.x < viewport.x || viewport.x + viewport.w < e.motion.x) return false;
11            if(e.motion.y < viewport.y || viewport.y + viewport.h < e.motion.y) return false;
12            if(e.button.button == SDL_BUTTON_LEFT) return false;
13            if(root == nullptr) return false;
14            return true;
15            break;
16        case SDL_MOUSEMOTION:
17            if(isMoving) return true;
18            if(currentScript == nullptr) return false;
19            if(currentScript->isReceiveEvent(e)) return true;
20            return false;
21            break;
22        default:
23            return false;
24            break;
25    }
26 }

```

7.1.3.14 locating()

```

int AVL::locating (
    Node * node,
    int shiftDown,
    int shiftRight ) [protected]

```

Definition at line 83 of file constructor.cpp.

```

84 {
85     if(node == nullptr)
86     {
87         int shift = maxHigh - shiftDown;
88         return (1 << shift) - 1;
89     }
90     int left = locating(node->lson, shiftDown + 1, shiftRight);
91
92     if(node->sprite != nullptr)
93     {
94         node->sprite->locatingX(shiftX + shiftRight * distanceX + left * distanceX);
95         node->sprite->locatingY(shiftY + shiftDown * distanceY);
96         node->sprite->aligning(HORIZONTAL_ALIGN::CENTER, VERTICAL_ALIGN::CENTER);
97     }
98     locating(node->rson, shiftDown + 1, shiftRight + left + 1);
99
100     return (1 << (maxHigh - shiftDown)) - 1;
101 }

```

7.1.3.15 `maxDepth()` [1/2]

```
int AVL::maxDepth ( )
```

Definition at line 70 of file `constructor.cpp`.

```
71 {
72     return maxDepth(root);
73 }
```

7.1.3.16 `maxDepth()` [2/2]

```
int AVL::maxDepth (
    Node * node ) [protected]
```

Definition at line 75 of file `constructor.cpp`.

```
76 {
77     if(node == nullptr) return 0;
78
79     return 1 + std::max(maxDepth(node->lson), maxDepth(node->rson));
80 }
```

7.1.3.17 `react()`

```
Button * AVL::react (
    SDL_Event & e )
```

Definition at line 28 of file `event.cpp`.

```
29 {
30     std::lock_guard<std::mutex> lk(animate_mutex);
31     switch(e.type)
32     {
33         case SDL_MOUSEBUTTONDOWN:
34             if(currentScript != nullptr && currentScript->isReceiveEvent(e))
35             {
36                 return currentScript->react(e);
37             }
38             if(isMoving)
39             {
40                 isMoving = false;
41                 int dx = e.motion.x - lastMousePressed.x;
42                 int dy = e.motion.y - lastMousePressed.y;
43                 shiftX += dx;
44                 shiftY += dy;
45             }else
46             {
47                 isMoving = true;
48                 lastMousePressed.x = e.motion.x;
49                 lastMousePressed.y = e.motion.y;
50             }
51             return nullptr;
52             break;
53         case SDL_MOUSEMOTION:
54             {
55                 if(currentScript != nullptr && currentScript->isReceiveEvent(e))
56                 {
57                     return currentScript->react(e);
58                 }
59                 if(!isMoving) return nullptr;
60                 int dx = e.motion.x - lastMousePressed.x;
61                 int dy = e.motion.y - lastMousePressed.y;
62                 lastMousePressed.x = e.motion.x;
63                 lastMousePressed.y = e.motion.y;
64                 shiftX += dx;
65                 shiftY += dy;
66                 if(cache != nullptr)
67                 {
68                     cache->sprite->moveX(dx);
69                     cache->sprite->moveY(dy);
70                 }
71             }
72     }
```

```

68         }
69         return nullptr;
70         break;
71     }
72     default:
73         return nullptr;
74         break;
75     }
76     return nullptr;
77 }

```

7.1.3.18 remove() [1/2]

```

bool AVL::remove (
    int key )

```

Definition at line 136 of file delete.cpp.

```

137 {
138     if (sizeofTree == 0)
139     {
140         return false;
141     }
142
143     currentScript = scripts[DATA_STRUCTURES_OPERATOR::DELETE];
144
145     isAnimate = true;
146
147     highlight({0});
148     waitForStep();
149     unhighlight({0});
150
151     root = remove(root, key);
152     locating(root, 0, 0);
153
154     cache = nullptr;
155     return true;
156 }

```

7.1.3.19 remove() [2/2]

```

AVL::Node * AVL::remove (
    Node * node,
    int key ) [protected]

```

Definition at line 15 of file delete.cpp.

```

16 {
17     if (node == nullptr)
18     {
19         highlight({1, 2, 3});
20         waitForStep();
21         unhighlight({1, 2, 3});
22         return node;
23     }
24     if (isAnimate)
25     {
26         animate_mutex.lock();
27         node->sprite->highlight();
28         animate_mutex.unlock();
29
30         waitForStep();
31
32         animate_mutex.lock();
33         node->sprite->unhighlight();
34         animate_mutex.unlock();
35     }
36     if (node->key < key)
37     {
38         highlight({7, 8});
39         waitForStep();

```



```

40     unhighlight({7, 8});
41
42     node->rson = remove(node->rson, key);
43
44     highlight({13});
45     waitForStep();
46
47     if(isAnimate)
48     {
49         animate_mutex.lock();
50         node->sprite->highlight();
51         animate_mutex.unlock();
52
53         waitForStep();
54
55         animate_mutex.lock();
56         node->sprite->unhighlight();
57         animate_mutex.unlock();
58     }
59     node = balancing(node);
60     unhighlight({13});
61
62     return node;
63 }
64 else if(node->key > key)
65 {
66     highlight({9, 10, 11});
67     waitForStep();
68     unhighlight({9, 10, 11});
69     node->lson = remove(node->lson, key);
70
71     highlight({13});
72     waitForStep();
73
74     if(isAnimate)
75     {
76         animate_mutex.lock();
77         node->sprite->highlight();
78         animate_mutex.unlock();
79
80         waitForStep();
81
82         animate_mutex.lock();
83         node->sprite->unhighlight();
84         animate_mutex.unlock();
85     }
86
87     node = balancing(node);
88     unhighlight({13});
89
90     return node;
91 }
92 else
93 {
94     highlight({5, 6});
95     waitForStep();
96     unhighlight({5, 6});
97
98     if(node->rson == nullptr)
99     {
100         Node* temp = node->lson;
101         node->lson = nullptr;
102         node->rson = nullptr;
103
104
105         highlight({13});
106         waitForStep();
107         temp = balancing(temp);
108         unhighlight({13});
109
110         delete node;
111         return temp;
112     } else
113     {
114         Node* temp = unplugSmallest(node->rson, cache);
115
116         cache->lson = node->lson;
117
118         cache->rson = temp;
119
120
121         node->lson = nullptr;
122         node->rson = nullptr;
123
124         delete node;
125         node = cache;
126

```

```

127         highlight({13});
128         waitForStep();
129         node = balancing(node);
130         unhighlight({13});
131         return node;
132     }
133 }
134 }

```

7.1.3.20 rendering()

```
void AVL::rendering ( )
```

Definition at line 17 of file rendering.cpp.

```

18 {
19     if (root == nullptr) return ;
20     SDL_RenderSetViewport(render, &viewport);
21
22     std::lock_guard< std::mutex > lock(animate_mutex);
23
24     locating(root, 0, 0);
25
26     std::queue< Node* > q;
27     q.push(root);
28
29     while (!q.empty())
30     {
31         Node* u = q.front();
32         q.pop();
33         if (u->lson != nullptr)
34         {
35             renderLine(u, u->lson);
36             q.push(u->lson);
37         }
38         if (u->rson != nullptr)
39         {
40             renderLine(u, u->rson);
41             q.push(u->rson);
42         }
43         u->sprite->rendering();
44     }
45     if (currentScript != nullptr)
46     {
47         SDL_RenderSetViewport(render, nullptr);
48         currentScript->rendering();
49     }
50     if (cache != nullptr)
51     {
52         //cache->sprite->rendering();
53     }
54 }

```

7.1.3.21 renderLine()

```
void AVL::renderLine (
    Node * src,
    Node * dst ) [protected]
```

Definition at line 4 of file rendering.cpp.

```

5 {
6     SDL_Point psrc = {src->sprite->getX() + src->sprite->getW() / 2, src->sprite->getY() +
7     src->sprite->getH() / 2};
8     SDL_Point pdst = {dst->sprite->getX() + dst->sprite->getW() / 2, dst->sprite->getY() +
9     dst->sprite->getH() / 2};
10
11     SDL_SetRenderDrawColor(render, edgesColor.r, edgesColor.g, edgesColor.b, edgesColor.a);
12     for (int i = -1; i <= 1; i++)
13     {
14         for (int j = -1; j <= 1; j++)
15             SDL_RenderDrawLine(render, psrc.x + i, psrc.y + j, pdst.x + i, pdst.y + j);
16     }
17 }

```

7.1.3.22 rotateLeft()

```
AVL::Node * AVL::rotateLeft (
    Node * node ) [protected]
```

Definition at line 15 of file rotate.cpp.

```
16 {
17     if (node == nullptr) return nullptr;
18     if (node->rson == nullptr) return node;
19
20     if (isAnimate)
21     {
22         waitForStep();
23         node->sprite->highlight();
24         node->rson->sprite->highlight();
25         if (node->rson->rson != nullptr)
26             node->rson->rson->sprite->highlight();
27         waitForStep();
28
29         node->sprite->unhighlight();
30         node->rson->sprite->unhighlight();
31         if (node->rson->rson != nullptr)
32             node->rson->rson->sprite->unhighlight();
33         waitForStep();
34     }
35
36     Node* tmp = node->rson;
37     node->rson = tmp->lson;
38     tmp->lson = node;
39     node->repair();
40     tmp->repair();
41
42     return tmp;
43 }
```

7.1.3.23 rotateRight()

```
AVL::Node * AVL::rotateRight (
    Node * node ) [protected]
```

Definition at line 45 of file rotate.cpp.

```
46 {
47     if (node == nullptr) return nullptr;
48     if (node->lson == nullptr) return node;
49     if (isAnimate)
50     {
51         waitForStep();
52         node->sprite->highlight();
53         node->lson->sprite->highlight();
54         if (node->lson->lson != nullptr)
55             node->lson->lson->sprite->highlight();
56         waitForStep();
57         node->sprite->unhighlight();
58         node->lson->sprite->unhighlight();
59         if (node->lson->lson != nullptr)
60             node->lson->lson->sprite->unhighlight();
61         waitForStep();
62     }
63     Node* tmp = node->lson;
64     node->lson = tmp->rson;
65     int maxDepth;
66     tmp->rson = node;
67     node->repair();
68     tmp->repair();
69     return tmp;
70 }
```

7.1.3.24 search() [1/2]

```
bool AVL::search (
    int key )
```

Definition at line 51 of file search.cpp.

```
52 {
53     if (sizeofTree == 0)
54     {
55         return false;
56     }
57     currentScript = scripts[DATA_STRUCTURES_OPERATOR::SEARCH];
58     cache = nullptr;
59     Sprite* sprite = new Sprite(render);
60     sprite->setFont(font);
61     sprite->linking("AVL/node");
62     sprite->setText(NUMBER::intToString(key));
63
64     cache = new Node(key, sprite);
65
66     isAnimate = true;
67
68     highlight({0});
69     waitForStep();
70     unhighlight({0});
71
72
73     search(root, key);
74
75     delete cache;
76     cache = nullptr;
77     return true;
78 }
```

7.1.3.25 search() [2/2]

```
void AVL::search (
    Node * node,
    int key ) [protected]
```

Definition at line 5 of file search.cpp.

```
6 {
7
8     if (node == nullptr)
9     {
10         highlight({2, 3, 4});
11         waitForStep();
12         unhighlight({2, 3, 4});
13         return ;
14     }
15
16     if (isAnimate)
17     {
18         animate_mutex.lock();
19         node->sprite->highlight();
20         animate_mutex.unlock();
21         waitForStep();
22
23         animate_mutex.lock();
24         node->sprite->unhighlight();
25         animate_mutex.unlock();
26     }
27
28     if (node->key == key)
29     {
30         highlight({6, 7, 8});
31         waitForStep();
32         unhighlight({6, 7, 8});
33         return ;
34     }
35     if (node->key < key)
36     {
37         highlight({10, 11});
38         waitForStep();
```

```

39         unhighlight({10, 11});
40         search(node->rson, key);
41     }
42     else
43     {
44         highlight({12, 13, 14});
45         waitForStep();
46         unhighlight({12, 13, 14});
47         search(node->lson, key);
48     }
49 }

```

7.1.3.26 setEdgesColor()

```

void AVL::setEdgesColor (
    SDL_Color c )

```

7.1.3.27 setNodeColor()

```

void AVL::setNodeColor (
    SDL_Color bg,
    SDL_Color fg )

```

7.1.3.28 setting()

```

void AVL::setting (
    SDL_Color c1,
    SDL_Color c2,
    SDL_Color c3,
    SDL_Color c4 )

```

Definition at line 103 of file constructor.cpp.

```

104 {
105     bgColor = c1;
106     nodeColor = c2;
107     fontColor = c3;
108     edgesColor = c4;
109
110     std::queue<Node*> q;
111     q.push(root);
112
113     while(!q.empty())
114     {
115         Node* node = q.front();
116         q.pop();
117         if(node == nullptr)
118             continue;
119         node->sprite->coloring(nodeColor);
120         node->sprite->setFontColor(fontColor);
121         node->sprite->coloring(nodeColor);
122         q.push(node->lson);
123         q.push(node->rson);
124     }
125 }

```

7.1.3.29 slowDown()

```
void AVL::slowDown ( )
```

Definition at line 84 of file step.cpp.

```
85 {  
86     if(stepWait <= 2000) stepWait = stepWait * 2;  
87 }
```

7.1.3.30 speedUp()

```
void AVL::speedUp ( )
```

Definition at line 79 of file step.cpp.

```
80 {  
81     if(stepWait >= 100) stepWait = stepWait / 2;  
82 }
```

7.1.3.31 unhighlight()

```
void AVL::unhighlight (   
                        std::vector< int > l ) [protected]
```

Definition at line 18 of file step.cpp.

```
19 {  
20     if(isAnimate)  
21     {  
22         animate_mutex.lock();  
23         for(int i = 0; i < l.size(); i++)  
24         {  
25             currentScript->unhighlight(l[i]);  
26         }  
27         animate_mutex.unlock();  
28     }  
29 }
```

7.1.3.32 unplugSmallest()

```
AVL::Node * AVL::unplugSmallest (   
    Node * Node,  
    struct Node *& n2 ) [protected]
```

Definition at line 4 of file delete.cpp.

```
5 {  
6     if(node == nullptr) return nullptr;  
7     if(node->lson == nullptr)  
8     {  
9         n2 = node;  
10        return node->rson;  
11    }  
12    node->lson = unplugSmallest(node->lson, n2);  
13    return node;  
14 }
```

7.1.3.33 waitForStep()

```
void AVL::waitForStep ( ) [protected]
```

Definition at line 31 of file step.cpp.

```
32 {
33     if(isAnimate)
34     {
35         ds_mutex.unlock();
36         std::this_thread::sleep_for(std::chrono::milliseconds(stepWait));
37         ds_mutex.lock();
38     }
39     std::lock_guard<std::mutex> pause_lock(pause_mutex);
40     if(isPause == false)
41     {
42         return ;
43     }
44
45     ds_mutex.unlock();
46     std::unique_lock<std::mutex> lk(step_mutex);
47     step_cv.wait(lk, [&]{return isQueue == true;});
48     isQueue = false;
49     ds_mutex.lock();
50 }
```

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

- [include/data_structures/AVL.hpp](#)
- [src/AVL/constructor.cpp](#)
- [src/AVL/destructor.cpp](#)
- [src/AVL/event.cpp](#)
- [src/AVL/operator/delete.cpp](#)
- [src/AVL/operator/init.cpp](#)
- [src/AVL/operator/insert.cpp](#)
- [src/AVL/operator/search.cpp](#)
- [src/AVL/rendering.cpp](#)
- [src/AVL/rotate.cpp](#)
- [src/AVL/step.cpp](#)

7.2 BTree4th Class Reference

```
#include <btree4th.hpp>
```

Public Member Functions

- [BTree4th \(\)](#)
- [~BTree4th \(\)](#)
- void [init](#) (std::vector< int > keys)
- void [insert](#) (int key)
- void [remove](#) (int key)
- bool [search](#) (int key)

Protected Member Functions

- Node * [leftMost](#) (Node *node)
- Node * [rightMost](#) (Node *node)
- Node * [split](#) (Node *node)
- Node * [addRecordToLeaf](#) (Node *node, int key)
- Node * [mergeChild](#) (Node *node, int index)
- Node * [mergeChild](#) (Node *node, Node *child)
- void [insert](#) (Node *&node, int key)
- bool [search](#) (Node *node, int key)

7.2.1 Detailed Description

Definition at line 8 of file btree4th.hpp.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 BTree4th()

```
BTree4th::BTree4th ( )
```

Definition at line 3 of file constructor.cpp.

```
4 {
5     root = nullptr;
6 }
```

7.2.2.2 ~BTree4th()

```
BTree4th::~~BTree4th ( )
```

Definition at line 3 of file destructor.cpp.

```
4 {
5     if(root != nullptr) delete root;
6 }
```

7.2.3 Member Function Documentation

7.2.3.1 addRecordToLeaf()

```
BTree4th::Node * BTree4th::addRecordToLeaf (
    Node * node,
    int key ) [protected]
```

Definition at line 3 of file addRecord.cpp.

```
4 {
5     if(node == nullptr) return nullptr;
6     if(!node->isLeaf()) return node;
7
8     int index = 0;
9     for(int i = 0; i < 3; i++)
10     {
11         if(node->key[i] == nullptr)
12         {
13             node->key[i] = new int(k);
14             index = i;
15             return node;
16         }
17     }
18     if(index == 1)
19     {
20         if(k < *(node->key[0]))
21             std::swap(node->key[0], node->key[1]);
```



```

22     }
23
24     if(index == 2)
25     {
26         if(*(node->key[0]) < *(node->key[1]))
27             std::swap(node->key[0], node->key[1]);
28         if(*(node->key[1]) < *(node->key[2]))
29             std::swap(node->key[1], node->key[2]);
30         if(*(node->key[0]) < *(node->key[2]))
31             std::swap(node->key[0], node->key[2]);
32     }
33
34     return node;
35 }

```

7.2.3.2 init()

```

void BTree4th::init (
    std::vector< int > keys )

```

Definition at line 4 of file init.cpp.

```

5 {
6     for(int i = 0; i < keys.size(); i++)
7     {
8         root = insert(root, keys[i]);
9     }
10 }

```

7.2.3.3 insert() [1/2]

```

void BTree4th::insert (
    int key )

```

Definition at line 3 of file insert.cpp.

```

4 {
5     if(root == nullptr)
6     {
7         root = new Node(nullptr, key);
8         return ;
9     }
10
11     if(root->isLeaf() && !root->isFull())
12     {
13         root = addRecordToLeaf(root, key);
14         return ;
15     }
16
17     if(root->isFull()) root = split(root);
18
19     Node* current = root;
20
21     do
22     {
23         int pnext = 3;
24         for(int i = 0; i < 3; i++)
25         {
26             if(current->key[i] == nullptr)
27             {
28                 pnext = i;
29                 break;
30             }
31             if(key < *(current->key[i]))
32             {
33                 pnext = i;
34                 break;
35             }
36         }
37         Node *& nnext = current->child[pnext];
38         if(nnext->isLeaf() && !nnext->isFull())

```

```

39         {
40             nxt = addRecordToLeaf(nxt, key);
41             break;
42         }
43         if(nxt->isFull())
44         {
45             nxt = split(nxt);
46             mergeChild(current, pnext);
47         }
48     }while(true);
49 }
50 }

```

7.2.3.4 insert() [2/2]

```

void BTree4th::insert (
    Node *& node,
    int key ) [protected]

```

7.2.3.5 leftMost()

```

Node* BTree4th::leftMost (
    Node * node ) [protected]

```

7.2.3.6 mergeChild() [1/2]

```

BTree4th::Node * BTree4th::mergeChild (
    Node * node,
    int index ) [protected]

```

Definition at line 57 of file addRecord.cpp.

```

58 {
59     if(node == nullptr) return nullptr;
60     if(node->child[index] == nullptr) return node;
61     if(!node->child[index]->isBinary()) return node;
62
63     for(int i = 1; i >= 0; i--)
64     {
65         node->key[i+1] = node->key[i];
66     }
67     for(int i = 2; i >= index; i--)
68     {
69         node->child[i+1] = node->child[i];
70     }
71     Node* tmp = node->child[index];
72
73     node->key[index] = tmp->key[0];
74     tmp->key[0] = nullptr;
75
76     node->child[index] = tmp->child[0];
77     tmp->child[0] = nullptr;
78
79     node->child[index + 1] = tmp->child[1];
80     tmp->child[1] = nullptr;
81
82     delete tmp;
83
84     node->repair(false);
85     return node;
86 }

```

7.2.3.7 mergeChild() [2/2]

```
BTree4th::Node * BTree4th::mergeChild (
    Node * node,
    Node * child ) [protected]
```

Definition at line 37 of file addRecord.cpp.

```
38 {
39     if (node == nullptr) return nullptr;
40     if (child == nullptr) return node;
41     if (!child->isBinary()) return node;
42
43     int index = -1;
44     for (int i = 0; i < 3; i++)
45     {
46         if (node->child[i] == child)
47         {
48             index = i;
49             break;
50         }
51     }
52     if (index == -1) return node;
53
54     return mergeChild(node, index);
55 }
```

7.2.3.8 remove()

```
void BTree4th::remove (
    int key )
```

7.2.3.9 rightMost()

```
Node* BTree4th::rightMost (
    Node * node ) [protected]
```

7.2.3.10 search() [1/2]

```
bool BTree4th::search (
    int key )
```

Definition at line 23 of file search.cpp.

```
24 {
25     return search(root, key);
26 }
```

7.2.3.11 search() [2/2]

```
bool BTree4th::search (
    Node * node,
    int key ) [protected]
```

Definition at line 3 of file search.cpp.

```
4 {
5     if(node == nullptr) return false;
6
7     if(node->key[0] == nullptr) return false;
8
9     if(node->containsKey(key)) return true;
10
11     for(int i = 0; i < 3; i++)
12     {
13         if(node->key[i] == nullptr) break;
14
15         if(key < *(node->key[i]))
16         {
17             return search(node->child[i], key);
18         }
19     }
20     return search(node->child[3], key);
21 }
```

7.2.3.12 split()

```
BTree4th::Node * BTree4th::split (
    Node * node ) [protected]
```

Definition at line 3 of file split.cpp.

```
4 {
5     if(node == nullptr) return nullptr;
6     if(!node->isFull()) return node;
7
8     Node* lson = new Node(node, *(node->key[0]), node->child[0], node->child[1]);
9     Node* rson = new Node(node, *(node->key[2]), node->child[2], node->child[3]);
10
11     delete node->key[0];
12     node->key[0] = node->key[1];
13
14     node->key[1] = nullptr;
15
16     delete node->key[2];
17     node->key[2] = nullptr;
18
19     node->child[0] = lson;
20     node->child[1] = rson;
21     node->child[2] = nullptr;
22     node->child[3] = nullptr;
23
24     return node;
25 }
```

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

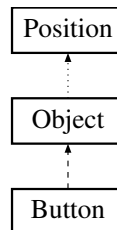
- [include/data_structures/btree4th.hpp](#)
- [src/btree4th/constructor.cpp](#)
- [src/btree4th/destructor.cpp](#)
- [src/btree4th/operator/addRecord.cpp](#)
- [src/btree4th/operator/init.cpp](#)
- [src/btree4th/operator/insert.cpp](#)
- [src/btree4th/operator/search.cpp](#)
- [src/btree4th/operator/split.cpp](#)

7.3 Button Class Reference

[Button](#) class that interact with user input.

```
#include <button.hpp>
```

Inheritance diagram for Button:



Public Member Functions

- [Button](#) (SDL_Renderer *render)
- [~Button](#) ()
- bool [isHover](#) (int x, int y)
- bool [isClicked](#) (int x, int y)
- [BUTTON_ACTION](#) [getAction](#) ()
- std::string [getNextScreen](#) ()
- [DATA_STRUCTURES_TYPE](#) [getDataType](#) ()
- [INPUT_TYPE](#) [getInputType](#) ()
- bool [isReceiveEvent](#) (SDL_Event &e)
- void [rendering](#) ()
- void [linking](#) (std::string n)
- void [move](#) (int dx, int dy)

Protected Member Functions

- void [initSprites](#) (const [json](#) &mem)
- void [initBackground](#) (const [json](#) &mem)
- void [initAction](#) (const [json](#) &mem)
- void [importFromJson](#) ()
- bool [isChooosed](#) (int x, int y)
- void [fillWithColor](#) ()
- void [fillCircleByColor](#) ()
- void [fillRectangleByColor](#) ()
- void [textToTexture](#) ()
- const SDL_Rect * [getCrop](#) ()
- void [cropping](#) (int x, int y, int w, int h)
- void [cropping](#) (SDL_Rect c)
- void [cropping](#) (const [json](#) &mem)
- void [noCropping](#) ()
- const SDL_Rect * [getLocation](#) ()
- int [getX](#) ()
- int [getY](#) ()
- int [getW](#) ()
- int [getH](#) ()

- virtual void [locating](#) (int x, int y, int w, int h)
- virtual void [locating](#) (SDL_Rect l)
- virtual void [locating](#) (const [json](#) &mem)
- virtual void [locatingX](#) (int x)
- virtual void [locatingY](#) (int y)
- virtual void [locatingW](#) (int w)
- virtual void [locatingH](#) (int h)
- virtual void [moveX](#) (int delta)
- virtual void [moveY](#) (int delta)
- virtual void [zoomW](#) (int delta)
- virtual void [zoomH](#) (int delta)
- virtual void [zoom](#) (double delta)
- virtual void [zoomInMiddle](#) (double delta)
- void [fitTheTexture](#) ()
- const SDL_Color * [getColor](#) ()
- void [coloring](#) (int r, int g, int b, int a)
- void [coloring](#) (SDL_Color c)
- void [coloring](#) (const [json](#) &mem)
- void [textureFromFile](#) (std::string dir)
- void [changeToCircle](#) ()
- void [changeToCircle](#) (SDL_Point c)
- void [changeToCircle](#) (int x, int y)
- void [changeToCircle](#) (SDL_Point c, int r)
- void [changeToCircle](#) (int x, int y, int r)
- void [changeToRectangle](#) ()
- void [setShape](#) (const [json](#) &mem)
- bool [isLielInside](#) (int x, int y)
- bool [isLielInside](#) (SDL_Point p)
- bool [isLielInside](#) (SDL_Rect r)
- bool [isLielInside](#) (int x, int y, int w, int h)
- void [show](#) ()
- void [hide](#) ()
- bool [isVisible](#) ()
- void [importFromJson](#) (const [json](#) &mem)
- void [setFont](#) (TTF_Font *f)
- void [setText](#) (std::string t)
- void [addText](#) (std::string t)
- void [addCharacter](#) (char c)
- void [removeCharacter](#) ()
- void [removeCharacter](#) (int n)
- std::string [getText](#) ()
- int [getSize](#) ()

7.3.1 Detailed Description

[Button](#) class that interact with user input.

Drawable

Definition at line 15 of file button.hpp.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 Button()

```
Button::Button (
    SDL_Renderer * render )
```

Definition at line 6 of file constructor.cpp.

```
6         : Object(render)
7 {
8     this->render = render;
9     status = BUTTON_STATUS::RELEASED;
10    action = BUTTON_ACTION::NONE;
11 }
```

7.3.2.2 ~Button()

```
Button::~~Button ( )
```

Definition at line 3 of file destructor.cpp.

```
3     {
4         render = nullptr;
5
6         for(auto&i : sprites) {
7             delete i;
8         }
9         sprites.clear();
10 }
```

7.3.3 Member Function Documentation

7.3.3.1 addCharacter()

```
void Object::addCharacter (
    char c ) [inherited]
```

Definition at line 22 of file font.cpp.

```
23 {
24     text += c;
25     textToTexture();
26 }
```

7.3.3.2 addText()

```
void Object::addText (
    std::string t ) [inherited]
```

Definition at line 16 of file font.cpp.

```
17 {
18     text += t;
19     textToTexture();
20 }
```

7.3.3.3 changeToCircle() [1/5]

```
void Object::changeToCircle ( ) [inherited]
```

Definition at line 5 of file shape.cpp.

```
6 {
7     shapeType = SHAPE::CIRCLE;
8     radius = std::min(getW(), getH()) / 2;
9
10    center.x = getX() + getW() / 2;
11    center.y = getY() + getH() / 2;
12    fillCircleByColor();
13 }
```

7.3.3.4 changeToCircle() [2/5]

```
void Object::changeToCircle (
    int x,
    int y ) [inherited]
```

Definition at line 24 of file shape.cpp.

```
25 {
26     changeToCircle({x, y});
27 }
```

7.3.3.5 changeToCircle() [3/5]

```
void Object::changeToCircle (
    int x,
    int y,
    int r ) [inherited]
```

Definition at line 37 of file shape.cpp.

```
38 {
39     shapeType = SHAPE::CIRCLE;
40     radius = r;
41     center.x = x;
42     center.y = y;
43     fillCircleByColor();
44 }
```


7.3.3.6 changeToCircle() [4/5]

```
void Object::changeToCircle (
    SDL_Point c ) [inherited]
```

Definition at line 15 of file shape.cpp.

```
16 {
17     shapeType = SHAPE::CIRCLE;
18     center = c;
19     radius = std::min(getW() - c.x, c.x - getX());
20     radius = std::min(radius, std::min(getH() - c.y, c.y - getY()));
21     fillCircleByColor();
22 }
```

7.3.3.7 changeToCircle() [5/5]

```
void Object::changeToCircle (
    SDL_Point c,
    int r ) [inherited]
```

Definition at line 29 of file shape.cpp.

```
30 {
31     shapeType = SHAPE::CIRCLE;
32     radius = r;
33     center = c;
34     fillCircleByColor();
35 }
```

7.3.3.8 changeToRectangle()

```
void Object::changeToRectangle ( ) [inherited]
```

Definition at line 46 of file shape.cpp.

```
47 {
48     shapeType = SHAPE::RECTANGLE;
49     fillRectangleByColor();
50 }
```

7.3.3.9 coloring() [1/3]

```
void Object::coloring (
    const json & mem ) [inherited]
```

Definition at line 30 of file coloring.cpp.

```
31 {
32     if(mem.contains("r") && mem.contains("g") && mem.contains("b"))
33     {
34         if(mem.contains("a")) coloring(mem["r"], mem["g"], mem["b"], mem["a"]);
35         else coloring(mem["r"], mem["g"], mem["b"], 255);
36     }
37 }
```

7.3.3.10 coloring() [2/3]

```
void Object::coloring (
    int r,
    int g,
    int b,
    int a ) [inherited]
```

Definition at line 8 of file coloring.cpp.

```
9 {
10     if(color == nullptr) color = new SDL_Color;
11     color->r = r;
12     color->g = g;
13     color->b = b;
14     color->a = a;
15
16     fillWithColor();
17 }
```

7.3.3.11 coloring() [3/3]

```
void Object::coloring (
    SDL_Color c ) [inherited]
```

Definition at line 19 of file coloring.cpp.

```
20 {
21     if(color == nullptr) color = new SDL_Color;
22     color->r = c.r;
23     color->g = c.g;
24     color->b = c.b;
25     color->a = c.a;
26
27     fillWithColor();
28 }
```

7.3.3.12 cropping() [1/3]

```
void Object::cropping (
    const json & mem ) [inherited]
```

Definition at line 26 of file cropping.cpp.

```
27 {
28     if(mem.contains("x") && mem.contains("y") && mem.contains("w") && mem.contains("h"))
29         cropping(mem["x"], mem["y"], mem["w"], mem["h"]);
30 }
```

7.3.3.13 cropping() [2/3]

```
void Object::cropping (
    int x,
    int y,
    int w,
    int h ) [inherited]
```

Definition at line 8 of file cropping.cpp.

```
9 {
10     if(crop == nullptr) crop = new SDL_Rect;
11     crop->x = x;
12     crop->y = y;
13     crop->w = w;
14     crop->h = h;
15 }
```

7.3.3.14 cropping() [3/3]

```
void Object::cropping (
    SDL_Rect c ) [inherited]
```

Definition at line 17 of file cropping.cpp.

```
18 {
19     if(crop == nullptr) crop = new SDL_Rect;
20     crop->x = c.x;
21     crop->y = c.y;
22     crop->w = c.w;
23     crop->h = c.h;
24 }
```

7.3.3.15 fillCircleByColor()

```
void Object::fillCircleByColor ( ) [protected], [inherited]
```

Definition at line 91 of file shape.cpp.

```
92 {
93     if(location == nullptr) locating(0, 0, 0, 0);
94
95     if(texture != nullptr) SDL_DestroyTexture(texture);
96     texture = nullptr;
97
98     Uint32 rmask, gmask, bmask, amask;
99     Uint32 pixelColor;
100     #if SDL_BYTEORDER == SDL_BIG_ENDIAN
101         rmask = 0xff000000;
102         gmask = 0x00ff0000;
103         bmask = 0x0000ff00;
104         amask = 0x000000ff;
105         pixelColor = (color->r << 24) | (color->g << 16) | (color->b << 8) | color->a;
106     #else
107         rmask = 0x000000ff;
108         gmask = 0x0000ff00;
109         bmask = 0x00ff0000;
110         amask = 0xff000000;
111         pixelColor = (color->a << 24) | (color->b << 16) | (color->g << 8) | color->r;
112     #endif
113
114     SDL_Surface *surf = SDL_CreateRGBSurface(0, getW(), getH(), 32, rmask, gmask, bmask, amask);
115     SDL_SetSurfaceBlendMode(surf, SDL_BLENDMODE_BLEND);
116
117     texture = SDL_CreateTextureFromSurface(render, surf);
118     SDL_FreeSurface(surf);
119
120     Uint32 *pixels = new Uint32[getW() * getH()];
121     memset(pixels, 0, getW() * getH() * sizeof(Uint32));
122
123     SDL_Point p = {getW() / 2, getH() / 2};
124     center = p;
125
126     if(radius > std::min(getW(), getH()) / 2) radius = std::min(getW(), getH()) / 2;
127
128     for(int i = p.x - radius; i <= p.x + radius; i++)
129         for(int j = p.y - radius; j <= p.y + radius; j++)
130             if((i - p.x) * (i - p.x) + (j - p.y) * (j - p.y) <= radius * radius)
131             {
132                 int index = i * getW() + j;
133                 if(index < 0 || index >= getW() * getH()) continue;
134                 pixels[index] = pixelColor;
135             }
136
137     SDL_UpdateTexture(texture, nullptr, pixels, getW() * sizeof(Uint32));
138     delete[] pixels;
139 }
```

7.3.3.16 fillRectangleByColor()

void Object::fillRectangleByColor () [protected], [inherited]

Definition at line 74 of file shape.cpp.

```

75 {
76     if(location == nullptr) locating(0, 0, 0, 0);
77
78     if(texture != nullptr) SDL_DestroyTexture(texture);
79     texture = nullptr;
80
81     SDL_Surface* surf = SDL_CreateRGBSurfaceWithFormat(0, getW(), getH(), 32, SDL_PIXELFORMAT_RGBA32);
82
83     SDL_SetSurfaceBlendMode(surf, SDL_BLENDMODE_BLEND);
84
85     SDL_FillRect(surf, nullptr, SDL_MapRGBA(surf->format, color->r, color->g, color->b, color->a));
86
87     texture = SDL_CreateTextureFromSurface(render, surf);
88
89     SDL_FreeSurface(surf);
90 }
```

7.3.3.17 fillWithColor()

void Object::fillWithColor () [protected], [inherited]

Definition at line 39 of file coloring.cpp.

```

40 {
41     if(shapeType == SHAPE::NONE) return fillRectangleByColor();
42     if(shapeType == SHAPE::RECTANGLE) return fillRectangleByColor();
43     if(shapeType == SHAPE::CIRCLE) return fillCircleByColor();
44 }
```

7.3.3.18 fitTheTexture()

void Object::fitTheTexture () [inherited]

Definition at line 140 of file locating.cpp.

```

141 {
142     if(texture == nullptr) return;
143     SDL_QueryTexture(texture, nullptr, nullptr, &location->w, &location->h);
144 }
```

7.3.3.19 getAction()

BUTTON_ACTION Button::getAction ()

Definition at line 3 of file action.cpp.

```

4 {
5     return action;
6 }
```

7.3.3.20 getColor()

```
const SDL_Color * Object::getColor ( ) [inherited]
```

Definition at line 3 of file coloring.cpp.

```
4 {
5     return color;
6 }
```

7.3.3.21 getCrop()

```
const SDL_Rect * Object::getCrop ( ) [inherited]
```

Definition at line 3 of file cropping.cpp.

```
4 {
5     return crop;
6 }
```

7.3.3.22 getDataType()

```
DATA_STRUCTURES_TYPE Button::getDataType ( )
```

Definition at line 15 of file action.cpp.

```
16 {
17     if(!args.contains("data-type"))
18         return DATA_STRUCTURES_TYPE::NONE;
19
20     std::string type = args["data-type"].get<std::string>();
21
22     if(type == "AVL")
23         return DATA_STRUCTURES_TYPE::AVL;
24
25     if(type == "HASH_TABLE")
26         return DATA_STRUCTURES_TYPE::HASH_TABLE;
27
28     if(type == "GRAPH")
29         return DATA_STRUCTURES_TYPE::GRAPH;
30
31     if(type == "TRIE")
32         return DATA_STRUCTURES_TYPE::TRIE;
33
34     if(type == "BTREE_4TH")
35         return DATA_STRUCTURES_TYPE::BTREE_4TH;
36     if(type == "MIN_HEAP")
37         return DATA_STRUCTURES_TYPE::MIN_HEAP;
38     if(type == "MAX_HEAP")
39         return DATA_STRUCTURES_TYPE::MAX_HEAP;
40
41     return DATA_STRUCTURES_TYPE::NONE;
42 }
```

7.3.3.23 getH()

```
int Object::getH ( ) [inherited]
```

Definition at line 47 of file locating.cpp.

```
48 {
49     return location->h;
50 }
```

7.3.3.24 getInputType()

`INPUT_TYPE` Button::getInputType ()

Definition at line 44 of file action.cpp.

```
45 {
46     if(!args.contains("input-type"))
47         return INPUT_TYPE::NONE;
48     std::string type = args["input-type"].get<std::string>();
49
50     if(type == "INT") return INPUT_TYPE::INT;
51     if(type == "ARRAY") return INPUT_TYPE::ARRAY;
52     if(type == "STRING") return INPUT_TYPE::STRING;
53     if(type == "STRINGS") return INPUT_TYPE::STRINGS;
54     return INPUT_TYPE::NONE;
55 }
```

7.3.3.25 getLocation()

`const SDL_Rect *` Object::getLocation () [inherited]

Definition at line 27 of file locating.cpp.

```
28 {
29     return location;
30 }
```

7.3.3.26 getNextScreen()

`std::string` Button::getNextScreen ()

Definition at line 8 of file action.cpp.

```
9 {
10     if(!args.contains("next-screen"))
11         return "";
12     return args["next-screen"].get<std::string>();
13 }
```

7.3.3.27 getSize()

`int` Object::getSize () [inherited]

Definition at line 68 of file font.cpp.

```
69 {
70     return text.size();
71 }
```

7.3.3.28 getText()

`std::string` Object::getText () [inherited]

Definition at line 63 of file font.cpp.

```
64 {
65     return text;
66 }
```

7.3.3.29 getW()

```
int Object::getW ( ) [inherited]
```

Definition at line 42 of file locating.cpp.

```
43 {  
44     return location->w;  
45 }
```

7.3.3.30 getX()

```
int Object::getX ( ) [inherited]
```

Definition at line 32 of file locating.cpp.

```
33 {  
34     return location->x;  
35 }
```

7.3.3.31 getY()

```
int Object::getY ( ) [inherited]
```

Definition at line 37 of file locating.cpp.

```
38 {  
39     return location->y;  
40 }
```

7.3.3.32 hide()

```
void Object::hide ( ) [inherited]
```

Definition at line 8 of file visible.cpp.

```
9 {  
10     visible = false;  
11 }
```

7.3.3.33 importFromJson() [1/2]

```
void Button::importFromJson ( ) [protected]
```

Definition at line 13 of file constructor.cpp.

```
14 {  
15     json* mem = JSON::readFile(PATH::ATB::BUTTON_ + name + ".json");  
16     if(mem->contains("background"))  
17         initBackground((*mem)["background"]);  
18     if(mem->contains("sprites"))  
19         initSprites((*mem)["sprites"]);  
20     if(mem->contains("action"))  
21         initAction((*mem)["action"]);  
22 }
```

7.3.3.34 importFromJson() [2/2]

```
void Object::importFromJson (
    const json & mem ) [inherited]
```

Definition at line 21 of file constructor.cpp.

```
22 {
23     if (mem.contains("location"))
24         locating(mem["location"]);
25
26     if (mem.contains("crop"))
27         cropping(mem["crop"]);
28
29     if (mem.contains("color"))
30         coloring(mem["color"]);
31
32     if (mem.contains("shape"))
33         setShape(mem["shape"]);
34
35     if (mem.contains("visible"))
36         visible = mem["visible"];
37
38     if (mem.contains("image"))
39         textureFromFile(PATH::ASSETS::GRAPHICS_ + mem["image"].get<std::string>());
40     return ;
41 }
```

7.3.3.35 initAction()

```
void Button::initAction (
    const json & mem ) [protected]
```

Definition at line 24 of file constructor.cpp.

```
25 {
26
27     if (mem.contains("type"))
28     {
29         if (mem["type"].get<std::string>() == "CHANGE_SCREEN")
30             action = BUTTON_ACTION::CHANGE_SCREEN;
31         else if (mem["type"].get<std::string>() == "INSERT")
32             action = BUTTON_ACTION::INSERT;
33         else if (mem["type"].get<std::string>() == "DELETE")
34             action = BUTTON_ACTION::DELETE;
35         else if (mem["type"].get<std::string>() == "INIT")
36             action = BUTTON_ACTION::INIT;
37         else if (mem["type"].get<std::string>() == "SEARCH")
38             action = BUTTON_ACTION::SEARCH;
39         else if (mem["type"].get<std::string>() == "SETTING")
40             action = BUTTON_ACTION::SETTING;
41         else if (mem["type"].get<std::string>() == "DONE")
42             action = BUTTON_ACTION::DONE;
43         else if (mem["type"].get<std::string>() == "EDGES")
44             action = BUTTON_ACTION::EDGES;
45         else if (mem["type"].get<std::string>() == "RANDOM")
46             action = BUTTON_ACTION::RANDOM;
47         else if (mem["type"].get<std::string>() == "GO_BACK")
48             action = BUTTON_ACTION::GO_BACK;
49         else if (mem["type"].get<std::string>() == "GO_NEXT")
50             action = BUTTON_ACTION::GO_NEXT;
51         else if (mem["type"].get<std::string>() == "GO_ON")
52             action = BUTTON_ACTION::GO_ON;
53         else if (mem["type"].get<std::string>() == "GO_OFF")
54             action = BUTTON_ACTION::GO_OFF;
55         else if (mem["type"].get<std::string>() == "SPEED_UP")
56             action = BUTTON_ACTION::SPEED_UP;
57         else if (mem["type"].get<std::string>() == "SLOW_DOWN")
58             action = BUTTON_ACTION::SLOW_DOWN;
59         else if (mem["type"].get<std::string>() == "TOP")
60             action = BUTTON_ACTION::TOP;
61         else if (mem["type"].get<std::string>() == "SIZE")
62             action = BUTTON_ACTION::SIZE;
63         else if (mem["type"].get<std::string>() == "CONNECTED_COMPONENTS")
64             action = BUTTON_ACTION::CONNECTED_COMPONENTS;
65         else if (mem["type"].get<std::string>() == "MST")
```



```

67         action = BUTTON_ACTION::MST;
68     else if(mem["type"].get<std::string>() == "DIJKSTRA")
69         action = BUTTON_ACTION::DIJKSTRA;
70     else if(mem["type"].get<std::string>() == "RANDOM2")
71         action = BUTTON_ACTION::RANDOM2;
72     else if(mem["type"].get<std::string>() == "RANDOM3")
73         action = BUTTON_ACTION::RANDOM3;
74     else if(mem["type"].get<std::string>() == "RANDOM4")
75         action = BUTTON_ACTION::RANDOM4;
76     else if(mem["type"].get<std::string>() == "RANDOM5")
77         action = BUTTON_ACTION::RANDOM5;
78     else if(mem["type"].get<std::string>() == "RANDOM6")
79         action = BUTTON_ACTION::RANDOM6;
80     else if(mem["type"].get<std::string>() == "RANDOM7")
81         action = BUTTON_ACTION::RANDOM7;
82     else if(mem["type"].get<std::string>() == "RANDOM8")
83         action = BUTTON_ACTION::RANDOM8;
84     else if(mem["type"].get<std::string>() == "RANDOM9")
85         action = BUTTON_ACTION::RANDOM9;
86     else if(mem["type"].get<std::string>() == "RANDOM10")
87         action = BUTTON_ACTION::RANDOM10;
88     else if(mem["type"].get<std::string>() == "RANDOM11")
89         action = BUTTON_ACTION::RANDOM11;
90     else if(mem["type"].get<std::string>() == "RANDOM12")
91         action = BUTTON_ACTION::RANDOM12;
92     else if(mem["type"].get<std::string>() == "RANDOM13")
93         action = BUTTON_ACTION::RANDOM13;
94     else if(mem["type"].get<std::string>() == "RANDOM14")
95         action = BUTTON_ACTION::RANDOM14;
96     else if(mem["type"].get<std::string>() == "RANDOM15")
97         action = BUTTON_ACTION::RANDOM15;
98     else if(mem["type"].get<std::string>() == "RANDOM16")
99         action = BUTTON_ACTION::RANDOM16;
100    else if(mem["type"].get<std::string>() == "FILE")
101        action = BUTTON_ACTION::FILE;
102    else if(mem["type"].get<std::string>() == "CLOSE")
103        action = BUTTON_ACTION::CLOSE;
104    else
105        action = BUTTON_ACTION::NONE;
106    }
107    if(mem.contains("args"))
108        args = mem["args"];
109 }

```

7.3.3.36 initBackground()

```

void Button::initBackground (
    const json & mem ) [protected]

```

Definition at line 111 of file constructor.cpp.

```

112 {
113     Object::importFromJson(mem);
114 }

```

7.3.3.37 initSprites()

```

void Button::initSprites (
    const json & mem ) [protected]

```

Definition at line 116 of file constructor.cpp.

```

117 {
118     for(auto& sprite : mem)
119     {
120         sprites.push_back(new Sprite(render));
121         sprites.back()->linking(sprite["name"].get<std::string>());
122         sprites.back()->moveX(getX());
123         sprites.back()->moveY(getY());
124     }
125 }

```

7.3.3.38 isChoosed()

```
bool Button::isChoosed (
    int x,
    int y ) [protected]
```

Definition at line 4 of file mouse_action.cpp.

```
5 {
6     return x >= getX() && x < getX() + getW() && y >= getY() && y < getY() + getH();
7 }
```

7.3.3.39 isClicked()

```
bool Button::isClicked (
    int x,
    int y )
```

Definition at line 32 of file mouse_action.cpp.

```
33 {
34     if (isChoosed(x, y))
35     {
36         sprites[0]->hide();
37         sprites[1]->show();
38         status = BUTTON_STATUS::HOVER;
39         return true;
40     }
41     else
42     {
43         sprites[0]->show();
44         sprites[1]->hide();
45         status = BUTTON_STATUS::RELEASED;
46         return false;
47     }
48 }
```

7.3.3.40 isHover()

```
bool Button::isHover (
    int x,
    int y )
```

Definition at line 9 of file mouse_action.cpp.

```
10 {
11     if (!isVisible())
12     {
13         return false;
14     }
15     if (isChoosed(x, y))
16     {
17         status = BUTTON_STATUS::HOVER;
18         sprites[0]->hide();
19         sprites[1]->show();
20         return true;
21     }
22     else
23     {
24         sprites[0]->show();
25         sprites[1]->hide();
26         status = BUTTON_STATUS::RELEASED;
27         return false;
28     }
29 }
```

7.3.3.41 isLieInside() [1/4]

```
bool Object::isLieInside (
    int x,
    int y ) [inherited]
```

Definition at line 3 of file locating.cpp.

```
4 {
5     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
6     return (x >= location->x && x < location->x + location->w && y >= location->y && y < location->y +
        location->h);
7 }
```

7.3.3.42 isLieInside() [2/4]

```
bool Object::isLieInside (
    int x,
    int y,
    int w,
    int h ) [inherited]
```

Definition at line 21 of file locating.cpp.

```
22 {
23     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
24     return (x >= location->x && x + w <= location->x + location->w && y >= location->y && y + h <=
        location->y + location->h);
25 }
```

7.3.3.43 isLieInside() [3/4]

```
bool Object::isLieInside (
    SDL_Point p ) [inherited]
```

Definition at line 9 of file locating.cpp.

```
10 {
11     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
12     return (p.x >= location->x && p.x < location->x + location->w && p.y >= location->y && p.y <
        location->y + location->h);
13 }
```

7.3.3.44 isLieInside() [4/4]

```
bool Object::isLieInside (
    SDL_Rect r ) [inherited]
```

Definition at line 15 of file locating.cpp.

```
16 {
17     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
18     return (r.x >= location->x && r.x + r.w <= location->x + location->w && r.y >= location->y && r.y +
        r.h <= location->y + location->h);
19 }
```

7.3.3.45 isReceiveEvent()

```
bool Button::isReceiveEvent (
    SDL_Event & e )
```

Definition at line 3 of file event.cpp.

```
4 {
5     switch(e.type)
6     {
7         case SDL_MOUSEMOTION:
8             return isChooosed(e.motion.x, e.motion.y) || status == BUTTON_STATUS::HOVER;
9             break;
10        case SDL_MOUSEBUTTONDOWN:
11            return isChooosed(e.button.x, e.button.y);
12            break;
13        default:
14            return false;
15    }
16 }
17
18 }
```

7.3.3.46 isVisible()

```
bool Object::isVisible ( ) [inherited]
```

Definition at line 13 of file visible.cpp.

```
14 {
15     return visible;
16 }
```

7.3.3.47 linking()

```
void Button::linking (
    std::string n )
```

Definition at line 128 of file constructor.cpp.

```
129 {
130     name = n;
131     importFromJson();
132 }
```

7.3.3.48 locating() [1/3]

```
void Object::locating (
    const json & mem ) [virtual], [inherited]
```

Definition at line 70 of file locating.cpp.

```
71 {
72     if(mem.contains("x") && mem.contains("y") && mem.contains("w") && mem.contains("h"))
73         locating(mem["x"], mem["y"], mem["w"], mem["h"]);
74 }
```

7.3.3.49 locating() [2/3]

```
void Object::locating (
    int x,
    int y,
    int w,
    int h ) [virtual], [inherited]
```

Reimplemented in [Sprite](#).

Definition at line 52 of file locating.cpp.

```
53 {
54     if(location == nullptr) location = new SDL_Rect;
55     location->x = x;
56     location->y = y;
57     location->w = w;
58     location->h = h;
59 }
```

7.3.3.50 locating() [3/3]

```
void Object::locating (
    SDL_Rect l ) [virtual], [inherited]
```

Reimplemented in [Sprite](#).

Definition at line 61 of file locating.cpp.

```
62 {
63     if(location == nullptr) location = new SDL_Rect;
64     location->x = l.x;
65     location->y = l.y;
66     location->w = l.w;
67     location->h = l.h;
68 }
```

7.3.3.51 locatingH()

```
void Object::locatingH (
    int h ) [virtual], [inherited]
```

Definition at line 94 of file locating.cpp.

```
95 {
96     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
97     location->h = h;
98 }
```

7.3.3.52 locatingW()

```
void Object::locatingW (
    int w ) [virtual], [inherited]
```

Definition at line 88 of file locating.cpp.

```
89 {
90     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
91     location->w = w;
92 }
```

7.3.3.53 locatingX()

```
void Object::locatingX (
    int x ) [virtual], [inherited]
```

Reimplemented in [Sprite](#).

Definition at line 76 of file locating.cpp.

```
77 {
78     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
79     location->x = x;
80 }
```

7.3.3.54 locatingY()

```
void Object::locatingY (
    int y ) [virtual], [inherited]
```

Reimplemented in [Sprite](#).

Definition at line 82 of file locating.cpp.

```
83 {
84     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
85     location->y = y;
86 }
```

7.3.3.55 move()

```
void Button::move (
    int dx,
    int dy )
```

Definition at line 134 of file constructor.cpp.

```
135 {
136     Object::moveX(dx);
137     Object::moveY(dy);
138     for(auto& sprite : sprites)
139     {
140         sprite->moveX(dx);
141         sprite->moveY(dy);
142     }
143 }
```

7.3.3.56 moveX()

```
void Object::moveX (
    int delta ) [virtual], [inherited]
```

Reimplemented in [Sprite](#).

Definition at line 100 of file locating.cpp.

```
101 {
102     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
103     location->x += delta;
104 }
```

7.3.3.57 moveY()

```
void Object::moveY (
    int delta ) [virtual], [inherited]
```

Reimplemented in [Sprite](#).

Definition at line 106 of file `locating.cpp`.

```
107 {
108     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
109     location->y += delta;
110 }
```

7.3.3.58 noCropping()

```
void Object::noCropping ( ) [inherited]
```

Definition at line 32 of file `cropping.cpp`.

```
33 {
34     if(crop != nullptr) delete crop;
35     crop = nullptr;
36 }
```

7.3.3.59 removeCharacter() [1/2]

```
void Object::removeCharacter ( ) [inherited]
```

Definition at line 28 of file `font.cpp`.

```
29 {
30     if (text.size() > 0)
31         text.pop_back();
32     textToTexture();
33 }
```

7.3.3.60 removeCharacter() [2/2]

```
void Object::removeCharacter (
    int n ) [inherited]
```

Definition at line 35 of file `font.cpp`.

```
36 {
37     if(n == 0) return ;
38     if(text.size() <= n) text.clear();
39     else text.erase(text.end() - n, text.end());
40     textToTexture();
41 }
```

7.3.3.61 rendering()

```
void Button::rendering ( ) [virtual]
```

Reimplemented from [Object](#).

Definition at line 4 of file rendering.cpp.

```
5 {
6     Object::rendering();
7     if(status == BUTTON_STATUS::HOVER)
8     {
9         sprites[1]->rendering();
10
11     }else if(status == BUTTON_STATUS::RELEASED)
12         sprites[0]->rendering();
13 }
```

7.3.3.62 setFont()

```
void Object::setFont (
    TTF_Font * f ) [inherited]
```

Definition at line 4 of file font.cpp.

```
5 {
6     font = f;
7     textToTexture();
8 }
```

7.3.3.63 setShape()

```
void Object::setShape (
    const json & mem ) [inherited]
```

Definition at line 52 of file shape.cpp.

```
53 {
54     if(mem["type"].get<std::string>() == "CIRCLE")
55     {
56         if(mem.contains("center"))
57         {
58             if(mem.contains("radius"))
59                 changeToCircle(mem["center"]["x"], mem["center"]["y"], mem["radius"]);
60             else changeToCircle(mem["center"]["x"], mem["center"]["y"]);
61         }else changeToCircle();
62
63         return ;
64     }
65
66     if(mem["type"].get<std::string>() == "NONE" || mem["type"].get<std::string>() == "RECTANGLE")
67     {
68         changeToRectangle();
69         return ;
70     }
71
72 }
```


7.3.3.64 setText()

```
void Object::setText (
    std::string t ) [inherited]
```

Definition at line 10 of file font.cpp.

```
11 {
12     text = t;
13     textToTexture();
14 }
```

7.3.3.65 show()

```
void Object::show ( ) [inherited]
```

Definition at line 3 of file visible.cpp.

```
4 {
5     visible = true;
6 }
```

7.3.3.66 textToTexture()

```
void Object::textToTexture ( ) [protected], [inherited]
```

Definition at line 43 of file font.cpp.

```
44 {
45     if(font == nullptr) return ;
46     if(color == nullptr) return ;
47     if(render == nullptr) return ;
48     if(texture != nullptr)
49     {
50         SDL_DestroyTexture(texture);
51     }
52     texture = nullptr;
53
54     SDL_Surface* surface = TTF_RenderText_Blended(font, text.c_str(), *color);
55
56     if(surface == nullptr) return ;
57
58     texture = SDL_CreateTextureFromSurface(render, surface);
59     SDL_FreeSurface(surface);
60     fitTheTexture();
61 }
```

7.3.3.67 textureFromFile()

```
void Object::textureFromFile (
    std::string dir ) [inherited]
```

Definition at line 4 of file external_storage.cpp.

```
5 {
6     SDL_Surface *surface = IMG_Load(dir.c_str());
7
8     texture = SDL_CreateTextureFromSurface(render, surface);
9     SDL_FreeSurface(surface);
10 }
```

7.3.3.68 zoom()

```
void Object::zoom (
    double delta ) [virtual], [inherited]
```

Definition at line 123 of file locating.cpp.

```
124 {
125     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
126     location->w *= delta;
127     location->h *= delta;
128 }
```

7.3.3.69 zoomH()

```
void Object::zoomH (
    int delta ) [virtual], [inherited]
```

Definition at line 118 of file locating.cpp.

```
119 {
120     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
121     location->h += delta;
122 }
```

7.3.3.70 zoomInMiddle()

```
void Object::zoomInMiddle (
    double delta ) [virtual], [inherited]
```

Definition at line 130 of file locating.cpp.

```
131 {
132     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
133     SDL_Point center = {location->x + location->w / 2, location->y + location->h / 2};
134     location->w *= delta;
135     location->h *= delta;
136     location->x = center.x - location->w / 2;
137     location->y = center.y - location->h / 2;
138 }
```

7.3.3.71 zoomW()

```
void Object::zoomW (
    int delta ) [virtual], [inherited]
```

Definition at line 112 of file locating.cpp.

```
113 {
114     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
115     location->w += delta;
116 }
```

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

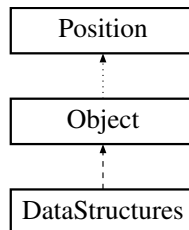
- [include/button.hpp](#)
- [src/button/action.cpp](#)
- [src/button/constructor.cpp](#)
- [src/button/destructor.cpp](#)
- [src/button/event.cpp](#)
- [src/button/mouse_action.cpp](#)
- [src/button/rendering.cpp](#)

7.4 DataStructures Class Reference

Container that contains all data structures.

```
#include <data_structures.hpp>
```

Inheritance diagram for DataStructures:



Public Member Functions

- [DataStructures](#) (SDL_Renderer *r, TTF_Font *f, std::mutex &m)
- [~DataStructures](#) ()
- void [linking](#) (std::string n)
- void [setDataType](#) (DATA_STRUCTURES_TYPE t)
- [DATA_STRUCTURES_TYPE](#) [getDataType](#) ()
- std::string [getName](#) ()
- void [rendering](#) ()
- void [init](#) (InputBox *inp)
- void [insert](#) (InputBox *inp)
- void [remove](#) (InputBox *inp)
- void [search](#) (InputBox *inp)
- void [setting](#) (InputBox *inp)
- void [top](#) ()
- void [size](#) ()
- void [scc](#) ()
- void [mst](#) ()
- void [dijkstra](#) (InputBox *inp)
- void [goBack](#) ()
- void [goNext](#) ()
- void [goOn](#) ()
- void [goOff](#) ()
- void [speedUp](#) ()
- void [slowDown](#) ()
- void [closeScript](#) ()
- bool [isReceiveEvent](#) (SDL_Event &e)
- [Button](#) * [react](#) (SDL_Event &e)

Public Attributes

- int [capacity](#)

Protected Member Functions

- void [initBackground](#) (const [json](#) &mem)
- void [initLinker](#) (const [json](#) &mem)
- void [initDisplay](#) (const [json](#) &mem)
- void [importFromJson](#) ()
- void [initAVL](#) ([InputBox](#) *inp)
- void [insertAVL](#) ([InputBox](#) *inp)
- void [removeAVL](#) ([InputBox](#) *inp)
- void [searchAVL](#) ([InputBox](#) *inp)
- void [settingAVL](#) ([InputBox](#) *inp)
- void [initTrie](#) ([InputBox](#) *inp)
- void [insertTrie](#) ([InputBox](#) *inp)
- void [removeTrie](#) ([InputBox](#) *inp)
- void [searchTrie](#) ([InputBox](#) *inp)
- void [settingTrie](#) ([InputBox](#) *inp)
- void [initHashTable](#) ([InputBox](#) *inp)
- void [insertHashTable](#) ([InputBox](#) *inp)
- void [removeHashTable](#) ([InputBox](#) *inp)
- void [searchHashTable](#) ([InputBox](#) *inp)
- void [settingHashTable](#) ([InputBox](#) *inp)
- void [initMinHeap](#) ([InputBox](#) *inp)
- void [insertMinHeap](#) ([InputBox](#) *inp)
- void [removeMinHeap](#) ([InputBox](#) *inp)
- void [searchMinHeap](#) ([InputBox](#) *inp)
- void [settingMinHeap](#) ([InputBox](#) *inp)
- void [initGraph](#) ([InputBox](#) *inp)
- void [Dijkstra](#) ([InputBox](#) *inp)
- void [MST](#) ()
- void [SCC](#) ()
- void [settingGraph](#) ([InputBox](#) *inp)
- void [fillWithColor](#) ()
- void [fillCircleByColor](#) ()
- void [fillRectangleByColor](#) ()
- void [textToTexture](#) ()
- const [SDL_Rect](#) * [getCrop](#) ()
- void [cropping](#) (int x, int y, int w, int h)
- void [cropping](#) ([SDL_Rect](#) c)
- void [cropping](#) (const [json](#) &mem)
- void [noCropping](#) ()
- const [SDL_Rect](#) * [getLocation](#) ()
- int [getX](#) ()
- int [getY](#) ()
- int [getW](#) ()
- int [getH](#) ()
- virtual void [locating](#) (int x, int y, int w, int h)
- virtual void [locating](#) ([SDL_Rect](#) l)
- virtual void [locating](#) (const [json](#) &mem)
- virtual void [locatingX](#) (int x)
- virtual void [locatingY](#) (int y)
- virtual void [locatingW](#) (int w)
- virtual void [locatingH](#) (int h)
- virtual void [moveX](#) (int delta)
- virtual void [moveY](#) (int delta)
- virtual void [zoomW](#) (int delta)

- virtual void [zoomH](#) (int delta)
- virtual void [zoom](#) (double delta)
- virtual void [zoomInMiddle](#) (double delta)
- void [fitTheTexture](#) ()
- const SDL_Color * [getColor](#) ()
- void [coloring](#) (int r, int g, int b, int a)
- void [coloring](#) (SDL_Color c)
- void [coloring](#) (const [json](#) &mem)
- void [textureFromFile](#) (std::string dir)
- void [changeToCircle](#) ()
- void [changeToCircle](#) (SDL_Point c)
- void [changeToCircle](#) (int x, int y)
- void [changeToCircle](#) (SDL_Point c, int r)
- void [changeToCircle](#) (int x, int y, int r)
- void [changeToRectangle](#) ()
- void [setShape](#) (const [json](#) &mem)
- bool [isLieInside](#) (int x, int y)
- bool [isLieInside](#) (SDL_Point p)
- bool [isLieInside](#) (SDL_Rect r)
- bool [isLieInside](#) (int x, int y, int w, int h)
- void [show](#) ()
- void [hide](#) ()
- bool [isVisible](#) ()
- void [importFromJson](#) (const [json](#) &mem)
- void [setFont](#) (TTF_Font *f)
- void [setText](#) (std::string t)
- void [addText](#) (std::string t)
- void [addCharacter](#) (char c)
- void [removeCharacter](#) ()
- void [removeCharacter](#) (int n)
- std::string [getText](#) ()
- int [getSize](#) ()

7.4.1 Detailed Description

Container that contains all data structures.

Drawable

contain [AVL](#)

contain [Trie](#)

contain Hash Table

contain [Graph](#)

contain Heap

Definition at line 40 of file data_structures.hpp.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 DataStructures()

```
DataStructures::DataStructures (
    SDL_Renderer * r,
    TTF_Font * f,
    std::mutex & m )
```

Definition at line 3 of file constructor.cpp.

```
3                                     : Object(r), ds_mutex(m)
4 {
5     font = f;
6     render = r;
7     avl = nullptr;
8     trie = nullptr;
9     hashTable = nullptr;
10    minheap = nullptr;
11    graph = nullptr;
12 }
```

7.4.2.2 ~DataStructures()

```
DataStructures::~DataStructures ( )
```

Definition at line 3 of file destructor.cpp.

```
4 {
5     render = nullptr;
6     for(auto& i : node)
7         delete i;
8     for(auto& i : displays)
9         delete i;
10    node.clear();
11    if(avl != nullptr) delete avl;
12    if(trie != nullptr) delete trie;
13    if(hashTable != nullptr) delete hashTable;
14    if(minheap != nullptr) delete minheap;
15    if(graph != nullptr) delete graph;
16 }
```

7.4.3 Member Function Documentation

7.4.3.1 addCharacter()

```
void Object::addCharacter (
    char c ) [inherited]
```

Definition at line 22 of file font.cpp.

```
23 {
24     text += c;
25     textToTexture();
26 }
```

7.4.3.2 addText()

```
void Object::addText (
    std::string t ) [inherited]
```

Definition at line 16 of file font.cpp.

```
17 {
18     text += t;
19     textToTexture();
20 }
```

7.4.3.3 changeToCircle() [1/5]

```
void Object::changeToCircle ( ) [inherited]
```

Definition at line 5 of file shape.cpp.

```
6 {
7     shapeType = SHAPE::CIRCLE;
8     radius = std::min(getW(), getH()) / 2;
9
10    center.x = getX() + getW() / 2;
11    center.y = getY() + getH() / 2;
12    fillCircleByColor();
13 }
```

7.4.3.4 changeToCircle() [2/5]

```
void Object::changeToCircle (
    int x,
    int y ) [inherited]
```

Definition at line 24 of file shape.cpp.

```
25 {
26     changeToCircle({x, y});
27 }
```

7.4.3.5 changeToCircle() [3/5]

```
void Object::changeToCircle (
    int x,
    int y,
    int r ) [inherited]
```

Definition at line 37 of file shape.cpp.

```
38 {
39     shapeType = SHAPE::CIRCLE;
40     radius = r;
41     center.x = x;
42     center.y = y;
43     fillCircleByColor();
44 }
```

7.4.3.6 changeToCircle() [4/5]

```
void Object::changeToCircle (
    SDL_Point c ) [inherited]
```

Definition at line 15 of file shape.cpp.

```
16 {
17     shapeType = SHAPE::CIRCLE;
18     center = c;
19     radius = std::min(getW() - c.x, c.x - getX());
20     radius = std::min(radius, std::min(getH() - c.y, c.y - getY()));
21     fillCircleByColor();
22 }
```

7.4.3.7 changeToCircle() [5/5]

```
void Object::changeToCircle (
    SDL_Point c,
    int r ) [inherited]
```

Definition at line 29 of file shape.cpp.

```
30 {
31     shapeType = SHAPE::CIRCLE;
32     radius = r;
33     center = c;
34     fillCircleByColor();
35 }
```

7.4.3.8 changeToRectangle()

```
void Object::changeToRectangle ( ) [inherited]
```

Definition at line 46 of file shape.cpp.

```
47 {
48     shapeType = SHAPE::RECTANGLE;
49     fillRectangleByColor();
50 }
```

7.4.3.9 closeScript()

```
void DataStructures::closeScript ( )
```

Definition at line 78 of file step.cpp.

```
79 {
80     switch(type)
81     {
82         case DATA_STRUCTURES_TYPE::AVL:
83             if(avl == nullptr) return ;
84             avl->closeScript();
85             break;
86         case DATA_STRUCTURES_TYPE::TRIE:
87             if(trie == nullptr) return ;
88             trie->closeScript();
89             break;
90         case DATA_STRUCTURES_TYPE::HASH_TABLE:
91             if(hashTable == nullptr) return ;
92             hashTable->closeScript();
```



```

93         break;
94     case DATA_STRUCTURES_TYPE::MIN_HEAP:
95         if(minheap == nullptr) return ;
96         minheap->closeScript();
97         break;
98     case DATA_STRUCTURES_TYPE::MAX_HEAP:
99         if(minheap == nullptr) return ;
100        minheap->closeScript();
101        break;
102    default:
103        break;
104    }
105 }

```

7.4.3.10 coloring() [1/3]

```

void Object::coloring (
    const json & mem ) [inherited]

```

Definition at line 30 of file coloring.cpp.

```

31 {
32     if(mem.contains("r") && mem.contains("g") && mem.contains("b"))
33     {
34         if(mem.contains("a")) coloring(mem["r"], mem["g"], mem["b"], mem["a"]);
35         else coloring(mem["r"], mem["g"], mem["b"], 255);
36     }
37 }

```

7.4.3.11 coloring() [2/3]

```

void Object::coloring (
    int r,
    int g,
    int b,
    int a ) [inherited]

```

Definition at line 8 of file coloring.cpp.

```

9 {
10     if(color == nullptr) color = new SDL_Color;
11     color->r = r;
12     color->g = g;
13     color->b = b;
14     color->a = a;
15
16     fillWithColor();
17 }

```

7.4.3.12 coloring() [3/3]

```

void Object::coloring (
    SDL_Color c ) [inherited]

```

Definition at line 19 of file coloring.cpp.

```

20 {
21     if(color == nullptr) color = new SDL_Color;
22     color->r = c.r;
23     color->g = c.g;
24     color->b = c.b;
25     color->a = c.a;
26
27     fillWithColor();
28 }

```

7.4.3.13 cropping() [1/3]

```
void Object::cropping (
    const json & mem ) [inherited]
```

Definition at line 26 of file cropping.cpp.

```
27 {
28     if(mem.contains("x") && mem.contains("y") && mem.contains("w") && mem.contains("h"))
29         cropping(mem["x"], mem["y"], mem["w"], mem["h"]);
30 }
```

7.4.3.14 cropping() [2/3]

```
void Object::cropping (
    int x,
    int y,
    int w,
    int h ) [inherited]
```

Definition at line 8 of file cropping.cpp.

```
9 {
10     if(crop == nullptr) crop = new SDL_Rect;
11     crop->x = x;
12     crop->y = y;
13     crop->w = w;
14     crop->h = h;
15 }
```

7.4.3.15 cropping() [3/3]

```
void Object::cropping (
    SDL_Rect c ) [inherited]
```

Definition at line 17 of file cropping.cpp.

```
18 {
19     if(crop == nullptr) crop = new SDL_Rect;
20     crop->x = c.x;
21     crop->y = c.y;
22     crop->w = c.w;
23     crop->h = c.h;
24 }
```

7.4.3.16 Dijkstra()

```
void DataStructures::Dijkstra (
    InputBox * inp ) [protected]
```

7.4.3.17 dijkstra()

```
void DataStructures::dijkstra (
    InputBox * inp )
```

Definition at line 3 of file dijkstra.cpp.

```
4 {
5     int src = NUMBER::stringToInt(inp->getText(1));
6     int dst = NUMBER::stringToInt(inp->getText(2));
7     graph->Dijkstra(src, dst);
8 }
```

7.4.3.18 fillCircleByColor()

```
void Object::fillCircleByColor ( ) [protected], [inherited]
```

Definition at line 91 of file shape.cpp.

```
92 {
93     if(location == nullptr) locating(0, 0, 0, 0);
94
95     if(texture != nullptr) SDL_DestroyTexture(texture);
96     texture = nullptr;
97
98     Uint32 rmask, gmask, bmask, amask;
99     Uint32 pixelColor;
100 #if SDL_BYTEORDER == SDL_BIG_ENDIAN
101     rmask = 0xff000000;
102     gmask = 0x00ff0000;
103     bmask = 0x0000ff00;
104     amask = 0x000000ff;
105     pixelColor = (color->r << 24) | (color->g << 16) | (color->b << 8) | color->a;
106 #else
107     rmask = 0x000000ff;
108     gmask = 0x0000ff00;
109     bmask = 0x00ff0000;
110     amask = 0xff000000;
111     pixelColor = (color->a << 24) | (color->b << 16) | (color->g << 8) | color->r;
112 #endif
113
114     SDL_Surface *surf = SDL_CreateRGBSurface(0, getW(), getH(), 32, rmask, gmask, bmask, amask);
115     SDL_SetSurfaceBlendMode(surf, SDL_BLENDMODE_BLEND);
116
117     texture = SDL_CreateTextureFromSurface(render, surf);
118     SDL_FreeSurface(surf);
119
120     Uint32 *pixels = new Uint32[getW() * getH()];
121     memset(pixels, 0, getW() * getH() * sizeof(Uint32));
122
123     SDL_Point p = {getW() / 2, getH() / 2};
124     center = p;
125
126     if(radius > std::min(getW(), getH()) / 2) radius = std::min(getW(), getH()) / 2;
127
128     for(int i = p.x - radius; i <= p.x + radius; i++)
129         for(int j = p.y - radius; j <= p.y + radius; j++)
130             if((i - p.x) * (i - p.x) + (j - p.y) * (j - p.y) <= radius * radius)
131             {
132                 int index = i * getW() + j;
133                 if(index < 0 || index >= getW() * getH()) continue;
134                 pixels[index] = pixelColor;
135             }
136
137     SDL_UpdateTexture(texture, nullptr, pixels, getW() * sizeof(Uint32));
138     delete[] pixels;
139 }
```

7.4.3.19 fillRectangleByColor()

void Object::fillRectangleByColor () [protected], [inherited]

Definition at line 74 of file shape.cpp.

```

75 {
76     if(location == nullptr) locating(0, 0, 0, 0);
77
78     if(texture != nullptr) SDL_DestroyTexture(texture);
79     texture = nullptr;
80
81     SDL_Surface* surf = SDL_CreateRGBSurfaceWithFormat(0, getW(), getH(), 32, SDL_PIXELFORMAT_RGBA32);
82
83     SDL_SetSurfaceBlendMode(surf, SDL_BLENDMODE_BLEND);
84
85     SDL_FillRect(surf, nullptr, SDL_MapRGBA(surf->format, color->r, color->g, color->b, color->a));
86
87     texture = SDL_CreateTextureFromSurface(render, surf);
88
89     SDL_FreeSurface(surf);
90 }
```

7.4.3.20 fillWithColor()

void Object::fillWithColor () [protected], [inherited]

Definition at line 39 of file coloring.cpp.

```

40 {
41     if(shapeType == SHAPE::NONE) return fillRectangleByColor();
42     if(shapeType == SHAPE::RECTANGLE) return fillRectangleByColor();
43     if(shapeType == SHAPE::CIRCLE) return fillCircleByColor();
44 }
```

7.4.3.21 fitTheTexture()

void Object::fitTheTexture () [inherited]

Definition at line 140 of file locating.cpp.

```

141 {
142     if(texture == nullptr) return;
143     SDL_QueryTexture(texture, nullptr, nullptr, &location->w, &location->h);
144 }
```

7.4.3.22 getColor()

const SDL_Color * Object::getColor () [inherited]

Definition at line 3 of file coloring.cpp.

```

4 {
5     return color;
6 }
```

7.4.3.23 getCrop()

```
const SDL_Rect * Object::getCrop ( ) [inherited]
```

Definition at line 3 of file cropping.cpp.

```
4 {  
5     return crop;  
6 }
```

7.4.3.24 getDataType()

```
DATA_STRUCTURES_TYPE DataStructures::getDataType ( )
```

Definition at line 4 of file operator.cpp.

```
5 {  
6     return type;  
7 }
```

7.4.3.25 getH()

```
int Object::getH ( ) [inherited]
```

Definition at line 47 of file locating.cpp.

```
48 {  
49     return location->h;  
50 }
```

7.4.3.26 getLocation()

```
const SDL_Rect * Object::getLocation ( ) [inherited]
```

Definition at line 27 of file locating.cpp.

```
28 {  
29     return location;  
30 }
```

7.4.3.27 getName()

```
std::string DataStructures::getName ( )
```

Definition at line 9 of file operator.cpp.

```
10 {  
11     return name;  
12 }
```

7.4.3.28 getSize()

int Object::getSize () [inherited]

Definition at line 68 of file font.cpp.

```
69 {  
70     return text.size();  
71 }
```

7.4.3.29 getText()

std::string Object::getText () [inherited]

Definition at line 63 of file font.cpp.

```
64 {  
65     return text;  
66 }
```

7.4.3.30 getW()

int Object::getW () [inherited]

Definition at line 42 of file locating.cpp.

```
43 {  
44     return location->w;  
45 }
```

7.4.3.31 getX()

int Object::getX () [inherited]

Definition at line 32 of file locating.cpp.

```
33 {  
34     return location->x;  
35 }
```

7.4.3.32 getY()

int Object::getY () [inherited]

Definition at line 37 of file locating.cpp.

```
38 {  
39     return location->y;  
40 }
```

7.4.3.33 goBack()

```
void DataStructures::goBack ( )
```

Definition at line 3 of file step.cpp.

```
4 {  
5  
6 }
```

7.4.3.34 goNext()

```
void DataStructures::goNext ( )
```

Definition at line 9 of file step.cpp.

```
10 {  
11     switch(type)  
12     {  
13         case DATA_STRUCTURES_TYPE::AVL:  
14             if(avl == nullptr) return ;  
15             avl->goNext();  
16             break;  
17         default:  
18             break;  
19     }  
20 }
```

7.4.3.35 goOff()

```
void DataStructures::goOff ( )
```

Definition at line 37 of file step.cpp.

```
38 {  
39     switch(type)  
40     {  
41         case DATA_STRUCTURES_TYPE::AVL:  
42             if(avl == nullptr) return ;  
43             avl->goOff();  
44             break;  
45         default:  
46             break;  
47     }  
48 }
```

7.4.3.36 goOn()

```
void DataStructures::goOn ( )
```

Definition at line 23 of file step.cpp.

```
24 {  
25     switch(type)  
26     {  
27         case DATA_STRUCTURES_TYPE::AVL:  
28             if(avl == nullptr) return ;  
29             avl->goOn();  
30             break;  
31         default:  
32             break;  
33     }  
34 }
```

7.4.3.37 hide()

```
void Object::hide ( ) [inherited]
```

Definition at line 8 of file visible.cpp.

```
9 {  
10     visible = false;  
11 }
```

7.4.3.38 importFromJson() [1/2]

```
void DataStructures::importFromJson ( ) [protected]
```

Definition at line 51 of file constructor.cpp.

```
52 {  
53     json* mem = JSON::readFile(PATH::ATB::DATA_STRUCTURES_+ name + ".json");  
54  
55     if(mem->contains("background"))  
56         initBackground((*mem)["background"]);  
57     if(mem->contains("sprite-structure"))  
58         initLinker((*mem)["sprite-structure"]);  
59     if(mem->contains("display"))  
60         initDisplay((*mem)["display"]);  
61 }
```

7.4.3.39 importFromJson() [2/2]

```
void Object::importFromJson (  
    const json & mem ) [inherited]
```

Definition at line 21 of file constructor.cpp.

```
22 {  
23     if(mem.contains("location"))  
24         locating(mem["location"]);  
25  
26     if(mem.contains("crop"))  
27         cropping(mem["crop"]);  
28  
29     if(mem.contains("color"))  
30         coloring(mem["color"]);  
31  
32     if(mem.contains("shape"))  
33         setShape(mem["shape"]);  
34  
35     if(mem.contains("visible"))  
36         visible = mem["visible"];  
37  
38     if(mem.contains("image"))  
39         textureFromFile(PATH::ASSETS::GRAPHICS_ + mem["image"].get<std::string>());  
40     return ;  
41 }
```


7.4.3.40 init()

```
void DataStructures::init (
    InputBox * inp )
```

Definition at line 14 of file operator.cpp.

```
15 {
16     switch (type)
17     {
18         case DATA_STRUCTURES_TYPE::AVL:
19             initAVL(inp);
20             break;
21         case DATA_STRUCTURES_TYPE::TRIE:
22             initTrie(inp);
23             break;
24         case DATA_STRUCTURES_TYPE::HASH_TABLE:
25             initHashTable(inp);
26             break;
27         case DATA_STRUCTURES_TYPE::MIN_HEAP:
28             initMinHeap(inp);
29             break;
30         case DATA_STRUCTURES_TYPE::MAX_HEAP:
31             initMinHeap(inp);
32             break;
33         case DATA_STRUCTURES_TYPE::BTREE_4TH:
34             //initBTree4th(inp);
35             break;
36         case DATA_STRUCTURES_TYPE::GRAPH:
37             initGraph(inp);
38             break;
39         case DATA_STRUCTURES_TYPE::NONE:
40             break;
41     }
42 }
```

7.4.3.41 initAVL()

```
void DataStructures::initAVL (
    InputBox * inp ) [protected]
```

Definition at line 3 of file init.cpp.

```
4 {
5     std::vector<int> v = NUMBER::stringToArray(inp->getText(1));
6     avl->init(v);
7 }
```

7.4.3.42 initBackground()

```
void DataStructures::initBackground (
    const json & mem ) [protected]
```

Definition at line 14 of file constructor.cpp.

```
15 {
16     Object::importFromJson(mem);
17 }
```

7.4.3.43 initDisplay()

```
void DataStructures::initDisplay (
    const json & mem ) [protected]
```

Definition at line 32 of file constructor.cpp.

```
33 {
34
35     for(auto& i : mem)
36     {
37         SDL_Rect viewport = {0, 0, 0, 0};
38         if(i.contains("viewport"))
39         {
40             viewport.x = i["viewport"]["x"];
41             viewport.y = i["viewport"]["y"];
42             viewport.w = i["viewport"]["w"];
43             viewport.h = i["viewport"]["h"];
44         }
45         displays.push_back(new Display(render, viewport));
46         if(i.contains("name"))
47             displays.back()->linking(i["name"].get<std::string>());
48     }
49 }
```

7.4.3.44 initGraph()

```
void DataStructures::initGraph (
    InputBox * inp ) [protected]
```

Definition at line 3 of file init.cpp.

```
4 {
5     std::vector<std::vector<int> > g;
6
7     g.resize(capacity);
8
9     for(int i = 0; i < capacity; i++)
10    {
11        g[i].resize(capacity);
12        for(int j = 0; j < capacity; j++)
13        {
14            g[i][j] = NUMBER::stringToInt(inp->getText(i * capacity + j + 1));
15        }
16    }
17
18    graph->init(g);
19 }
```

7.4.3.45 initHashTable()

```
void DataStructures::initHashTable (
    InputBox * inp ) [protected]
```

Definition at line 3 of file init.cpp.

```
4 {
5     int HASH_KEY = NUMBER::stringToInt(inp->getText(1));
6     std::vector<int> v = NUMBER::stringToArray(inp->getText(2));
7     hashTable->init(v, HASH_KEY);
8 }
```

7.4.3.46 initLinker()

```
void DataStructures::initLinker (
    const json & mem ) [protected]
```

Definition at line 25 of file constructor.cpp.

```
26 {
27     if (!mem.contains("name"))
28         return ;
29     spriteLinker = mem["name"];
30 }
```

7.4.3.47 initMinHeap()

```
void DataStructures::initMinHeap (
    InputBox * inp ) [protected]
```

Definition at line 3 of file init.cpp.

```
4 {
5     std::vector<int> v = NUMBER::stringToArray(inp->getText(1));
6     minheap->init(v);
7 }
```

7.4.3.48 initTrie()

```
void DataStructures::initTrie (
    InputBox * inp ) [protected]
```

Definition at line 3 of file init.cpp.

```
4 {
5     std::vector<std::string> v = SIUSTRING::split(inp->getText(1));
6     trie->init(v);
7 }
```

7.4.3.49 insert()

```
void DataStructures::insert (
    InputBox * inp )
```

Definition at line 44 of file operator.cpp.

```
45 {
46     switch(type)
47     {
48         case DATA_STRUCTURES_TYPE::AVL:
49             insertAVL(inp);
50             break;
51         case DATA_STRUCTURES_TYPE::TRIE:
52             insertTrie(inp);
53             break;
54         case DATA_STRUCTURES_TYPE::HASH_TABLE:
55             insertHashTable(inp);
56             break;
57         case DATA_STRUCTURES_TYPE::MIN_HEAP:
58             insertMinHeap(inp);
59             break;
60         default:
61             break;
62         case DATA_STRUCTURES_TYPE::MAX_HEAP:
63             insertMinHeap(inp);
64             break;
65         case DATA_STRUCTURES_TYPE::BTREE_4TH:
66             break;
67         case DATA_STRUCTURES_TYPE::GRAPH:
68             break;
69         case DATA_STRUCTURES_TYPE::NONE:
70             break;
71     }
72 }
```

7.4.3.50 insertAVL()

```
void DataStructures::insertAVL (
    InputBox * inp ) [protected]
```

Definition at line 3 of file insert.cpp.

```
4 {
5     int value = NUMBER::stringToInt(inp->getText(1));
6     avl->insert(value);
7 }
```

7.4.3.51 insertHashTable()

```
void DataStructures::insertHashTable (
    InputBox * inp ) [protected]
```

Definition at line 4 of file insert.cpp.

```
5 {
6     int value = NUMBER::stringToInt(inp->getText(1));
7     hashTable->insert(value);
8 }
```

7.4.3.52 insertMinHeap()

```
void DataStructures::insertMinHeap (
    InputBox * inp ) [protected]
```

Definition at line 3 of file insert.cpp.

```
4 {
5     int value = NUMBER::stringToInt(inp->getText(1));
6     minheap->insert(value);
7 }
```

7.4.3.53 insertTrie()

```
void DataStructures::insertTrie (
    InputBox * inp ) [protected]
```

Definition at line 3 of file insert.cpp.

```
4 {
5     std::string value = inp->getText(1);
6     trie->insert(value);
7 }
```

7.4.3.54 isLieInside() [1/4]

```
bool Object::isLieInside (
    int x,
    int y ) [inherited]
```

Definition at line 3 of file locating.cpp.

```
4 {
5     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
6     return (x >= location->x && x < location->x + location->w && y >= location->y && y < location->y +
        location->h);
7 }
```

7.4.3.55 isLieInside() [2/4]

```
bool Object::isLieInside (
    int x,
    int y,
    int w,
    int h ) [inherited]
```

Definition at line 21 of file locating.cpp.

```
22 {
23     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
24     return (x >= location->x && x + w <= location->x + location->w && y >= location->y && y + h <=
        location->y + location->h);
25 }
```

7.4.3.56 isLieInside() [3/4]

```
bool Object::isLieInside (
    SDL_Point p ) [inherited]
```

Definition at line 9 of file locating.cpp.

```
10 {
11     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
12     return (p.x >= location->x && p.x < location->x + location->w && p.y >= location->y && p.y <
        location->y + location->h);
13 }
```

7.4.3.57 isLieInside() [4/4]

```
bool Object::isLieInside (
    SDL_Rect r ) [inherited]
```

Definition at line 15 of file locating.cpp.

```
16 {
17     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
18     return (r.x >= location->x && r.x + r.w <= location->x + location->w && r.y >= location->y && r.y +
        r.h <= location->y + location->h);
19 }
```

7.4.3.58 isReceiveEvent()

```
bool DataStructures::isReceiveEvent (
    SDL_Event & e )
```

Definition at line 6 of file event.cpp.

```
7 {
8     switch(e.type)
9     {
10         case SDL_QUIT:
11             return false;
12             break;
13         default:
14             if(avl != nullptr && avl->isReceiveEvent(e))
15                 return true;
16             if(trie != nullptr && trie->isReceiveEvent(e))
17                 return true;
18             if(hashTable != nullptr && hashTable->isReceiveEvent(e))
19                 return true;
20             if(minheap != nullptr && minheap->isReceiveEvent(e))
21                 return true;
22             if(graph != nullptr && graph->isReceiveEvent(e))
23                 return true;
24             for(auto &i : displays)
25                 if(i->isReceiveEvent(e))
26                     return true;
27             return false;
28             break;
29     }
30 }
```

7.4.3.59 isVisible()

```
bool Object::isVisible ( ) [inherited]
```

Definition at line 13 of file visible.cpp.

```
14 {
15     return visible;
16 }
```

7.4.3.60 linking()

```
void DataStructures::linking (
    std::string n )
```

Definition at line 19 of file constructor.cpp.

```
20 {
21     name = n;
22     importFromJson();
23 }
```

7.4.3.61 locating() [1/3]

```
void Object::locating (
    const json & mem ) [virtual], [inherited]
```

Definition at line 70 of file locating.cpp.

```
71 {
72     if(mem.contains("x") && mem.contains("y") && mem.contains("w") && mem.contains("h"))
73         locating(mem["x"], mem["y"], mem["w"], mem["h"]);
74 }
```

7.4.3.62 locating() [2/3]

```
void Object::locating (
    int x,
    int y,
    int w,
    int h ) [virtual], [inherited]
```

Reimplemented in [Sprite](#).

Definition at line 52 of file locating.cpp.

```
53 {
54     if(location == nullptr) location = new SDL_Rect;
55     location->x = x;
56     location->y = y;
57     location->w = w;
58     location->h = h;
59 }
```

7.4.3.63 locating() [3/3]

```
void Object::locating (
    SDL_Rect l ) [virtual], [inherited]
```

Reimplemented in [Sprite](#).

Definition at line 61 of file locating.cpp.

```
62 {
63     if(location == nullptr) location = new SDL_Rect;
64     location->x = l.x;
65     location->y = l.y;
66     location->w = l.w;
67     location->h = l.h;
68 }
```

7.4.3.64 locatingH()

```
void Object::locatingH (
    int h ) [virtual], [inherited]
```

Definition at line 94 of file locating.cpp.

```
95 {
96     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
97     location->h = h;
98 }
```

7.4.3.65 locatingW()

```
void Object::locatingW (
    int w ) [virtual], [inherited]
```

Definition at line 88 of file locating.cpp.

```
89 {
90     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
91     location->w = w;
92 }
```

7.4.3.66 locatingX()

```
void Object::locatingX (  
    int x )    [virtual], [inherited]
```

Reimplemented in [Sprite](#).

Definition at line 76 of file locating.cpp.

```
77 {  
78     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});  
79     location->x = x;  
80 }
```

7.4.3.67 locatingY()

```
void Object::locatingY (  
    int y )    [virtual], [inherited]
```

Reimplemented in [Sprite](#).

Definition at line 82 of file locating.cpp.

```
83 {  
84     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});  
85     location->y = y;  
86 }
```

7.4.3.68 moveX()

```
void Object::moveX (  
    int delta )    [virtual], [inherited]
```

Reimplemented in [Sprite](#).

Definition at line 100 of file locating.cpp.

```
101 {  
102     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});  
103     location->x += delta;  
104 }
```

7.4.3.69 moveY()

```
void Object::moveY (  
    int delta )    [virtual], [inherited]
```

Reimplemented in [Sprite](#).

Definition at line 106 of file locating.cpp.

```
107 {  
108     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});  
109     location->y += delta;  
110 }
```


7.4.3.70 MST()

```
void DataStructures::MST ( ) [protected]
```

7.4.3.71 mst()

```
void DataStructures::mst ( )
```

Definition at line 3 of file mst.cpp.

```
4 {
5     graph->MST();
6 }
```

7.4.3.72 noCropping()

```
void Object::noCropping ( ) [inherited]
```

Definition at line 32 of file cropping.cpp.

```
33 {
34     if(crop != nullptr) delete crop;
35     crop = nullptr;
36 }
```

7.4.3.73 react()

```
Button * DataStructures::react (
    SDL_Event & e )
```

Definition at line 33 of file event.cpp.

```
34 {
35     Button* but = nullptr;
36     if(avl != nullptr && avl->isReceiveEvent(e))
37         but = avl->react(e);
38     if(but == nullptr && trie != nullptr && trie->isReceiveEvent(e))
39         but = trie->react(e);
40     if(but == nullptr && hashTable != nullptr && hashTable->isReceiveEvent(e))
41         but = hashTable->react(e);
42     if(but == nullptr && minheap != nullptr && minheap->isReceiveEvent(e))
43         but = minheap->react(e);
44     if(but == nullptr && graph != nullptr && graph->isReceiveEvent(e))
45         but = graph->react(e);
46     if(but != nullptr) return but;
47     for(auto &i : displays)
48         if(i->isReceiveEvent(e))
49             but = i->react(e);
50     return but;
51 }
```

7.4.3.74 remove()

```
void DataStructures::remove (
    InputBox * inp )
```

Definition at line 74 of file operator.cpp.

```
75 {
76     switch (type)
77     {
78         case DATA_STRUCTURES_TYPE::AVL:
79             removeAVL(inp);
80             break;
81         case DATA_STRUCTURES_TYPE::TRIE:
82             removeTrie(inp);
83             break;
84         case DATA_STRUCTURES_TYPE::HASH_TABLE:
85             removeHashTable(inp);
86             break;
87         case DATA_STRUCTURES_TYPE::MIN_HEAP:
88             removeMinHeap(inp);
89             break;
90         case DATA_STRUCTURES_TYPE::MAX_HEAP:
91             removeMinHeap(inp);
92             break;
93         case DATA_STRUCTURES_TYPE::BTREE_4TH:
94             break;
95         case DATA_STRUCTURES_TYPE::GRAPH:
96             break;
97         case DATA_STRUCTURES_TYPE::NONE:
98             break;
99     }
100 }
```

7.4.3.75 removeAVL()

```
void DataStructures::removeAVL (
    InputBox * inp ) [protected]
```

Definition at line 3 of file remove.cpp.

```
4 {
5     int value = NUMBER::stringToInt(inp->getText(1));
6
7     avl->remove(value);
8 }
```

7.4.3.76 removeCharacter() [1/2]

```
void Object::removeCharacter ( ) [inherited]
```

Definition at line 28 of file font.cpp.

```
29 {
30     if (text.size() > 0)
31         text.pop_back();
32     textToTexture();
33 }
```

7.4.3.77 removeCharacter() [2/2]

```
void Object::removeCharacter (
    int n ) [inherited]
```

Definition at line 35 of file font.cpp.

```
36 {
37     if (n == 0) return ;
38     if (text.size() <= n) text.clear();
39     else text.erase(text.end() - n, text.end());
40     textToTexture();
41 }
```

7.4.3.78 removeHashTable()

```
void DataStructures::removeHashTable (
    InputBox * inp ) [protected]
```

Definition at line 3 of file remove.cpp.

```
4 {
5     int value = NUMBER::stringToInt(inp->getText(1));
6
7     hashTable->remove(value);
8 }
```

7.4.3.79 removeMinHeap()

```
void DataStructures::removeMinHeap (
    InputBox * inp ) [protected]
```

Definition at line 3 of file pop.cpp.

```
4 {
5     int value = NUMBER::stringToInt(inp->getText(1));
6
7     while(value-->0) minheap->pop();
8 }
```

7.4.3.80 removeTrie()

```
void DataStructures::removeTrie (
    InputBox * inp ) [protected]
```

Definition at line 3 of file remove.cpp.

```
4 {
5     trie->remove(inp->getText(1));
6 }
```

7.4.3.81 rendering()

```
void DataStructures::rendering ( ) [virtual]
```

Reimplemented from [Object](#).

Definition at line 3 of file rendering.cpp.

```
4 {
5     Object::rendering();
6     for(auto& i : node)
7         i->rendering();
8     for(auto& i : displays)
9         i->rendering();
10    if(avl != nullptr)
11        avl->rendering();
12    if(trie != nullptr)
13        trie->rendering();
14    if(hashTable != nullptr)
15        hashTable->rendering();
16    if(minheap != nullptr)
17        minheap->rendering();
18    if(graph != nullptr)
19        graph->rendering();
20 }
```

7.4.3.82 SCC()

```
void DataStructures::SCC ( ) [protected]
```

7.4.3.83 scc()

```
void DataStructures::scc ( )
```

Definition at line 3 of file scc.cpp.

```
4 {
5     graph->SCC();
6 }
```

7.4.3.84 search()

```
void DataStructures::search (
    InputBox * inp )
```

Definition at line 102 of file operator.cpp.

```
103 {
104     switch(type)
105     {
106         case DATA_STRUCTURES_TYPE::AVL:
107             searchAVL(inp);
108             break;
109         case DATA_STRUCTURES_TYPE::TRIE:
110             searchTrie(inp);
111             break;
112         case DATA_STRUCTURES_TYPE::HASH_TABLE:
113             searchHashTable(inp);
114             break;
115         default:
116             break;
117     }
118 }
```

7.4.3.85 searchAVL()

```
void DataStructures::searchAVL (
    InputBox * inp ) [protected]
```

Definition at line 3 of file search.cpp.

```
4 {
5     int value = NUMBER::stringToInt(inp->getText(1));
6
7     avl->search(value);
8 }
```

7.4.3.86 searchHashTable()

```
void DataStructures::searchHashTable (
    InputBox * inp ) [protected]
```

Definition at line 3 of file search.cpp.

```
4 {
5     int value = NUMBER::stringToInt(inp->getText(1));
6
7     hashTable->search(value);
8 }
```

7.4.3.87 searchMinHeap()

```
void DataStructures::searchMinHeap (
    InputBox * inp ) [protected]
```

7.4.3.88 searchTrie()

```
void DataStructures::searchTrie (
    InputBox * inp ) [protected]
```

Definition at line 3 of file search.cpp.

```
4 {
5     trie->search(inp->getText(1));
6 }
```

7.4.3.89 setDataType()

```
void DataStructures::setDataType (
    DATA_STRUCTURES_TYPE t )
```

Definition at line 63 of file constructor.cpp.

```
64 {
65     type = t;
66     switch(type)
67     {
68         case DATA_STRUCTURES_TYPE::AVL:
69             linking("AVL");
70             avl = new AVL(render, ds_mutex, font, {10, 10, 800, 600}, 128);
71             break;
72         case DATA_STRUCTURES_TYPE::TRIE:
73             linking("trie");
74             trie = new Trie(render, ds_mutex, font, {10, 10, 800, 600}, 3000);
75             break;
76         case DATA_STRUCTURES_TYPE::HASH_TABLE:
77             linking("hash_table");
78             hashTable = new HashTable(render, ds_mutex, font, {10, 10, 800, 600}, 128);
79             break;
80         case DATA_STRUCTURES_TYPE::GRAPH:
81             linking("graph");
82             graph = new Graph(render, ds_mutex, font, {10, 10, 800, 600}, 128);
83             break;
84         case DATA_STRUCTURES_TYPE::BTREE_4TH:
85             linking("btree4th");
86             break;
87         case DATA_STRUCTURES_TYPE::MIN_HEAP:
88             linking("minheap");
89             minheap = new minHeap(render, ds_mutex, font, {10, 10, 800, 600}, 128);
90             minheap->setmin();
91             break;
92         case DATA_STRUCTURES_TYPE::MAX_HEAP:
93             linking("minheap");
94             minheap = new minHeap(render, ds_mutex, font, {10, 10, 800, 600}, 128);
95             minheap->setmax();
96             break;
97         case DATA_STRUCTURES_TYPE::NONE:
98             break;
99     }
100 }
```

7.4.3.90 setFont()

```
void Object::setFont (
    TTF_Font * f ) [inherited]
```

Definition at line 4 of file font.cpp.

```
5 {
6     font = f;
7     textToTexture();
8 }
```

7.4.3.91 setShape()

```
void Object::setShape (
    const json & mem ) [inherited]
```

Definition at line 52 of file shape.cpp.

```
53 {
54     if(mem["type"].get<std::string>() == "CIRCLE")
55     {
56         if(mem.contains("center"))
57         {
```

```

58         if(mem.contains("radius"))
59             changeToCircle(mem["center"]["x"], mem["center"]["y"], mem["radius"]);
60         else changeToCircle(mem["center"]["x"], mem["center"]["y"]);
61     }else changeToCircle();
62
63     return ;
64 }
65
66 if(mem["type"].get<std::string>() == "NONE" || mem["type"].get<std::string>() == "RECTANGLE")
67 {
68     changeToRectangle();
69     return ;
70 }
71
72 }

```

7.4.3.92 setText()

```

void Object::setText (
    std::string t ) [inherited]

```

Definition at line 10 of file font.cpp.

```

11 {
12     text = t;
13     textToTexture();
14 }

```

7.4.3.93 setting()

```

void DataStructures::setting (
    InputBox * inp )

```

Definition at line 152 of file operator.cpp.

```

153 {
154     switch(type)
155     {
156         case DATA_STRUCTURES_TYPE::AVL:
157             settingAVL(inp);
158             break;
159         case DATA_STRUCTURES_TYPE::TRIE:
160             settingTrie(inp);
161             break;
162         case DATA_STRUCTURES_TYPE::GRAPH:
163             settingGraph(inp);
164             break;
165         case DATA_STRUCTURES_TYPE::MAX_HEAP:
166             settingMinHeap(inp);
167             break;
168         case DATA_STRUCTURES_TYPE::MIN_HEAP:
169             settingMinHeap(inp);
170             break;
171         case DATA_STRUCTURES_TYPE::HASH_TABLE:
172             settingHashTable(inp);
173             break;
174         case DATA_STRUCTURES_TYPE::BTREE_4TH:
175             break;
176         case DATA_STRUCTURES_TYPE::NONE:
177             break;
178     }
179 }

```

7.4.3.94 settingAVL()

```
void DataStructures::settingAVL (
    InputBox * inp ) [protected]
```

Definition at line 5 of file setting.cpp.

```
6 {
7     std::vector<int> v1 = NUMBER::stringToArray(inp->getText(1));
8     std::vector<int> v2 = NUMBER::stringToArray(inp->getText(2));
9     std::vector<int> v3 = NUMBER::stringToArray(inp->getText(3));
10    std::vector<int> v4 = NUMBER::stringToArray(inp->getText(4));
11
12    SDL_Color c1;
13    c1.r = v1[0];
14    c1.g = v1[1];
15    c1.b = v1[2];
16    if(v1.size() >= 4) c1.a = v1[3];
17    else c1.a = 255;
18
19    SDL_Color c2;
20    c2.r = v2[0];
21    c2.g = v2[1];
22    c2.b = v2[2];
23    if(v2.size() >= 4) c2.a = v2[3];
24    else c2.a = 255;
25
26    SDL_Color c3;
27    c3.r = v3[0];
28    c3.g = v3[1];
29    c3.b = v3[2];
30    if(v3.size() >= 4) c3.a = v3[3];
31    else c3.a = 255;
32
33    SDL_Color c4;
34    c4.r = v4[0];
35    c4.g = v4[1];
36    c4.b = v4[2];
37    if(v4.size() >= 4) c4.a = v4[3];
38    else c4.a = 255;
39
40    Object::coloring(c1);
41    Object::fillWithColor();
42
43    avl->setting(c1, c2, c3, c4);
44 }
```

7.4.3.95 settingGraph()

```
void DataStructures::settingGraph (
    InputBox * inp ) [protected]
```

Definition at line 5 of file setting.cpp.

```
6 {
7     std::vector<int> v1 = NUMBER::stringToArray(inp->getText(1));
8     std::vector<int> v2 = NUMBER::stringToArray(inp->getText(2));
9     std::vector<int> v3 = NUMBER::stringToArray(inp->getText(3));
10    std::vector<int> v4 = NUMBER::stringToArray(inp->getText(4));
11
12    SDL_Color c1;
13    c1.r = v1[0];
14    c1.g = v1[1];
15    c1.b = v1[2];
16    if(v1.size() >= 4) c1.a = v1[3];
17    else c1.a = 255;
18
19    SDL_Color c2;
20    c2.r = v2[0];
21    c2.g = v2[1];
22    c2.b = v2[2];
23    if(v2.size() >= 4) c2.a = v2[3];
24    else c2.a = 255;
25
26    SDL_Color c3;
27    c3.r = v3[0];
28    c3.g = v3[1];
```



```

29     c3.b = v3[2];
30     if(v3.size() >= 4) c3.a = v3[3];
31     else c3.a = 255;
32
33     SDL_Color c4;
34     c4.r = v4[0];
35     c4.g = v4[1];
36     c4.b = v4[2];
37     if(v4.size() >= 4) c4.a = v4[3];
38     else c4.a = 255;
39
40     Object::coloring(c1);
41     Object::fillWithColor();
42
43     graph->setting(c1, c2, c3, c4);
44 }

```

7.4.3.96 settingHashTable()

```

void DataStructures::settingHashTable (
    InputBox * inp ) [protected]

```

Definition at line 5 of file setting.cpp.

```

6 {
7     std::vector<int> v1 = NUMBER::stringToArray(inp->getText(1));
8     std::vector<int> v2 = NUMBER::stringToArray(inp->getText(2));
9     std::vector<int> v3 = NUMBER::stringToArray(inp->getText(3));
10    std::vector<int> v4 = NUMBER::stringToArray(inp->getText(4));
11
12    SDL_Color c1;
13    c1.r = v1[0];
14    c1.g = v1[1];
15    c1.b = v1[2];
16    if(v1.size() >= 4) c1.a = v1[3];
17    else c1.a = 255;
18
19    SDL_Color c2;
20    c2.r = v2[0];
21    c2.g = v2[1];
22    c2.b = v2[2];
23    if(v2.size() >= 4) c2.a = v2[3];
24    else c2.a = 255;
25
26    SDL_Color c3;
27    c3.r = v3[0];
28    c3.g = v3[1];
29    c3.b = v3[2];
30    if(v3.size() >= 4) c3.a = v3[3];
31    else c3.a = 255;
32
33    SDL_Color c4;
34    c4.r = v4[0];
35    c4.g = v4[1];
36    c4.b = v4[2];
37    if(v4.size() >= 4) c4.a = v4[3];
38    else c4.a = 255;
39
40    Object::coloring(c1);
41    Object::fillWithColor();
42
43    hashTable->setting(c1, c2, c3, c4);
44 }

```

7.4.3.97 settingMinHeap()

```

void DataStructures::settingMinHeap (
    InputBox * inp ) [protected]

```

Definition at line 5 of file setting.cpp.

```

6 {
7     std::vector<int> v1 = NUMBER::stringToArray(inp->getText(1));
8     std::vector<int> v2 = NUMBER::stringToArray(inp->getText(2));
9     std::vector<int> v3 = NUMBER::stringToArray(inp->getText(3));
10    std::vector<int> v4 = NUMBER::stringToArray(inp->getText(4));
11
12    SDL_Color c1;
13    c1.r = v1[0];
14    c1.g = v1[1];
15    c1.b = v1[2];
16    if(v1.size() >= 4) c1.a = v1[3];
17    else c1.a = 255;
18
19    SDL_Color c2;
20    c2.r = v2[0];
21    c2.g = v2[1];
22    c2.b = v2[2];
23    if(v2.size() >= 4) c2.a = v2[3];
24    else c2.a = 255;
25
26    SDL_Color c3;
27    c3.r = v3[0];
28    c3.g = v3[1];
29    c3.b = v3[2];
30    if(v3.size() >= 4) c3.a = v3[3];
31    else c3.a = 255;
32
33    SDL_Color c4;
34    c4.r = v4[0];
35    c4.g = v4[1];
36    c4.b = v4[2];
37    if(v4.size() >= 4) c4.a = v4[3];
38    else c4.a = 255;
39
40    Object::coloring(c1);
41    Object::fillWithColor();
42
43    minheap->setting(c1, c2, c3, c4);
44 }

```

7.4.3.98 settingTrie()

```

void DataStructures::settingTrie (
    InputBox * inp ) [protected]

```

Definition at line 5 of file setting.cpp.

```

6 {
7     std::vector<int> v1 = NUMBER::stringToArray(inp->getText(1));
8     std::vector<int> v2 = NUMBER::stringToArray(inp->getText(2));
9     std::vector<int> v3 = NUMBER::stringToArray(inp->getText(3));
10    std::vector<int> v4 = NUMBER::stringToArray(inp->getText(4));
11
12    SDL_Color c1;
13    c1.r = v1[0];
14    c1.g = v1[1];
15    c1.b = v1[2];
16    if(v1.size() >= 4) c1.a = v1[3];
17    else c1.a = 255;
18
19    SDL_Color c2;
20    c2.r = v2[0];
21    c2.g = v2[1];
22    c2.b = v2[2];
23    if(v2.size() >= 4) c2.a = v2[3];
24    else c2.a = 255;
25
26    SDL_Color c3;
27    c3.r = v3[0];
28    c3.g = v3[1];
29    c3.b = v3[2];
30    if(v3.size() >= 4) c3.a = v3[3];
31    else c3.a = 255;
32
33    SDL_Color c4;
34    c4.r = v4[0];
35    c4.g = v4[1];
36    c4.b = v4[2];
37    if(v4.size() >= 4) c4.a = v4[3];

```

```
38     else c4.a = 255;
39
40     Object::coloring(c1);
41     Object::fillWithColor();
42
43     trie->setting(c1, c2, c3, c4);
44 }
```

7.4.3.99 show()

```
void Object::show ( ) [inherited]
```

Definition at line 3 of file visible.cpp.

```
4 {
5     visible = true;
6 }
```

7.4.3.100 size()

```
void DataStructures::size ( )
```

Definition at line 137 of file operator.cpp.

```
138 {
139     switch(type)
140     {
141         case DATA_STRUCTURES_TYPE::MIN_HEAP:
142             minheap->size();
143             break;
144         case DATA_STRUCTURES_TYPE::MAX_HEAP:
145             minheap->size();
146             break;
147         default:
148             break;
149     }
150 }
```

7.4.3.101 slowDown()

```
void DataStructures::slowDown ( )
```

Definition at line 65 of file step.cpp.

```
66 {
67     switch(type)
68     {
69         case DATA_STRUCTURES_TYPE::AVL:
70             if(avl == nullptr) return ;
71             avl->slowDown();
72             break;
73         default:
74             break;
75     }
76 }
```

7.4.3.102 speedUp()

void DataStructures::speedUp ()

Definition at line 51 of file step.cpp.

```
52 {  
53     switch(type)  
54     {  
55         case DATA_STRUCTURES_TYPE::AVL:  
56             if(avl == nullptr) return ;  
57             avl->speedUp();  
58             break;  
59         default:  
60             break;  
61     }  
62 }
```

7.4.3.103 textToTexture()

void Object::textToTexture () [protected], [inherited]

Definition at line 43 of file font.cpp.

```
44 {  
45     if(font == nullptr) return ;  
46     if(color == nullptr) return ;  
47     if(render == nullptr) return ;  
48     if(texture != nullptr)  
49     {  
50         SDL_DestroyTexture(texture);  
51     }  
52     texture = nullptr;  
53  
54     SDL_Surface* surface = TTF_RenderText_Blended(font, text.c_str(), *color);  
55  
56     if(surface == nullptr) return ;  
57  
58     texture = SDL_CreateTextureFromSurface(render, surface);  
59     SDL_FreeSurface(surface);  
60     fitTheTexture();  
61 }
```

7.4.3.104 textureFromFile()

void Object::textureFromFile (
 std::string dir) [inherited]

Definition at line 4 of file external_storage.cpp.

```
5 {  
6     SDL_Surface *surface = IMG_Load(dir.c_str());  
7  
8     texture = SDL_CreateTextureFromSurface(render, surface);  
9     SDL_FreeSurface(surface);  
10 }
```

7.4.3.105 top()

```
void DataStructures::top ( )
```

Definition at line 120 of file operator.cpp.

```
121 {  
122     switch(type)  
123     {  
124         case DATA_STRUCTURES_TYPE::MIN_HEAP:  
125             minheap->top();  
126             break;  
127         case DATA_STRUCTURES_TYPE::MAX_HEAP:  
128             minheap->top();  
129             break;  
130             break;  
131         default:  
132             break;  
133     }  
134 }
```

7.4.3.106 zoom()

```
void Object::zoom (  
    double delta ) [virtual], [inherited]
```

Definition at line 123 of file locating.cpp.

```
124 {  
125     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});  
126     location->w *= delta;  
127     location->h *= delta;  
128 }
```

7.4.3.107 zoomH()

```
void Object::zoomH (  
    int delta ) [virtual], [inherited]
```

Definition at line 118 of file locating.cpp.

```
119 {  
120     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});  
121     location->h += delta;  
122 }
```

7.4.3.108 zoomInMiddle()

```
void Object::zoomInMiddle (  
    double delta ) [virtual], [inherited]
```

Definition at line 130 of file locating.cpp.

```
131 {  
132     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});  
133     SDL_Point center = {location->x + location->w / 2, location->y + location->h / 2};  
134     location->w *= delta;  
135     location->h *= delta;  
136     location->x = center.x - location->w / 2;  
137     location->y = center.y - location->h / 2;  
138 }
```

7.4.3.109 zoomW()

```
void Object::zoomW (
    int delta )    [virtual], [inherited]
```

Definition at line 112 of file locating.cpp.

```
113 {
114     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
115     location->w += delta;
116 }
```

7.4.4 Member Data Documentation

7.4.4.1 capacity

```
int DataStructures::capacity
```

Definition at line 130 of file data_structures.hpp.

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

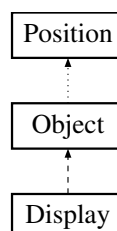
- [include/data_structures.hpp](#)
- [src/data_structures/constructor.cpp](#)
- [src/data_structures/destructor.cpp](#)
- [src/data_structures/event.cpp](#)
- [src/data_structures/operator/AVL/init.cpp](#)
- [src/data_structures/operator/AVL/insert.cpp](#)
- [src/data_structures/operator/AVL/remove.cpp](#)
- [src/data_structures/operator/AVL/search.cpp](#)
- [src/data_structures/operator/AVL/setting.cpp](#)
- [src/data_structures/operator/graph/dijkstra.cpp](#)
- [src/data_structures/operator/graph/mst.cpp](#)
- [src/data_structures/operator/graph/scc.cpp](#)
- [src/data_structures/operator/minheap/pop.cpp](#)
- [src/data_structures/operator.cpp](#)
- [src/data_structures/rendering.cpp](#)
- [src/data_structures/step.cpp](#)

7.5 Display Class Reference

container of button intermediate between button and user input, window

```
#include <display.hpp>
```

Inheritance diagram for Display:



Public Member Functions

- [Display](#) (SDL_Renderer *r, SDL_Rect v)
- [~Display](#) ()
- void [linking](#) (std::string n)
- bool [isReceiveEvent](#) (SDL_Event &e)
- [Button](#) * [react](#) (SDL_Event &e)
- void [rendering](#) ()

Protected Member Functions

- void [initBackground](#) (const [json](#) &mem)
- void [initButtons](#) (const [json](#) &mem)
- void [importFromJson](#) ()
- bool [isButtonReceiveEvent](#) (SDL_Event &e)
- void [fillWithColor](#) ()
- void [fillCircleByColor](#) ()
- void [fillRectangleByColor](#) ()
- void [textToTexture](#) ()
- const SDL_Rect * [getCrop](#) ()
- void [cropping](#) (int x, int y, int w, int h)
- void [cropping](#) (SDL_Rect c)
- void [cropping](#) (const [json](#) &mem)
- void [noCropping](#) ()
- const SDL_Rect * [getLocation](#) ()
- int [getX](#) ()
- int [getY](#) ()
- int [getW](#) ()
- int [getH](#) ()
- virtual void [locating](#) (int x, int y, int w, int h)
- virtual void [locating](#) (SDL_Rect l)
- virtual void [locating](#) (const [json](#) &mem)
- virtual void [locatingX](#) (int x)
- virtual void [locatingY](#) (int y)
- virtual void [locatingW](#) (int w)
- virtual void [locatingH](#) (int h)
- virtual void [moveX](#) (int delta)
- virtual void [moveY](#) (int delta)
- virtual void [zoomW](#) (int delta)
- virtual void [zoomH](#) (int delta)
- virtual void [zoom](#) (double delta)
- virtual void [zoomInMiddle](#) (double delta)
- void [fitTheTexture](#) ()
- const SDL_Color * [getColor](#) ()
- void [coloring](#) (int r, int g, int b, int a)
- void [coloring](#) (SDL_Color c)
- void [coloring](#) (const [json](#) &mem)
- void [textureFromFile](#) (std::string dir)
- void [changeToCircle](#) ()
- void [changeToCircle](#) (SDL_Point c)
- void [changeToCircle](#) (int x, int y)
- void [changeToCircle](#) (SDL_Point c, int r)
- void [changeToCircle](#) (int x, int y, int r)
- void [changeToRectangle](#) ()

- void [setShape](#) (const [json](#) &mem)
- bool [isLielInside](#) (int x, int y)
- bool [isLielInside](#) (SDL_Point p)
- bool [isLielInside](#) (SDL_Rect r)
- bool [isLielInside](#) (int x, int y, int w, int h)
- void [show](#) ()
- void [hide](#) ()
- bool [isVisible](#) ()
- void [importFromJson](#) (const [json](#) &mem)
- void [setFont](#) (TTF_Font *f)
- void [setText](#) (std::string t)
- void [addText](#) (std::string t)
- void [addCharacter](#) (char c)
- void [removeCharacter](#) ()
- void [removeCharacter](#) (int n)
- std::string [getText](#) ()
- int [getSize](#) ()

7.5.1 Detailed Description

container of button intermediate between button and user input, window

Drawable

Definition at line 19 of file display.hpp.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 Display()

```
Display::Display (
    SDL_Renderer * r,
    SDL_Rect v )
```

Definition at line 4 of file constructor.cpp.

```
4                                     : Object (r)
5 {
6     render = r;
7     viewport = v;
8 }
```

7.5.2.2 ~Display()

```
Display::~Display ( )
```

Definition at line 3 of file destructor.cpp.

```
4 {
5
6     return ;
7 }
```


7.5.3 Member Function Documentation

7.5.3.1 addCharacter()

```
void Object::addCharacter (
    char c ) [inherited]
```

Definition at line 22 of file font.cpp.

```
23 {
24     text += c;
25     textToTexture();
26 }
```

7.5.3.2 addText()

```
void Object::addText (
    std::string t ) [inherited]
```

Definition at line 16 of file font.cpp.

```
17 {
18     text += t;
19     textToTexture();
20 }
```

7.5.3.3 changeToCircle() [1/5]

```
void Object::changeToCircle ( ) [inherited]
```

Definition at line 5 of file shape.cpp.

```
6 {
7     shapeType = SHAPE::CIRCLE;
8     radius = std::min(getW(), getH()) / 2;
9
10    center.x = getX() + getW() / 2;
11    center.y = getY() + getH() / 2;
12    fillCircleByColor();
13 }
```

7.5.3.4 changeToCircle() [2/5]

```
void Object::changeToCircle (
    int x,
    int y ) [inherited]
```

Definition at line 24 of file shape.cpp.

```
25 {
26     changeToCircle({x, y});
27 }
```

7.5.3.5 changeToCircle() [3/5]

```
void Object::changeToCircle (
    int x,
    int y,
    int r ) [inherited]
```

Definition at line 37 of file shape.cpp.

```
38 {
39     shapeType = SHAPE::CIRCLE;
40     radius = r;
41     center.x = x;
42     center.y = y;
43     fillCircleByColor();
44 }
```

7.5.3.6 changeToCircle() [4/5]

```
void Object::changeToCircle (
    SDL_Point c ) [inherited]
```

Definition at line 15 of file shape.cpp.

```
16 {
17     shapeType = SHAPE::CIRCLE;
18     center = c;
19     radius = std::min(getW() - c.x, c.x - getX());
20     radius = std::min(radius, std::min(getH() - c.y, c.y - getY()));
21     fillCircleByColor();
22 }
```

7.5.3.7 changeToCircle() [5/5]

```
void Object::changeToCircle (
    SDL_Point c,
    int r ) [inherited]
```

Definition at line 29 of file shape.cpp.

```
30 {
31     shapeType = SHAPE::CIRCLE;
32     radius = r;
33     center = c;
34     fillCircleByColor();
35 }
```

7.5.3.8 changeToRectangle()

```
void Object::changeToRectangle ( ) [inherited]
```

Definition at line 46 of file shape.cpp.

```
47 {
48     shapeType = SHAPE::RECTANGLE;
49     fillRectangleByColor();
50 }
```

7.5.3.9 coloring() [1/3]

```
void Object::coloring (
    const json & mem ) [inherited]
```

Definition at line 30 of file coloring.cpp.

```
31 {
32     if(mem.contains("r") && mem.contains("g") && mem.contains("b"))
33     {
34         if(mem.contains("a")) coloring(mem["r"], mem["g"], mem["b"], mem["a"]);
35     else coloring(mem["r"], mem["g"], mem["b"], 255);
36     }
37 }
```

7.5.3.10 coloring() [2/3]

```
void Object::coloring (
    int r,
    int g,
    int b,
    int a ) [inherited]
```

Definition at line 8 of file coloring.cpp.

```
9 {
10     if(color == nullptr) color = new SDL_Color;
11     color->r = r;
12     color->g = g;
13     color->b = b;
14     color->a = a;
15
16     fillWithColor();
17 }
```

7.5.3.11 coloring() [3/3]

```
void Object::coloring (
    SDL_Color c ) [inherited]
```

Definition at line 19 of file coloring.cpp.

```
20 {
21     if(color == nullptr) color = new SDL_Color;
22     color->r = c.r;
23     color->g = c.g;
24     color->b = c.b;
25     color->a = c.a;
26
27     fillWithColor();
28 }
```

7.5.3.12 cropping() [1/3]

```
void Object::cropping (
    const json & mem ) [inherited]
```

Definition at line 26 of file cropping.cpp.

```
27 {
28     if(mem.contains("x") && mem.contains("y") && mem.contains("w") && mem.contains("h"))
29         cropping(mem["x"], mem["y"], mem["w"], mem["h"]);
30 }
```

7.5.3.13 cropping() [2/3]

```
void Object::cropping (
    int x,
    int y,
    int w,
    int h ) [inherited]
```

Definition at line 8 of file cropping.cpp.

```
9 {
10     if(crop == nullptr) crop = new SDL_Rect;
11     crop->x = x;
12     crop->y = y;
13     crop->w = w;
14     crop->h = h;
15 }
```

7.5.3.14 cropping() [3/3]

```
void Object::cropping (
    SDL_Rect c ) [inherited]
```

Definition at line 17 of file cropping.cpp.

```
18 {
19     if(crop == nullptr) crop = new SDL_Rect;
20     crop->x = c.x;
21     crop->y = c.y;
22     crop->w = c.w;
23     crop->h = c.h;
24 }
```

7.5.3.15 fillCircleByColor()

```
void Object::fillCircleByColor ( ) [protected], [inherited]
```

Definition at line 91 of file shape.cpp.

```
92 {
93     if(location == nullptr) locating(0, 0, 0, 0);
94
95     if(texture != nullptr) SDL_DestroyTexture(texture);
96     texture = nullptr;
97
98     Uint32 rmask, gmask, bmask, amask;
99     Uint32 pixelColor;
100     #if SDL_BYTEORDER == SDL_BIG_ENDIAN
101         rmask = 0xff000000;
102         gmask = 0x00ff0000;
103         bmask = 0x0000ff00;
104         amask = 0x000000ff;
105         pixelColor = (color->r << 24) | (color->g << 16) | (color->b << 8) | color->a;
106     #else
107         rmask = 0x000000ff;
108         gmask = 0x0000ff00;
109         bmask = 0x00ff0000;
110         amask = 0xff000000;
111         pixelColor = (color->a << 24) | (color->b << 16) | (color->g << 8) | color->r;
112     #endif
113
114     SDL_Surface *surf = SDL_CreateRGBSurface(0, getW(), getH(), 32, rmask, gmask, bmask, amask);
115     SDL_SetSurfaceBlendMode(surf, SDL_BLENDMODE_BLEND);
116
117     texture = SDL_CreateTextureFromSurface(render, surf);
118     SDL_FreeSurface(surf);
119
120     Uint32 *pixels = new Uint32[getW() * getH()];
```

```

121     memset(pixels, 0, getWidth() * getHeight() * sizeof(Uint32));
122
123     SDL_Point p = {getWidth() / 2, getHeight() / 2};
124     center = p;
125
126     if(radius > std::min(getWidth(), getHeight()) / 2) radius = std::min(getWidth(), getHeight()) / 2;
127
128     for(int i = p.x - radius; i <= p.x + radius; i++)
129         for(int j = p.y - radius; j <= p.y + radius; j++)
130             if((i - p.x) * (i - p.x) + (j - p.y) * (j - p.y) <= radius * radius)
131                 {
132                     int index = i * getWidth() + j;
133                     if(index < 0 || index >= getWidth() * getHeight()) continue;
134                     pixels[index] = pixelColor;
135                 }
136
137     SDL_UpdateTexture(texture, nullptr, pixels, getWidth() * sizeof(Uint32));
138     delete[] pixels;
139 }

```

7.5.3.16 fillRectangleByColor()

void Object::fillRectangleByColor () [protected], [inherited]

Definition at line 74 of file shape.cpp.

```

75 {
76     if(location == nullptr) locating(0, 0, 0, 0);
77
78     if(texture != nullptr) SDL_DestroyTexture(texture);
79     texture = nullptr;
80
81     SDL_Surface* surf = SDL_CreateRGBSurfaceWithFormat(0, getWidth(), getHeight(), 32, SDL_PIXELFORMAT_RGBA32);
82
83     SDL_SetSurfaceBlendMode(surf, SDL_BLENDMODE_BLEND);
84
85     SDL_FillRect(surf, nullptr, SDL_MapRGBA(surf->format, color->r, color->g, color->b, color->a));
86
87     texture = SDL_CreateTextureFromSurface(render, surf);
88     SDL_FreeSurface(surf);
89 }

```

7.5.3.17 fillWithColor()

void Object::fillWithColor () [protected], [inherited]

Definition at line 39 of file coloring.cpp.

```

40 {
41     if(shapeType == SHAPE::NONE) return fillRectangleByColor();
42     if(shapeType == SHAPE::RECTANGLE) return fillRectangleByColor();
43     if(shapeType == SHAPE::CIRCLE) return fillCircleByColor();
44 }

```

7.5.3.18 fitTheTexture()

void Object::fitTheTexture () [inherited]

Definition at line 140 of file locating.cpp.

```

141 {
142     if(texture == nullptr) return;
143     SDL_QueryTexture(texture, nullptr, nullptr, &location->w, &location->h);
144 }

```

7.5.3.19 getColor()

```
const SDL_Color * Object::getColor ( ) [inherited]
```

Definition at line 3 of file coloring.cpp.

```
4 {  
5     return color;  
6 }
```

7.5.3.20 getCrop()

```
const SDL_Rect * Object::getCrop ( ) [inherited]
```

Definition at line 3 of file cropping.cpp.

```
4 {  
5     return crop;  
6 }
```

7.5.3.21 getH()

```
int Object::getH ( ) [inherited]
```

Definition at line 47 of file locating.cpp.

```
48 {  
49     return location->h;  
50 }
```

7.5.3.22 getLocation()

```
const SDL_Rect * Object::getLocation ( ) [inherited]
```

Definition at line 27 of file locating.cpp.

```
28 {  
29     return location;  
30 }
```

7.5.3.23 getSize()

```
int Object::getSize ( ) [inherited]
```

Definition at line 68 of file font.cpp.

```
69 {  
70     return text.size();  
71 }
```

7.5.3.24 getText()

```
std::string Object::getText ( ) [inherited]
```

Definition at line 63 of file font.cpp.

```
64 {  
65     return text;  
66 }
```

7.5.3.25 getW()

```
int Object::getW ( ) [inherited]
```

Definition at line 42 of file locating.cpp.

```
43 {  
44     return location->w;  
45 }
```

7.5.3.26 getX()

```
int Object::getX ( ) [inherited]
```

Definition at line 32 of file locating.cpp.

```
33 {  
34     return location->x;  
35 }
```

7.5.3.27 getY()

```
int Object::getY ( ) [inherited]
```

Definition at line 37 of file locating.cpp.

```
38 {  
39     return location->y;  
40 }
```

7.5.3.28 hide()

```
void Object::hide ( ) [inherited]
```

Definition at line 8 of file visible.cpp.

```
9 {  
10     visible = false;  
11 }
```

7.5.3.29 importFromJson() [1/2]

```
void Display::importFromJson ( ) [protected]
```

Definition at line 25 of file constructor.cpp.

```
26 {
27     json* mem = JSON::readFile(PATH::ATB::DISPLAY_ + name + ".json");
28
29     if(mem->contains("background"))
30         initBackground((*mem)["background"]);
31     if(mem->contains("buttons"))
32         initButtons((*mem)["buttons"]);
33     delete mem;
34 }
```

7.5.3.30 importFromJson() [2/2]

```
void Object::importFromJson (
    const json & mem ) [inherited]
```

Definition at line 21 of file constructor.cpp.

```
22 {
23     if(mem.contains("location"))
24         locating(mem["location"]);
25
26     if(mem.contains("crop"))
27         cropping(mem["crop"]);
28
29     if(mem.contains("color"))
30         coloring(mem["color"]);
31
32     if(mem.contains("shape"))
33         setShape(mem["shape"]);
34
35     if(mem.contains("visible"))
36         visible = mem["visible"];
37
38     if(mem.contains("image"))
39         textureFromFile(PATH::ASSETS::GRAPHICS_ + mem["image"].get<std::string>());
40     return ;
41 }
```

7.5.3.31 initBackground()

```
void Display::initBackground (
    const json & mem ) [protected]
```

Definition at line 10 of file constructor.cpp.

```
11 {
12     Object::importFromJson(mem);
13 }
```


7.5.3.32 initButtons()

```
void Display::initButtons (
    const json & mem ) [protected]
```

Definition at line 15 of file constructor.cpp.

```
16 {
17     for(auto& i : mem)
18     {
19         Button* b = new Button(render);
20         b->linking(i["name"].get<std::string>());
21         buts.push_back(b);
22     }
23 }
```

7.5.3.33 isButtonReceiveEvent()

```
bool Display::isButtonReceiveEvent (
    SDL_Event & e ) [protected]
```

Definition at line 3 of file event.cpp.

```
4 {
5     for(auto& but : buts)
6         if(but->isReceiveEvent(e)) return true;
7     return false;
8 }
```

7.5.3.34 isLieInside() [1/4]

```
bool Object::isLieInside (
    int x,
    int y ) [inherited]
```

Definition at line 3 of file locating.cpp.

```
4 {
5     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
6     return (x >= location->x && x < location->x + location->w && y >= location->y && y < location->y +
7         location->h);
8 }
```

7.5.3.35 isLieInside() [2/4]

```
bool Object::isLieInside (
    int x,
    int y,
    int w,
    int h ) [inherited]
```

Definition at line 21 of file locating.cpp.

```
22 {
23     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
24     return (x >= location->x && x + w <= location->x + location->w && y >= location->y && y + h <=
25         location->y + location->h);
26 }
```

7.5.3.36 isLieInside() [3/4]

```
bool Object::isLieInside (
    SDL_Point p ) [inherited]
```

Definition at line 9 of file locating.cpp.

```
10 {
11     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
12     return (p.x >= location->x && p.x < location->x + location->w && p.y >= location->y && p.y <
        location->y + location->h);
13 }
```

7.5.3.37 isLieInside() [4/4]

```
bool Object::isLieInside (
    SDL_Rect r ) [inherited]
```

Definition at line 15 of file locating.cpp.

```
16 {
17     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
18     return (r.x >= location->x && r.x + r.w <= location->x + location->w && r.y >= location->y && r.y +
        r.h <= location->y + location->h);
19 }
```

7.5.3.38 isReceiveEvent()

```
bool Display::isReceiveEvent (
    SDL_Event & e )
```

Definition at line 10 of file event.cpp.

```
11 {
12
13     switch(e.type)
14     {
15         case SDL_QUIT:
16             return false;
17             break;
18         default:
19             if(isButtonReceiveEvent(e)) return true;
20             return false;
21             break;
22     }
23 }
```

7.5.3.39 isVisible()

```
bool Object::isVisible ( ) [inherited]
```

Definition at line 13 of file visible.cpp.

```
14 {
15     return visible;
16 }
```

7.5.3.40 linking()

```
void Display::linking (
    std::string n )
```

Definition at line 37 of file constructor.cpp.

```
38 {
39     name = n;
40     importFromJson();
41 }
```

7.5.3.41 locating() [1/3]

```
void Object::locating (
    const json & mem ) [virtual], [inherited]
```

Definition at line 70 of file locating.cpp.

```
71 {
72     if(mem.contains("x") && mem.contains("y") && mem.contains("w") && mem.contains("h"))
73         locating(mem["x"], mem["y"], mem["w"], mem["h"]);
74 }
```

7.5.3.42 locating() [2/3]

```
void Object::locating (
    int x,
    int y,
    int w,
    int h ) [virtual], [inherited]
```

Reimplemented in [Sprite](#).

Definition at line 52 of file locating.cpp.

```
53 {
54     if(location == nullptr) location = new SDL_Rect;
55     location->x = x;
56     location->y = y;
57     location->w = w;
58     location->h = h;
59 }
```

7.5.3.43 locating() [3/3]

```
void Object::locating (
    SDL_Rect l ) [virtual], [inherited]
```

Reimplemented in [Sprite](#).

Definition at line 61 of file locating.cpp.

```
62 {
63     if(location == nullptr) location = new SDL_Rect;
64     location->x = l.x;
65     location->y = l.y;
66     location->w = l.w;
67     location->h = l.h;
68 }
```

7.5.3.44 locatingH()

```
void Object::locatingH (  
    int h )    [virtual], [inherited]
```

Definition at line 94 of file locating.cpp.

```
95 {  
96     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});  
97     location->h = h;  
98 }
```

7.5.3.45 locatingW()

```
void Object::locatingW (  
    int w )    [virtual], [inherited]
```

Definition at line 88 of file locating.cpp.

```
89 {  
90     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});  
91     location->w = w;  
92 }
```

7.5.3.46 locatingX()

```
void Object::locatingX (  
    int x )    [virtual], [inherited]
```

Reimplemented in [Sprite](#).

Definition at line 76 of file locating.cpp.

```
77 {  
78     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});  
79     location->x = x;  
80 }
```

7.5.3.47 locatingY()

```
void Object::locatingY (  
    int y )    [virtual], [inherited]
```

Reimplemented in [Sprite](#).

Definition at line 82 of file locating.cpp.

```
83 {  
84     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});  
85     location->y = y;  
86 }
```

7.5.3.48 moveX()

```
void Object::moveX (
    int delta ) [virtual], [inherited]
```

Reimplemented in [Sprite](#).

Definition at line 100 of file locating.cpp.

```
101 {
102     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
103     location->x += delta;
104 }
```

7.5.3.49 moveY()

```
void Object::moveY (
    int delta ) [virtual], [inherited]
```

Reimplemented in [Sprite](#).

Definition at line 106 of file locating.cpp.

```
107 {
108     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
109     location->y += delta;
110 }
```

7.5.3.50 noCropping()

```
void Object::noCropping ( ) [inherited]
```

Definition at line 32 of file cropping.cpp.

```
33 {
34     if(crop != nullptr) delete crop;
35     crop = nullptr;
36 }
```

7.5.3.51 react()

```
Button * Display::react (
    SDL_Event & e )
```

Definition at line 25 of file event.cpp.

```
26 {
27     switch(e.type)
28     {
29         case SDL_MOUSEMOTION:
30             for(auto& but : buts)
31                 but->isHover(e.motion.x, e.motion.y);
32             return nullptr;
33             break;
34         case SDL_MOUSEBUTTONDOWN:
35             for(auto& but :buts)
36                 if(but->isClicked(e.motion.x, e.motion.y)) return but;
37             return nullptr;
38             break;
39         default:
40             return nullptr;
41             break;
42     }
43 }
```

7.5.3.52 removeCharacter() [1/2]

void Object::removeCharacter () [inherited]

Definition at line 28 of file font.cpp.

```
29 {  
30     if (text.size() > 0)  
31         text.pop_back();  
32     textToTexture();  
33 }
```

7.5.3.53 removeCharacter() [2/2]

void Object::removeCharacter (
 int n) [inherited]

Definition at line 35 of file font.cpp.

```
36 {  
37     if(n == 0) return ;  
38     if(text.size() <= n) text.clear();  
39     else text.erase(text.end() - n, text.end());  
40     textToTexture();  
41 }
```

7.5.3.54 rendering()

void Display::rendering () [virtual]

Reimplemented from [Object](#).

Definition at line 4 of file rendering.cpp.

```
5 {  
6     SDL_RenderSetViewport(render, &viewport);  
7     Object::rendering();  
8     for(auto& i : buts)  
9         i->rendering();  
10 }
```

7.5.3.55 setFont()

void Object::setFont (
 TTF_Font * f) [inherited]

Definition at line 4 of file font.cpp.

```
5 {  
6     font = f;  
7     textToTexture();  
8 }
```

7.5.3.56 setShape()

```
void Object::setShape (
    const json & mem ) [inherited]
```

Definition at line 52 of file shape.cpp.

```
53 {
54     if (mem["type"].get<std::string>() == "CIRCLE")
55     {
56         if (mem.contains("center"))
57         {
58             if (mem.contains("radius"))
59                 changeToCircle(mem["center"]["x"], mem["center"]["y"], mem["radius"]);
60             else changeToCircle(mem["center"]["x"], mem["center"]["y"]);
61         } else changeToCircle();
62     }
63     return ;
64 }
65
66 if (mem["type"].get<std::string>() == "NONE" || mem["type"].get<std::string>() == "RECTANGLE")
67 {
68     changeToRectangle();
69     return ;
70 }
71
72 }
```

7.5.3.57 setText()

```
void Object::setText (
    std::string t ) [inherited]
```

Definition at line 10 of file font.cpp.

```
11 {
12     text = t;
13     textToTexture();
14 }
```

7.5.3.58 show()

```
void Object::show ( ) [inherited]
```

Definition at line 3 of file visible.cpp.

```
4 {
5     visible = true;
6 }
```

7.5.3.59 textToTexture()

void Object::textToTexture () [protected], [inherited]

Definition at line 43 of file font.cpp.

```

44 {
45     if(font == nullptr) return ;
46     if(color == nullptr) return ;
47     if(render == nullptr) return ;
48     if(texture != nullptr)
49     {
50         SDL_DestroyTexture(texture);
51     }
52     texture = nullptr;
53
54     SDL_Surface* surface = TTF_RenderText_Blended(font, text.c_str(), *color);
55
56     if(surface == nullptr) return ;
57
58     texture = SDL_CreateTextureFromSurface(render, surface);
59     SDL_FreeSurface(surface);
60     fitTheTexture();
61 }
```

7.5.3.60 textureFromFile()

void Object::textureFromFile (
 std::string dir) [inherited]

Definition at line 4 of file external_storage.cpp.

```

5 {
6     SDL_Surface *surface = IMG_Load(dir.c_str());
7
8     texture = SDL_CreateTextureFromSurface(render, surface);
9     SDL_FreeSurface(surface);
10 }
```

7.5.3.61 zoom()

void Object::zoom (
 double delta) [virtual], [inherited]

Definition at line 123 of file locating.cpp.

```

124 {
125     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
126     location->w *= delta;
127     location->h *= delta;
128 }
```

7.5.3.62 zoomH()

void Object::zoomH (
 int delta) [virtual], [inherited]

Definition at line 118 of file locating.cpp.

```

119 {
120     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
121     location->h += delta;
122 }
```


7.5.3.63 zoomInMiddle()

```
void Object::zoomInMiddle (
    double delta ) [virtual], [inherited]
```

Definition at line 130 of file locating.cpp.

```
131 {
132     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
133     SDL_Point center = {location->x + location->w / 2, location->y + location->h / 2};
134     location->w *= delta;
135     location->h *= delta;
136     location->x = center.x - location->w / 2;
137     location->y = center.y - location->h / 2;
138 }
```

7.5.3.64 zoomW()

```
void Object::zoomW (
    int delta ) [virtual], [inherited]
```

Definition at line 112 of file locating.cpp.

```
113 {
114     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
115     location->w += delta;
116 }
```

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

- include/display.hpp
- src/display/constructor.cpp
- src/display/destructor.cpp
- src/display/event.cpp
- src/display/rendering.cpp

7.6 distanceHeap Struct Reference

Public Member Functions

- distanceHeap (Graph *g)
- ~distanceHeap ()
- bool swapable (int i, int j)
- void swap (int i, int j)
- void insert (int v)
- int pop ()
- bool empty ()

Public Attributes

- Graph * g
- std::vector< int > value

7.6.1 Detailed Description

Definition at line 3 of file dijkstra.cpp.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 distanceHeap()

```
distanceHeap::distanceHeap (
    Graph * g ) [inline]
```

Definition at line 8 of file dijkstra.cpp.

```
9     {
10         this->g = g;
11     }
```

7.6.2.2 ~distanceHeap()

```
distanceHeap::~distanceHeap ( ) [inline]
```

Definition at line 12 of file dijkstra.cpp.

```
13     {
14         value.clear();
15         g = nullptr;
16     }
```

7.6.3 Member Function Documentation

7.6.3.1 empty()

```
bool distanceHeap::empty ( ) [inline]
```

Definition at line 71 of file dijkstra.cpp.

```
71 { return value.empty(); }
```

7.6.3.2 insert()

```
void distanceHeap::insert (
    int v ) [inline]
```

Definition at line 28 of file dijkstra.cpp.

```
29     {
30         value.push_back(v);
31         int index = value.size() - 1;
32         do
33         {
34             int parent = (index - 1) / 2;
35             if(swapable(parent, index))
36             {
37                 swap(parent, index);
38             }
39             index = parent;
40         }while(index != 0);
41     }
```

7.6.3.3 pop()

```
int distanceHeap::pop ( ) [inline]
```

Definition at line 42 of file dijkstra.cpp.

```
43     {
44         if(value.size() == 0) return 2e9;
45         int result = value[0];
46
47         value[0] = value[value.size() - 1];
48         value.pop_back();
49
50         int index = 0;
51
52         while(index < value.size())
53         {
54             int left = index * 2 + 1;
55             int right = index * 2 + 2;
56             int next = index;
57
58             if(left < value.size() && swapable(next, left))
59                 next = left;
60             if(right < value.size() && swapable(next, right))
61                 next = right;
62
63             if(next == index) break;
64
65             swap(index, next);
66             index = next;
67         }
68
69         return result;
70     }
```

7.6.3.4 swap()

```
void distanceHeap::swap (
    int i,
    int j ) [inline]
```

Definition at line 24 of file dijkstra.cpp.

```
25     {
26         std::swap(value[i], value[j]);
27     }
```

7.6.3.5 swapable()

```
bool distanceHeap::swapable (
    int i,
    int j ) [inline]
```

Definition at line 18 of file dijkstra.cpp.

```
19 {
20     if (g->distance[value[i]] == -1) return true;
21     if (g->distance[value[j]] == -1) return false;
22     return g->distance[value[i]] > g->distance[value[j]];
23 }
```

7.6.4 Member Data Documentation

7.6.4.1 g

[Graph*](#) distanceHeap::g

Definition at line 5 of file dijkstra.cpp.

7.6.4.2 value

[std::vector<int>](#) distanceHeap::value

Definition at line 6 of file dijkstra.cpp.

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

- [src/graph/operator/dijkstra.cpp](#)

7.7 DSU Struct Reference

Public Member Functions

- [DSU](#) (int n)
- [~DSU](#) ()
- void [unionEdge](#) (Graph::Edge *e)
- void [unionEdge](#) (Graph::Node *u, Graph::Node *v)
- void [unionEdge](#) (int u, int v)
- int [find](#) (Graph::Node *v)
- int [find](#) (int v)
- bool [isUnionized](#) (Graph::Node *u, Graph::Node *v)
- bool [isUnionized](#) (Graph::Edge *e)
- bool [isUnionized](#) (int u, int v)

Public Attributes

- `std::vector< int >` [parent](#)

7.7.1 Detailed Description

Definition at line 4 of file `mst.cpp`.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 DSU()

```
DSU::DSU (
    int n ) [inline]
```

Definition at line 7 of file `mst.cpp`.

```
8     {
9         parent.resize(n + 1);
10        for(int i = 0; i < n; i++)
11            parent[i] = -1;
12    }
```

7.7.2.2 ~DSU()

```
DSU::~DSU ( ) [inline]
```

Definition at line 13 of file `mst.cpp`.

```
14    {
15        parent.clear();
16    }
```

7.7.3 Member Function Documentation

7.7.3.1 find() [1/2]

```
int DSU::find (
    Graph::Node * v ) [inline]
```

Definition at line 36 of file `mst.cpp`.

```
37    {
38        return find(v->value);
39    }
```

7.7.3.2 find() [2/2]

```
int DSU::find (
    int v ) [inline]
```

Definition at line 40 of file mst.cpp.

```
41     {
42         if (parent[v] < 0)
43             return v;
44         return parent[v] = find(parent[v]);
45     }
```

7.7.3.3 isUnionized() [1/3]

```
bool DSU::isUnionized (
    Graph::Edge * e ) [inline]
```

Definition at line 51 of file mst.cpp.

```
52     {
53         return isUnionized(e->u, e->v);
54     }
```

7.7.3.4 isUnionized() [2/3]

```
bool DSU::isUnionized (
    Graph::Node * u,
    Graph::Node * v ) [inline]
```

Definition at line 47 of file mst.cpp.

```
48     {
49         return isUnionized(u->value, v->value);
50     }
```

7.7.3.5 isUnionized() [3/3]

```
bool DSU::isUnionized (
    int u,
    int v ) [inline]
```

Definition at line 55 of file mst.cpp.

```
56     {
57         return find(u) == find(v);
58     }
```

7.7.3.6 unionEdge() [1/3]

```
void DSU::unionEdge (
    Graph::Edge * e ) [inline]
```

Definition at line 18 of file mst.cpp.

```
19 {
20     unionEdge(e->u, e->v);
21 }
```

7.7.3.7 unionEdge() [2/3]

```
void DSU::unionEdge (
    Graph::Node * u,
    Graph::Node * v ) [inline]
```

Definition at line 22 of file mst.cpp.

```
23 {
24     unionEdge(u->value, v->value);
25 }
```

7.7.3.8 unionEdge() [3/3]

```
void DSU::unionEdge (
    int u,
    int v ) [inline]
```

Definition at line 26 of file mst.cpp.

```
27 {
28     int a = find(u);
29     int b = find(v);
30     if(a == b) return ;
31
32     parent[a] += parent[b];
33     parent[b] = a;
34 }
```

7.7.4 Member Data Documentation

7.7.4.1 parent

```
std::vector<int> DSU::parent
```

Definition at line 6 of file mst.cpp.

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

- src/graph/operator/mst.cpp

7.8 Graph Class Reference

[Graph](#) class.

```
#include <graph.hpp>
```

Public Member Functions

- [Graph](#) (SDL_Renderer *r, std::mutex &m, TTF_Font *f, SDL_Rect v, int capacity)
- [~Graph](#) ()
- void [Dijkstra](#) (int start, int end)
- void [MST](#) ()
- void [SCC](#) ()
- void [init](#) (std::vector< std::vector< int > > value)
- bool [isReceiveEvent](#) (SDL_Event &e)
- [Button](#) * [react](#) (SDL_Event &e)
- void [rendering](#) ()
- void [setting](#) (SDL_Color c1, SDL_Color c2, SDL_Color c3, SDL_Color c4)

Protected Member Functions

- void [unionEdges](#) ()
- void [Tarjan](#) (Node *u)
- void [repair](#) ()
- void [renderEdge](#) (Edge *edge)
- void [waitForStep](#) ()

Friends

- struct [distanceHeap](#)
- struct [DSU](#)

7.8.1 Detailed Description

[Graph](#) class.

Drawable graph.

Definition at line 23 of file graph.hpp.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 Graph()

```
Graph::Graph (
    SDL_Renderer * r,
    std::mutex & m,
    TTF_Font * f,
    SDL_Rect v,
    int capacity )
```

Definition at line 33 of file constructor.cpp.

```
33                                     : ds_mutex(m)
34 {
35     render = r;
36     font = f;
37     viewport = v;
38     capacity = capacity;
39     edgesColor = {255, 255, 255, 255};
40     shiftX = 20;
41     shiftY = 20;
42
43     nodirect = false;
44
45     nodeColor = {20, 75, 185, 255};
46     fontColor = {255, 255, 255, 255};
47
48     stepWait = 600;
49
50     isMoving = false;
51     chosenNode = nullptr;
52     lastMousePressed = {0, 0};
53
54
55 }
```

7.8.2.2 ~Graph()

```
Graph::~Graph ( )
```

Definition at line 19 of file destructor.cpp.

```
20 {
21     for(auto &i : nodes)
22         delete i;
23     nodes.clear();
24
25     for(auto &i : edges)
26         delete i;
27     edges.clear();
28 }
```

7.8.3 Member Function Documentation

7.8.3.1 Dijkstra()

```
void Graph::Dijkstra (
    int start,
    int end )
```

Definition at line 74 of file dijkstra.cpp.

```
75 {
76     repair();
```

```

77     distance.clear();
78     distance.resize(nodes.size() + 1, -1);
79     distanceHeap heap(this);
80     distance[start] = 0;
81
82     heap.insert(start);
83
84     for(auto i : nodes) i->sprite->coloring(SDL_Color{50, 50, 50, 255});
85
86     while(!heap.empty())
87     {
88         int u = heap.pop();
89         nodes[u]->sprite->coloring(SDL_Color{0, 125, 0, 255});
90         waitForStep();
91         for(auto e : nodes[u]->edges)
92         {
93             int v = e->v->value;
94
95
96             if(distance[v] == -1 || distance[v] > distance[u] + e->weight)
97             {
98                 nodes[v]->sprite->coloring(SDL_Color{255, 255, 0, 255});
99                 waitForStep();
100                 nodes[v]->sprite->coloring(SDL_Color{0, 255, 255, 255});
101                 waitForStep();
102                 distance[v] = distance[u] + e->weight;
103                 heap.insert(v);
104             }
105         }
106         nodes[u]->sprite->coloring(SDL_Color{0, 255, 0, 255});
107     }
108 }

```

7.8.3.2 init()

```

void Graph::init (
    std::vector< std::vector< int > > value )

```

Definition at line 56 of file init.cpp.

```

57 {
58     capacity = g.size();
59
60     for(auto i : nodes)
61         delete i;
62     for(auto i : edges)
63         delete i;
64     nodes.clear();
65     edges.clear();
66
67     for(int i = 0; i < capacity; i++)
68     {
69         int x = 10;
70         int y = 10;
71         int rep = 20;
72         do
73         {
74             x = RANDOM::getInt(10, 720);
75             y = RANDOM::getInt(10, 520);
76             rep--;
77         }while(rep != 0 && isCollision(x, y));
78         Sprite* spr = new Sprite(render);
79         spr->setFont(font);
80         spr->linking("AVL/node");
81         spr->locatingX(x);
82         spr->locatingY(y);
83         spr->setText(NUMBER::intToString(i));
84         spr->setFontColor(fontColor);
85         spr->coloring(nodeColor);
86         spr->aligning(HORIZONTAL_ALIGN::CENTER, VERTICAL_ALIGN::CENTER);
87
88         Node* node = new Node(i, spr);
89
90         nodes.push_back(node);
91     }
92
93     for(int i = 0; i < capacity; i++)
94     {
95         for(int j = 0; j < capacity && j < g[i].size(); j++)

```

```

96     {
97         if(g[i][j] != 0)
98         {
99             Sprite* spr = new Sprite(render);
100             spr->setFont(font);
101             spr->linking("graph/weight");
102
103             spr->locatingX((nodes[i]->sprite->getX() + nodes[j]->sprite->getX()) / 2);
104             spr->locatingY((nodes[i]->sprite->getY() + nodes[j]->sprite->getY()) / 2);
105
106             spr->setText(NUMBER::intToString(g[i][j]));
107             spr->aligning(HORIZONTAL_ALIGN::CENTER, VERTICAL_ALIGN::CENTER);
108             Edge* edge = new Edge(nodes[i], nodes[j], g[i][j], spr);
109             edges.push_back(edge);
110             nodes[i]->addEdge(edge);
111         }
112     }
113 }
114 }

```

7.8.3.3 isReceiveEvent()

```

bool Graph::isReceiveEvent (
    SDL_Event & e )

```

Definition at line 4 of file event.cpp.

```

5 {
6     std::lock_guard<std::mutex> lk(animate_mutex);
7     switch(e.type)
8     {
9         case SDL_MOUSEBUTTONDOWN:
10             if(e.motion.x < viewport.x || viewport.x + viewport.w < e.motion.x) return false;
11             if(e.motion.y < viewport.y || viewport.y + viewport.h < e.motion.y) return false;
12             if(e.button.button == SDL_BUTTON_LEFT) return false;
13             for(auto i : nodes)
14                 if(i->sprite->isLieInside(e.motion.x, e.motion.y))
15                     return true;
16             return false;
17             break;
18         case SDL_MOUSEMOTION:
19             if(isMoving) return true;
20             return false;
21             break;
22         default:
23             return false;
24             break;
25     }
26 }

```

7.8.3.4 MST()

```

void Graph::MST ( )

```

Definition at line 62 of file mst.cpp.

```

63 {
64     sortedEdges.clear();
65
66     for(auto i : edges)
67         sortedEdges.push_back(i);
68
69     std::sort(
70         sortedEdges.begin(),
71         sortedEdges.end(),
72         [&](Edge* u, Edge* v){
73             if(u == nullptr) return false;
74             if(v == nullptr) return true;
75             return u->weight < v->weight;
76         }
77     );
78
79     unionEdges();
80 }

```

7.8.3.5 react()

```
Button * Graph::react (
    SDL_Event & e )
```

Definition at line 29 of file event.cpp.

```
30 {
31     std::lock_guard<std::mutex> lk(animate_mutex);
32     switch(e.type)
33     {
34         case SDL_MOUSEBUTTONDOWN:
35             if(isMoving)
36             {
37                 isMoving = false;
38                 chosenNode = nullptr;
39             }else
40             {
41                 isMoving = true;
42
43                 lastMousePressed.x = e.motion.x;
44                 lastMousePressed.y = e.motion.y;
45                 for(auto i : nodes)
46                     if(i->sprite->isLieInside(e.motion.x, e.motion.y))
47                     {
48                         chosenNode = i;
49                         break;
50                     }
51             }
52             return nullptr;
53             break;
54         case SDL_MOUSEMOTION:
55             {
56                 if(!isMoving) return nullptr;
57                 int dx = e.motion.x - lastMousePressed.x;
58                 int dy = e.motion.y - lastMousePressed.y;
59                 lastMousePressed.x = e.motion.x;
60                 lastMousePressed.y = e.motion.y;
61                 if(chosenNode == nullptr) return nullptr;
62                 chosenNode->sprite->moveX(dx);
63                 chosenNode->sprite->moveY(dy);
64                 return nullptr;
65                 break;
66             }
67     default:
68         return nullptr;
69         break;
70     }
71     return nullptr;
72 }
```

7.8.3.6 renderEdge()

```
void Graph::renderEdge (
    Edge * edge ) [protected]
```

Definition at line 8 of file rendering.cpp.

```
9 {
10     if(e->mark == 3)
11         SDL_SetRenderDrawColor(render, 50, 50, 50, 255);
12     else if(e->mark == 2)
13         SDL_SetRenderDrawColor(render, 255, 0, 0, 255);
14     else if(e->mark == 1)
15         SDL_SetRenderDrawColor(render, 0, 255, 0, 255);
16     else
17         SDL_SetRenderDrawColor(render, edgesColor.r, edgesColor.g, edgesColor.b, edgesColor.a);
18     const SDL_Rect* srcloc = (e->u->sprite->getLocation());
19     const SDL_Rect* dstloc = (e->v->sprite->getLocation());
20
21     SDL_Point src;
22     src.x = srcloc->x + srcloc->w/2;
23     src.y = srcloc->y + srcloc->h/2;
24
25
26     SDL_Point dst;
27     dst.x = dstloc->x + dstloc->w/2;
```

```

28     dst.y = dstloc->y + dstloc->h/2;
29
30     int sign = 1;
31     if(dst.x < src.x)
32         sign = -1;
33     e->sprite->locatingX((src.x + dst.x) / 2 + sign * 10);
34
35     sign = 1;
36     if(dst.y < src.y)
37         sign = -1;
38     e->sprite->locatingY((src.y + dst.y) / 2 + sign * 10);
39
40     for (int i = -1; i <= 1; i++)
41     {
42         for(int j = -1; j <= 1; j++)
43         {
44             SDL_RenderDrawLine(render, src.x + i, src.y + j, dst.x + i, dst.y + j);
45         }
46     }
47     if(nodirect) return ;
48     Point v;
49     v.x = dst.x - src.x;
50     v.y = dst.y - src.y;
51     double len = sqrt(v.x * v.x + v.y * v.y);
52     v.x /= len;
53     v.y /= len;
54
55
56     Point u = {v.y, -v.x};
57
58     SDL_Point p1;
59     p1.x = dst.x - v.x * 40 + u.x * 10;
60     p1.y = dst.y - v.y * 40 + u.y * 10;
61
62     SDL_Point p2;
63     p2.x = dst.x - v.x * 40 - u.x * 10;
64     p2.y = dst.y - v.y * 40 - u.y * 10;
65
66     SDL_Point p3;
67     p3.x = dst.x - v.x * 28;
68     p3.y = dst.y - v.y * 28;
69
70     for(int i = -1; i <= 1; i++)
71         for(int j = -1; j <= 1; j++)
72         {
73             SDL_RenderDrawLine(render, p1.x + i, p1.y + j, p3.x + i, p3.y + j);
74             SDL_RenderDrawLine(render, p2.x + i, p2.y + j, p3.x + i, p3.y + j);
75         }
76 }

```

7.8.3.7 rendering()

```
void Graph::rendering ( )
```

Definition at line 78 of file rendering.cpp.

```

79 {
80     for(auto i : edges)
81     {
82         renderEdge(i);
83         if(i->sprite != nullptr) i->sprite->rendering();
84     }
85     for(auto i : nodes)
86     {
87         i->sprite->rendering();
88     }
89 }

```

7.8.3.8 repair()

```
void Graph::repair ( ) [protected]
```

Definition at line 3 of file repair.cpp.

```

4 {
5     for(auto i : edges)
6     {
7         i->mark = 0;
8     }
9     setting(bgColor, nodeColor, fontColor, edgesColor);
10    nodirect = false;
11 }
```

7.8.3.9 SCC()

```
void Graph::SCC ( )
```

Definition at line 37 of file scc.cpp.

```

38 {
39     low.clear();
40     order.clear();
41     components.clear();
42     repair();
43     state = 0;
44
45     low.resize(nodes.size() + 1);
46     order.resize(nodes.size() + 1);
47
48     for(auto i : nodes)
49     {
50         if(order[i->value] == 0)
51             Tarjan(i);
52     }
53
54     for(auto i : components)
55     {
56         SDL_Color c;
57         c.r = RANDOM::getInt(0, 255);
58         c.g = RANDOM::getInt(0, 255);
59         c.b = RANDOM::getInt(0, 255);
60         c.a = 255;
61         for(auto j : i)
62         {
63             j->sprite->coloring(c);
64         }
65     }
66 }
```

7.8.3.10 setting()

```
void Graph::setting (
    SDL_Color c1,
    SDL_Color c2,
    SDL_Color c3,
    SDL_Color c4 )
```

Definition at line 57 of file constructor.cpp.

```

58 {
59     bgColor = c1;
60     nodeColor = c2;
61     fontColor = c3;
62     edgesColor = c4;
63     for(auto i : nodes)
64     {
65         i->sprite->setFontColor(fontColor);
66         i->sprite->coloring(nodeColor);
67     }
68 }
```

7.8.3.11 Tarjan()

```
void Graph::Tarjan (
    Node * u )    [protected]
```

Definition at line 4 of file scc.cpp.

```
5 {
6     order[u->value] = low[u->value] = ++state;
7     buffer.push(u);
8
9     for(auto i : u->edges)
10    {
11        if(order[i->v->value] == 0)
12        {
13            Tarjan(i->v);
14            low[u->value] = std::min(low[u->value], low[i->v->value]);
15        }
16        else if(order[i->v->value] != 0 && low[i->v->value] != 0)
17        {
18            low[u->value] = std::min(low[u->value], order[i->v->value]);
19        }
20    }
21
22    if(low[u->value] == order[u->value])
23    {
24        std::vector<Node*> component;
25        Node* v;
26        do
27        {
28            v = buffer.top();
29            buffer.pop();
30            component.push_back(v);
31            low[v->value] = 0;
32        } while(v != u);
33        components.push_back(component);
34    }
35 }
```

7.8.3.12 unionEdges()

```
void Graph::unionEdges ( )    [protected]
```

Definition at line 82 of file mst.cpp.

```
83 {
84     repair();
85     DSU dsu(nodes.size());
86     nodirect = true;
87     for(auto i : sortedEdges)
88     {
89         i->mark = 3;
90         waitForStep();
91         if(!dsu.isUnionized(i))
92         {
93             i->mark = 1;
94             waitForStep();
95             dsu.unionEdge(i);
96         }else
97         {
98             i->mark = 2;
99             waitForStep();
100        }
101    }
102 }
```

7.8.3.13 waitForStep()

```
void Graph::waitForStep ( )    [protected]
```

Definition at line 75 of file event.cpp.

```
76 {
77     ds_mutex.unlock();
78     std::this_thread::sleep_for(std::chrono::milliseconds(stepWait));
79     ds_mutex.lock();
80 }
```

7.8.4 Friends And Related Function Documentation

7.8.4.1 distanceHeap

```
friend struct distanceHeap [friend]
```

Definition at line 78 of file graph.hpp.

7.8.4.2 DSU

```
friend struct DSU [friend]
```

Definition at line 86 of file graph.hpp.

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

- include/data_structures/graph.hpp
- src/graph/constructor.cpp
- src/graph/destructor.cpp
- src/graph/event.cpp
- src/graph/operator/dijkstra.cpp
- src/graph/operator/init.cpp
- src/graph/operator/mst.cpp
- src/graph/operator/repair.cpp
- src/graph/operator/scc.cpp
- src/graph/rendering.cpp

7.9 HashTable Class Reference

[HashTable](#) class.

```
#include <hash_table.hpp>
```

Public Member Functions

- [HashTable](#) (SDL_Renderer *render, std::mutex &m, TTF_Font *font, SDL_Rect v, int cap)
- [~HashTable](#) ()
- void [init](#) (std::vector< int > v, int KEY)
- void [insert](#) (int key)
- void [remove](#) (int key)
- bool [search](#) (int key)
- void [setEdgesColor](#) (SDL_Color c)
- void [setNodeColor](#) (SDL_Color bg, SDL_Color fg)
- void [goOff](#) ()
- void [goOn](#) ()
- void [goNext](#) ()
- void [goBack](#) ()
- void [speedUp](#) ()
- void [slowDown](#) ()
- bool [isReceiveEvent](#) (SDL_Event &e)
- [Button](#) * [react](#) (SDL_Event &e)
- void [closeScript](#) ()
- void [rendering](#) ()
- void [setting](#) (SDL_Color c1, SDL_Color c2, SDL_Color c3, SDL_Color c4)

Protected Member Functions

- Node * [insert](#) (Node *root, int k)
- Node * [remove](#) (Node *root, int k)
- bool [search](#) (Node *root, int k)
- int [locating](#) (Node *node, int shiftDown, int shiftRight)
- int [locating](#) (Head **table, int shiftDown, int shiftRight)
- void [renderEdges](#) (Node *src, Node *dst)
- void [defaultSetting](#) ()
- void [drawEdge](#) (Node *src, Node *dst)
- void [waitForStep](#) ()
- void [highlight](#) (std::vector< int > l)
- void [unhighlight](#) (std::vector< int > l)

7.9.1 Detailed Description

[HashTable](#) class.

Drawable [HashTable](#).

Definition at line 23 of file hash_table.hpp.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 HashTable()

```
HashTable::HashTable (
    SDL_Renderer * render,
    std::mutex & m,
    TTF_Font * font,
    SDL_Rect v,
    int cap )
```

Definition at line 18 of file constructor.cpp.

```
18
19 {
20     render = r;
21     font = f;
22     viewport = v;
23     capacity = cap;
24
25     table = nullptr;
26
27     currentSize = 0;
28
29     isQueue = false;
30     isPause = false;
31
32     edgesColor = {255, 255, 255, 255};
33     nodeColor = {20, 85, 185, 255};
34     fontColor = {255, 255, 255, 255};
35
36     shiftX = 20;
37     shiftY = 20;
38     distanceX = 100;
39     distanceY = 70;
40     isMoving = false;
41
: ds_mutex(m)
```

```

42     stepWait = 600;
43
44     std::string fontpath = PATH::ASSETS::FONTS_ + "nimbus-sans-l/regular.otf";
45     scriptFont = TTF_OpenFont(fontpath.c_str(), 18);
46
47     currentScript = nullptr;
48     Script* insert = new Script(render, scriptFont);
49     insert->linking("hash_table/insert");
50     scripts[DATA_STRUCTURES_OPERATOR::INSERT] = insert;
51
52     Script* remove = new Script(render, scriptFont);
53     remove->linking("hash_table/remove");
54     scripts[DATA_STRUCTURES_OPERATOR::DELETE] = remove;
55
56     Script* search = new Script(render, scriptFont);
57     search->linking("hash_table/search");
58     scripts[DATA_STRUCTURES_OPERATOR::SEARCH] = search;
59
60     Script* init = new Script(render, scriptFont);
61     init->linking("hash_table/init");
62     scripts[DATA_STRUCTURES_OPERATOR::INIT] = init;
63
64 }

```

7.9.2.2 ~HashTable()

HashTable::~~HashTable ()

Definition at line 15 of file destructor.cpp.

```

16 {
17     if(table != nullptr)
18     {
19         for(int i = 0; i < HASH_KEY; i++)
20         {
21             if(table[i] != nullptr) delete table[i];
22         }
23         delete [] table;
24     }
25     for(auto i : scripts)
26         delete i.second;
27     TTF_CloseFont(scriptFont);
28 }

```

7.9.3 Member Function Documentation

7.9.3.1 closeScript()

void HashTable::closeScript ()

Definition at line 53 of file event.cpp.

```

54 {
55     currentScript = nullptr;
56 }

```

7.9.3.2 defaultSetting()

```
void HashTable::defaultSetting ( ) [protected]
```

Definition at line 100 of file constructor.cpp.

```
101 {
102     HASH_KEY = 19;
103     table = new Head*[HASH_KEY];
104     for(int i = 0; i < HASH_KEY; i++)
105     {
106         Sprite* spr = new Sprite(render);
107         spr->setFont(font);
108         spr->setFontColor(fontColor);
109         spr->coloring(nodeColor);
110         spr->linking("hash-table/head");
111         table[i] = new Head(spr);
112     }
113     locating(table, 0, 0);
114 }
```

7.9.3.3 drawEdge()

```
void HashTable::drawEdge (
    Node * src,
    Node * dst ) [protected]
```

Definition at line 3 of file rendering.cpp.

```
4 {
5     SDL_Rect srcRect = *src->sprite->getLocation();
6     SDL_Rect dstRect = *dst->sprite->getLocation();
7
8     SDL_SetRenderDrawColor(render, edgesColor.r, edgesColor.g, edgesColor.b, edgesColor.a);
9     for(int j = -1; j <= 1; j++)
10     {
11         SDL_RenderDrawLine(render, srcRect.x + srcRect.w, srcRect.y + srcRect.h / 2 + j, dstRect.x,
12             dstRect.y + dstRect.h / 2 + j);
13     }
14     for(int j = -3; j <= 3; j++)
15     {
16         SDL_RenderDrawLine(render, dstRect.x - 8, dstRect.y - 5 + j + dstRect.h / 2, dstRect.x, dstRect.y
17             + dstRect.h / 2);
18         SDL_RenderDrawLine(render, dstRect.x - 8, dstRect.y + 5 + j + dstRect.h / 2, dstRect.x, dstRect.y
19             + dstRect.h / 2);
20         SDL_RenderDrawLine(render, dstRect.x - 8, dstRect.y - 5 + j + dstRect.h / 2, dstRect.x - 8,
21             dstRect.y + 5 + j + dstRect.h / 2);
22     }
23 }
```

7.9.3.4 goBack()

```
void HashTable::goBack ( )
```

7.9.3.5 goNext()

```
void HashTable::goNext ( )
```

7.9.3.6 goOff()

```
void HashTable::goOff ( )
```

7.9.3.7 goOn()

```
void HashTable::goOn ( )
```

7.9.3.8 highlight()

```
void HashTable::highlight (
    std::vector< int > l ) [protected]
```

Definition at line 23 of file step.cpp.

```
24 {
25     if (isAnimate)
26     {
27         animate_mutex.lock();
28         for (int i = 0; i < l.size(); i++)
29         {
30             currentScript->highlight (l[i]);
31         }
32         animate_mutex.unlock();
33     }
34 }
```

7.9.3.9 init()

```
void HashTable::init (
    std::vector< int > v,
    int KEY )
```

Definition at line 6 of file init.cpp.

```
7 {
8     if (table != nullptr)
9     {
10         for (int i = 0; i < HASH_KEY; i++)
11         {
12             delete table[i];
13             table[i] = nullptr;
14         }
15         delete[] table;
16     }
17     HASH_KEY = key;
18     table = new Head*[HASH_KEY];
19     currentScript = scripts[DATA_STRUCTURES_OPERATOR::INIT];
20
21     isAnimate = false;
22
23     for (int i = 0; i < HASH_KEY; i++)
24     {
25         Sprite* spr = new Sprite(render);
26         spr->setFont(font);
27         spr->linking("hash-table/head");
28         spr->setFontColor(fontColor);
29         spr->coloring(SDL_Color{bgColor.r, bgColor.r, bgColor.r, 255});
30         table[i] = new Head(spr);
31     }
32     for (int i : v)
33     {
34         int k = i % HASH_KEY;
35         table[k]->root = insert(table[k]->root, i);
36     }
37     locating(table, 0, 0);
38 }
```

7.9.3.10 insert() [1/2]

```
void HashTable::insert (
    int key )
```

Definition at line 73 of file insert.cpp.

```
74 {
75     if(table == nullptr) defaultSetting();
76     currentScript = scripts[DATA_STRUCTURES_OPERATOR::INSERT];
77
78     isAnimate = true;
79
80     highlight({0});
81     waitForStep();
82     unhighlight({0});
83
84     highlight({1, 2});
85     waitForStep();
86     unhighlight({1, 2});
87
88     int k = key % HASH_KEY;
89
90     if(isAnimate && table[k]->root == nullptr)
91     {
92         animate_mutex.lock();
93         table[k]->sprite->highlight();
94         animate_mutex.unlock();
95
96         waitForStep();
97
98         animate_mutex.lock();
99         table[k]->sprite->unhighlight();
100        animate_mutex.unlock();
101    }
102
103    table[k]->root = insert(table[k]->root, key);
104    locating(table, 0, 0);
105 }
```

7.9.3.11 insert() [2/2]

```
HashTable::Node * HashTable::insert (
    Node * root,
    int k ) [protected]
```

Definition at line 4 of file insert.cpp.

```
5 {
6     if(node == nullptr)
7     {
8         Sprite* spr = new Sprite(render);
9         spr->setFont(font);
10        spr->linking("hash-table/node");
11        spr->setText(NUMBER::intToString(k));
12        spr->setFontColor(fontColor);
13        spr->coloring(nodeColor);
14        highlight({1, 2});
15        waitForStep();
16        unhighlight({1, 2});
17        return new Node(k, spr);
18    }
19    if(isAnimate)
20    {
21        animate_mutex.lock();
22        node->sprite->highlight();
23        animate_mutex.unlock();
24        waitForStep();
25    }
26    if(node->pnext == nullptr)
27    {
28        if(isAnimate)
29        {
30            animate_mutex.lock();
31            node->sprite->unhighlight();
32            animate_mutex.unlock();
```

```

33     }
34     Sprite* spr = new Sprite(render);
35     spr->setFont(font);
36     spr->linking("hash-table/node");
37     spr->setText(NUMBER::intToString(k));
38     node->pnext = new Node(k, spr);
39
40     if(isAnimate)
41     {
42         animate_mutex.lock();
43         node->pnext->sprite->highlight();
44         animate_mutex.unlock();
45     }
46
47     highlight({10});
48     waitForStep();
49     unhighlight({10});
50
51     if(isAnimate)
52     {
53         animate_mutex.lock();
54         node->pnext->sprite->unhighlight();
55         animate_mutex.unlock();
56     }
57 }else
58 {
59     highlight({7, 8, 9});
60     waitForStep();
61     unhighlight({7, 8, 9});
62     if(isAnimate)
63     {
64         animate_mutex.lock();
65         node->sprite->unhighlight();
66         animate_mutex.unlock();
67     }
68     node->pnext = insert(node->pnext, k);
69 }
70 return node;
71 }

```

7.9.3.12 isReceiveEvent()

```

bool HashTable::isReceiveEvent (
    SDL_Event & e )

```

Definition at line 3 of file event.cpp.

```

4 {
5     switch(e.type)
6     {
7         case SDL_MOUSEBUTTONDOWN:
8             if(currentScript != nullptr && currentScript->isReceiveEvent(e)) return true;
9             if(e.motion.x < viewport.x || viewport.x + viewport.w < e.motion.x) return false;
10            if(e.motion.y < viewport.y || viewport.y + viewport.h < e.motion.y) return false;
11            if(e.button.button == SDL_BUTTON_LEFT) return false;
12            if(table == nullptr) return false;
13            return true;
14            break;
15        case SDL_MOUSEMOTION:
16            if(isMoving) return true;
17            if(currentScript == nullptr) return false;
18            if(currentScript->isReceiveEvent(e)) return true;
19            return false;
20            break;
21        default:
22            return false;
23            break;
24    }
25 }

```

7.9.3.13 locating() [1/2]

```
int HashTable::locating (
    Head ** table,
    int shiftDown,
    int shiftRight ) [protected]
```

Definition at line 78 of file constructor.cpp.

```
79 {
80     if(table == nullptr) return 0;
81
82     for(int i = 0; i < HASH_KEY; i++)
83     {
84         if(table[i] == nullptr) continue;
85
86         table[i]->sprite->locatingX(shiftX + shiftRight * distanceX);
87         table[i]->sprite->locatingY(shiftY + (i + shiftDown) * distanceY);
88         table[i]->sprite->aligning(HORIZONTAL_ALIGN::CENTER, VERTICAL_ALIGN::CENTER);
89         if(table[i]->root == nullptr) table[i]->sprite->show();
90         else
91         {
92             table[i]->sprite->hide();
93             locating(table[i]->root, i + shiftDown, shiftRight);
94         }
95     }
96
97     return 0;
98 }
```

7.9.3.14 locating() [2/2]

```
int HashTable::locating (
    Node * node,
    int shiftDown,
    int shiftRigh ) [protected]
```

Definition at line 66 of file constructor.cpp.

```
67 {
68     if(node == nullptr) return 0;
69
70     node->sprite->locatingX(shiftX + shiftRight * distanceX);
71     node->sprite->locatingY(shiftY + shiftDown * distanceY);
72     node->sprite->aligning(HORIZONTAL_ALIGN::CENTER, VERTICAL_ALIGN::CENTER);
73
74     locating(node->pnext, shiftDown, shiftRight + 1);
75     return 0;
76 }
```

7.9.3.15 react()

```
Button * HashTable::react (
    SDL_Event & e )
```

Definition at line 27 of file event.cpp.

```
28 {
29     switch(e.type)
30     {
31         case SDL_MOUSEBUTTONDOWN:
32             if(currentScript != nullptr && currentScript->isReceiveEvent(e))
33             {
34                 return currentScript->react(e);
35             }
36             if(isMoving)
37             {
```

```

38         isMoving = false;
39         int dx = e.motion.x - lastMousePressed.x;
40         int dy = e.motion.y - lastMousePressed.y;
41         shiftX += dx;
42         shiftY += dy;
43     }else
44     {
45         isMoving = true;
46         lastMousePressed.x = e.motion.x;
47         lastMousePressed.y = e.motion.y;
48     }
49     return nullptr;
50     break;
51 case SDL_MOUSEMOTION:
52     {
53         if(currentScript != nullptr && currentScript->isReceiveEvent(e))
54             return currentScript->react(e);
55         if(!isMoving) return nullptr;
56         int dx = e.motion.x - lastMousePressed.x;
57         int dy = e.motion.y - lastMousePressed.y;
58         lastMousePressed.x = e.motion.x;
59         lastMousePressed.y = e.motion.y;
60         shiftX += dx;
61         shiftY += dy;
62         locating(table, 0, 0);
63         return nullptr;
64         break;
65     }
66     default:
67         return nullptr;
68         break;
69 }
70 return nullptr;
71 }

```

7.9.3.16 remove() [1/2]

```

void HashTable::remove (
    int key )

```

Definition at line 41 of file remove.cpp.

```

42 {
43     if(table == nullptr) return;
44     currentScript = scripts[DATA_STRUCTURES_OPERATOR::DELETE];
45     isAnimate = true;
46
47     highlight({0});
48     waitForStep();
49     unhighlight({0});
50
51     highlight({1, 2});
52     waitForStep();
53     unhighlight({1, 2});
54
55     if(table[key % HASH_KEY]->root == nullptr)
56     {
57         highlight({3, 4, 5});
58         waitForStep();
59         unhighlight({3, 4, 5});
60         return;
61     }
62
63     if(table[key % HASH_KEY]->root->key == key)
64     {
65         if(isAnimate)
66         {
67             animate_mutex.lock();
68             table[key % HASH_KEY]->root->sprite->highlight();
69             animate_mutex.unlock();
70         }
71         highlight({6, 7, 8});
72         waitForStep();
73         unhighlight({6, 7, 8});
74         if(isAnimate)
75         {
76             animate_mutex.lock();
77             table[key % HASH_KEY]->root->sprite->unhighlight();
78             animate_mutex.unlock();

```



```

79         }
80
81         isAnimate = false;
82         table[key % HASH_KEY]->root = remove(table[key % HASH_KEY]->root, key);
83         return ;
84     }
85     highlight({10});
86     waitForStep();
87     table[key % HASH_KEY]->root = remove(table[key % HASH_KEY]->root, key);
88     unhighlight({10});
89     locating(table[key % HASH_KEY]->root, key % HASH_KEY, 0);
90 }

```

7.9.3.17 remove() [2/2]

```

HashTable::Node * HashTable::remove (
    Node * root,
    int k ) [protected]

```

Definition at line 4 of file remove.cpp.

```

5 {
6     if(node == nullptr)
7     {
8         return nullptr;
9     }
10    if(isAnimate)
11    {
12        animate_mutex.lock();
13        node->sprite->highlight();
14        animate_mutex.unlock();
15        waitForStep();
16    }
17    if(node->key == key)
18    {
19        Node* tmp = node->pnext;
20        node->pnext = nullptr;
21        delete node;
22        return tmp;
23    }
24    if(isAnimate)
25    {
26        animate_mutex.lock();
27        node->sprite->unhighlight();
28        animate_mutex.unlock();
29    }
30    if(node->pnext->key == key)
31    {
32        highlight({11, 12, 13, 14});
33        waitForStep();
34        unhighlight({11, 12, 13, 14});
35    }
36    node->pnext = remove(node->pnext, key);
37    return node;
38 }

```

7.9.3.18 renderEdges()

```

void HashTable::renderEdges (
    Node * src,
    Node * dst ) [protected]

```

7.9.3.19 rendering()

```
void HashTable::rendering ( )
```

Definition at line 22 of file rendering.cpp.

```

23 {
24     if(table == nullptr) return ;
25     locating(table, 0, 0);
26     SDL_RenderSetViewport(render, &viewport);
27
28     for(int i = 0; i < HASH_KEY; i++)
29     {
30         if(table[i]->root == nullptr)
31             table[i]->sprite->rendering();
32         else
33         {
34             Node* current = table[i]->root;
35             while(current != nullptr)
36             {
37                 if(current->pnext != nullptr) drawEdge(current, current->pnext);
38                 current->sprite->rendering();
39                 current = current->pnext;
40             }
41         }
42     }
43     if(currentScript != nullptr)
44     {
45         SDL_RenderSetViewport(render, nullptr);
46         currentScript->rendering();
47     }
48 }
```

7.9.3.20 search() [1/2]

```
bool HashTable::search (
    int key )
```

Definition at line 33 of file search.cpp.

```

34 {
35     if(table == nullptr) return false;
36     isAnimate = true;
37     currentScript = scripts[DATA_STRUCTURES_OPERATOR::SEARCH];
38
39     highlight({0});
40     waitForStep();
41     unhighlight({0});
42
43     highlight({1, 2});
44     waitForStep();
45     unhighlight({1, 2});
46
47     if(isAnimate)
48     {
49         animate_mutex.lock();
50         table[key % HASH_KEY]->sprite->highlight();
51         animate_mutex.unlock();
52     }
53     highlight({3});
54     waitForStep();
55     if(isAnimate)
56     {
57         animate_mutex.lock();
58         table[key % HASH_KEY]->sprite->unhighlight();
59         animate_mutex.unlock();
60     }
61     search(table[key % HASH_KEY]->root, key);
62
63     unhighlight({3});
64
65     return true;
66 }
```

7.9.3.21 search() [2/2]

```
bool HashTable::search (
    Node * root,
    int k ) [protected]
```

Definition at line 4 of file search.cpp.

```
5 {
6     if(node == nullptr)
7     {
8         highlight({9});
9         waitForStep();
10        unhighlight({9});
11        return false;
12    }
13    if(isAnimate)
14    {
15        animate_mutex.lock();
16        node->sprite->highlight();
17        animate_mutex.unlock();
18        waitForStep();
19        animate_mutex.lock();
20        node->sprite->unhighlight();
21        animate_mutex.unlock();
22    }
23    if(node->key == key)
24    {
25        highlight({4, 5});
26        waitForStep();
27        unhighlight({4, 5});
28        return true;
29    }
30    return search(node->pnext, key);
31 }
```

7.9.3.22 setEdgesColor()

```
void HashTable::setEdgesColor (
    SDL_Color c )
```

7.9.3.23 setNodeColor()

```
void HashTable::setNodeColor (
    SDL_Color bg,
    SDL_Color fg )
```

7.9.3.24 setting()

```
void HashTable::setting (
    SDL_Color c1,
    SDL_Color c2,
    SDL_Color c3,
    SDL_Color c4 )
```

Definition at line 116 of file constructor.cpp.

```
117 {
```

```

118     nodeColor = c2;
119     fontColor = c3;
120     edgesColor = c4;
121     bgColor.r = nodeColor.r * 0.5;
122     bgColor.g = nodeColor.g * 0.5;
123     bgColor.b = nodeColor.b * 0.5;
124     bgColor.a = 255;
125
126     for(int i = 0; i < HASH_KEY; i++)
127     {
128         table[i]->sprite->coloring(bgColor);
129         table[i]->sprite->setFontColor(fontColor);
130
131         Node* current = table[i]->root;
132
133         while(current != nullptr)
134         {
135             current->sprite->coloring(nodeColor);
136             current->sprite->setFontColor(fontColor);
137             current = current->pnext;
138         }
139     }
140 }

```

7.9.3.25 slowDown()

```
void HashTable::slowDown ( )
```

7.9.3.26 speedUp()

```
void HashTable::speedUp ( )
```

7.9.3.27 unhighlight()

```
void HashTable::unhighlight (
    std::vector< int > l ) [protected]
```

Definition at line 36 of file step.cpp.

```

37 {
38     if(isAnimate)
39     {
40         animate_mutex.lock();
41         for(int i = 0; i < l.size(); i++)
42         {
43             currentScript->unhighlight(l[i]);
44         }
45         animate_mutex.unlock();
46     }
47 }

```

7.9.3.28 waitForStep()

void HashTable::waitForStep () [protected]

Definition at line 2 of file step.cpp.

```

3 {
4     if(isAnimate)
5     {
6         ds_mutex.unlock();
7         std::this_thread::sleep_for(std::chrono::milliseconds(stepWait));
8         ds_mutex.lock();
9     }
10    std::lock_guard<std::mutex> pause_lock(pause_mutex);
11    if(isPause == false)
12    {
13        return ;
14    }
15
16    ds_mutex.unlock();
17    std::unique_lock<std::mutex> lk(step_mutex);
18    step_cv.wait(lk, [&]{return isQueue == true;});
19    isQueue = false;
20    ds_mutex.lock();
21 }
```

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

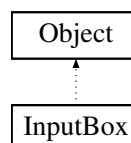
- [include/data_structures/hash_table.hpp](#)
- [src/data_structures/event.cpp](#)
- [src/hash_table/constructor.cpp](#)
- [src/hash_table/destructor.cpp](#)
- [src/hash_table/operator/init.cpp](#)
- [src/hash_table/operator/insert.cpp](#)
- [src/hash_table/operator/remove.cpp](#)
- [src/hash_table/operator/search.cpp](#)
- [src/hash_table/rendering.cpp](#)
- [src/hash_table/step.cpp](#)

7.10 InputBox Class Reference

Register for user keyboard input.

```
#include <inputbox.hpp>
```

Inheritance diagram for InputBox:



Public Member Functions

- [InputBox](#) (SDL_Renderer *render, TTF_Font *font)
- [~InputBox](#) ()
- void [setDuplicate](#) (int n, int m)
- void [linking](#) (std::string n)
- [DATA_STRUCTURES_OPERATOR](#) [getOperator](#) ()
- bool [isReceiveEvent](#) (SDL_Event &e)
- [Button](#) * [react](#) (SDL_Event &e)
- std::string [getText](#) (int index)
- void [setText](#) (int index, std::string text)
- void [setText](#) (std::string text)
- void [rendering](#) ()

Protected Member Functions

- void `initBackground` (const `json` &mem)
- void `initButtons` (const `json` &mem)
- void `initSprites` (const `json` &mem)
- void `initOperator` (const `json` &mem)
- void `importFromJson` ()
- bool `isButtonReceiveEvent` (SDL_Event &event)
- bool `isInputReceiveEvent` (SDL_Event &event)

Protected Attributes

- int `n`
- int `m`

Private Member Functions

- void `importFromJson` (const `json` &mem)
- std::string `getText` ()

7.10.1 Detailed Description

Register for user keyboard input.

Drawable

Definition at line 18 of file `inputbox.hpp`.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 InputBox()

```
InputBox::InputBox (
    SDL_Renderer * render,
    TTF_Font * font )
```

Definition at line 9 of file `constructor.cpp`.

```
9                                     : Object (render)
10 {
11     this->font = font;
12     this->render = render;
13     n = 1;
14     m = 1;
15 }
```

7.10.2.2 ~InputBox()

```
InputBox::~~InputBox ( )
```

Definition at line 3 of file destructor.cpp.

```
4 {
5     for(auto& i : buts) delete i;
6     for(auto& i : inputs) delete i;
7 }
```

7.10.3 Member Function Documentation

7.10.3.1 getOperator()

```
DATA_STRUCTURES_OPERATOR InputBox::getOperator ( )
```

Definition at line 3 of file operator.cpp.

```
4 {
5     return op;
6 }
```

7.10.3.2 getText()

```
std::string InputBox::getText (
    int index )
```

Definition at line 3 of file typing.cpp.

```
4 {
5     if(index == -1 || index >= (int) inputs.size())
6         return "";
7     return inputs[index]->getText();
8 }
```

7.10.3.3 importFromJson()

```
void InputBox::importFromJson ( ) [protected]
```

Definition at line 39 of file constructor.cpp.

```
40 {
41     json* mem = JSON::readFile(PATH::ATB::INPUTBOX_ + name + ".json");
42
43     if(mem->contains("background"))
44         initBackground((*mem) ["background"]);
45
46     if(mem->contains("buttons"))
47         initButtons((*mem) ["buttons"]);
48
49     if(mem->contains("sprites"))
50         initSprites((*mem) ["sprites"]);
51     if(mem->contains("operator"))
52         initOperator((*mem) ["operator"]);
53     delete mem;
54 }
```

7.10.3.4 initBackground()

```
void InputBox::initBackground (
    const json & mem ) [protected]
```

Definition at line 56 of file constructor.cpp.

```
57 {
58     Object::importFromJson (mem);
59 }
```

7.10.3.5 initButtons()

```
void InputBox::initButtons (
    const json & mem ) [protected]
```

Definition at line 61 of file constructor.cpp.

```
62 {
63     for(auto& i : mem)
64     {
65         buts.push_back(new Button(render));
66         if(i.contains("name")) buts.back()->linking(i["name"]);
67         if(i.contains("dx")) buts.back()->move(i["dx"], i["dy"]);
68     }
69 }
```

7.10.3.6 initOperator()

```
void InputBox::initOperator (
    const json & mem ) [protected]
```

Definition at line 23 of file constructor.cpp.

```
24 {
25     std::string type = mem.get<std::string>();
26
27     if(type == "INIT") op = DATA_STRUCTURES_OPERATOR::INIT;
28     else if (type == "INSERT") op = DATA_STRUCTURES_OPERATOR::INSERT;
29     else if (type == "DELETE") op = DATA_STRUCTURES_OPERATOR::DELETE;
30     else if (type == "SEARCH") op = DATA_STRUCTURES_OPERATOR::SEARCH;
31     else if (type == "TOP") op = DATA_STRUCTURES_OPERATOR::TOP;
32     else if (type == "SIZE") op = DATA_STRUCTURES_OPERATOR::SIZE;
33     else if (type == "SCC") op = DATA_STRUCTURES_OPERATOR::SCC;
34     else if (type == "DIJKSTRA") op = DATA_STRUCTURES_OPERATOR::DIJKSTRA;
35     else if (type == "MST") op = DATA_STRUCTURES_OPERATOR::MST;
36     else if (type == "SETTING") op = DATA_STRUCTURES_OPERATOR::SETTING;
37 }
```


7.10.3.7 initSprites()

```
void InputBox::initSprites (
    const json & mem ) [protected]
```

Definition at line 71 of file constructor.cpp.

```
72 {
73     for(auto& i : mem)
74     {
75         if(n * m >= 1 && n > 0 && m > 0 && i["name"].get<std::string>() == "graph/init/edge")
76         {
77             int dx = 35;
78             int dy = 35;
79
80             for(int u = 0; u < n; u++)
81             {
82                 for(int v= 0; v < m; v++)
83                 {
84                     inputs.push_back(new Sprite(render));
85                     inputs.back()->setFont(font);
86                     if(i.contains("name")) inputs.back()->linking(i["name"].get<std::string>());
87                     inputs.back()->moveX(dx * u);
88                     inputs.back()->moveY(dy * v);
89                 }
90             }
91         }else
92         {
93             inputs.push_back(new Sprite(render));
94             inputs.back()->setFont(font);
95             if(i.contains("name")) inputs.back()->linking(i["name"].get<std::string>());
96         }
97     }
98 }
```

7.10.3.8 isButtonReceiveEvent()

```
bool InputBox::isButtonReceiveEvent (
    SDL_Event & event ) [protected]
```

Definition at line 46 of file event.cpp.

```
47 {
48     for(auto& but : buts)
49     {
50         if(but->isReceiveEvent(e))
51         {
52             return true;
53         }
54     }
55     return false;
56 }
```

7.10.3.9 isInputReceiveEvent()

```
bool InputBox::isInputReceiveEvent (
    SDL_Event & event ) [protected]
```

Definition at line 19 of file event.cpp.

```
20 {
21     switch(e.type)
22     {
23         case SDL_MOUSEBUTTONDOWN:
24             for(auto& inp : inputs)
25                 if(inp->isLieInside(e.motion.x, e.motion.y))
26                     return true;
27             return false;
28             break;
```

```

29         case SDL_MOUSEMOTION:
30             for(auto& inp : inputs)
31                 if(inp->isLieInside(e.motion.x, e.motion.y))
32                     return true;
33             return false;
34             break;
35         case SDL_KEYDOWN:
36             if(typingIndex == -1)
37                 return false;
38             return true;
39             break;
40         default:
41             return false;
42             break;
43     }
44 }

```

7.10.3.10 isReceiveEvent()

```

bool InputBox::isReceiveEvent (
    SDL_Event & e )

```

Definition at line 4 of file event.cpp.

```

5 {
6     switch(e.type)
7     {
8         case SDL_QUIT:
9             return false;
10            break;
11        default:
12            if(isButtonReceiveEvent(e) || isInputReceiveEvent(e))
13                return true;
14            return false;
15            break;
16    }
17 }

```

7.10.3.11 linking()

```

void InputBox::linking (
    std::string n )

```

Definition at line 17 of file constructor.cpp.

```

18 {
19     name = n;
20     importFromJson();
21 }

```

7.10.3.12 react()

```

Button * InputBox::react (
    SDL_Event & e )

```

Definition at line 58 of file event.cpp.

```

59 {
60     switch(e.type)
61     {
62         case SDL_KEYDOWN:
63             {
64                 if(typingIndex == -1)

```

```

65         return nullptr;
66         if(e.key.keysym.sym == SDLK_BACKSPACE)
67         {
68             inputs[typingIndex]->backspace();
69         }
70         if(SDLK_SPACE <= e.key.keysym.sym && e.key.keysym.sym <= SDLK_z)
71         {
72             inputs[typingIndex]->typing(e.key.keysym.sym);
73         }
74         return nullptr;
75         break;
76     }
77     case SDL_MOUSEMOTION:
78     {
79         for(auto& but : buts)
80             but->isHover(e.motion.x, e.motion.y);
81         return nullptr;
82         break;
83     }
84     case SDL_MOUSEBUTTONDOWN:
85     {
86         int i = 0;
87         for(auto& inp : inputs)
88         {
89             if(inp->isLieInside(e.motion.x, e.motion.y))
90             {
91                 typingIndex = i;
92                 return nullptr;
93             }
94             i++;
95         }
96         typingIndex = 0;
97         for(auto& but : buts)
98             if(but->isClicked(e.motion.x, e.motion.y))
99                 return but;
100         return nullptr;
101         break;
102     }
103     default:
104         return nullptr;
105     }
106 }

```

7.10.3.13 rendering()

```
void InputBox::rendering ( ) [virtual]
```

Reimplemented from [Object](#).

Definition at line 3 of file rendering.cpp.

```

4 {
5     Object::rendering();
6     for(auto& i : buts) i->rendering();
7     for(auto& i : inputs) i->rendering();
8 }

```

7.10.3.14 setDuplicate()

```
void InputBox::setDuplicate (
    int n,
    int m )
```

Definition at line 103 of file constructor.cpp.

```

104 {
105     this->n = n;
106     this->m = m;
107 }

```

7.10.3.15 setText() [1/2]

```
void InputBox::setText (
    int index,
    std::string text )
```

Definition at line 10 of file typing.cpp.

```
11 {
12     if (index != -1 && index < (int) inputs.size())
13     {
14         inputs[index]->setText (text);
15     }
16 }
```

7.10.3.16 setText() [2/2]

```
void InputBox::setText (
    std::string text )
```

Definition at line 18 of file typing.cpp.

```
19 {
20     if (typingIndex != -1)
21     {
22         inputs[typingIndex]->setText (text);
23     }
24 }
```

7.10.4 Member Data Documentation

7.10.4.1 m

```
int InputBox::m [protected]
```

Definition at line 38 of file inputbox.hpp.

7.10.4.2 n

```
int InputBox::n [protected]
```

Definition at line 38 of file inputbox.hpp.

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

- [include/inputbox.hpp](#)
- [src/inputbox/constructor.cpp](#)
- [src/inputbox/destructor.cpp](#)
- [src/inputbox/event.cpp](#)
- [src/inputbox/operator.cpp](#)
- [src/inputbox/rendering.cpp](#)
- [src/inputbox/typing.cpp](#)

7.11 maxHeap Class Reference

```
#include <maxheap.hpp>
```

Public Member Functions

- [maxHeap](#) ()
- [~maxHeap](#) ()
- void [init](#) (std::vector< int > &value)
- void [insert](#) (int value)
- void [pop](#) ()
- int [top](#) ()
- int [size](#) ()

Protected Member Functions

- void [heapify](#) (int index)
- void [swap](#) (int index1, int index2)
- bool [swapable](#) (int index1, int index2)

7.11.1 Detailed Description

Definition at line 7 of file maxheap.hpp.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 maxHeap()

```
maxHeap::maxHeap ( )
```

Definition at line 8 of file constructor.cpp.

```
9 {  
10  
11 }
```

7.11.2.2 ~maxHeap()

```
maxHeap::~~maxHeap ( )
```

Definition at line 7 of file destructor.cpp.

```
8 {  
9     for(auto &i : value)  
10     {  
11         delete i;  
12     }  
13 }
```

7.11.3 Member Function Documentation

7.11.3.1 heapify()

```
void maxHeap::heapify (
    int index ) [protected]
```

Definition at line 13 of file heapify.cpp.

```
14 {
15     int left = 2 * index + 1;
16     int right = 2 * index + 2;
17     int largest = index;
18
19     if(left < value.size() && swappable(index, left))
20     {
21         largest = left;
22     }
23
24     if(right < value.size() && swappable(largest, right))
25     {
26         largest = right;
27     }
28
29     if(largest != index)
30     {
31         swap(index, largest);
32         heapify(largest);
33     }
34 }
```

7.11.3.2 init()

```
void maxHeap::init (
    std::vector< int > & value )
```

Definition at line 4 of file init.cpp.

```
5 {
6     for(auto &i : value)
7     {
8         insert(i);
9     }
10 }
```

7.11.3.3 insert()

```
void maxHeap::insert (
    int value )
```

Definition at line 3 of file insert.cpp.

```
4 {
5     Node* node = new Node(value);
6     this->value.push_back(node);
7
8     int index = this->value.size() - 1;
9
10    if(index == 0) return ;
11
12    do
13    {
```

```
14         int parent = (index - 1) / 2;
15
16         if (swapable(parent, index))
17         {
18             swap(parent, index);
19         }
20
21         index = parent;
22     } while (index != 0);
23
24 }
```

7.11.3.4 pop()

```
void maxHeap::pop ( )
```

Definition at line 3 of file remove.cpp.

```
4 {
5     if (value.size() == 0)
6     {
7         return;
8     }
9
10    swap(0, value.size() - 1);
11    value.pop_back();
12
13    heapify(0);
14 }
```

7.11.3.5 size()

```
int maxHeap::size ( )
```

Definition at line 4 of file size.cpp.

```
5 {
6     return value.size();
7 }
```

7.11.3.6 swap()

```
void maxHeap::swap (
    int index1,
    int index2 ) [protected]
```

Definition at line 3 of file heapify.cpp.

```
4 {
5     std::swap(value[index1], value[index2]);
6 }
```

7.11.3.7 swapable()

```
bool maxHeap::swapable (
    int index1,
    int index2 ) [protected]
```

Definition at line 8 of file heapify.cpp.

```
9 {
10     return value[index1]->value < value[index2]->value;
11 }
```

7.11.3.8 top()

```
int maxHeap::top ( )
```

Definition at line 3 of file getmax.cpp.

```
4 {
5     return value[0]->value;
6 }
```

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

- [include/data_structures/maxheap.hpp](#)
- [src/maxheap/constructor.cpp](#)
- [src/maxheap/destructor.cpp](#)
- [src/maxheap/operator/getmax.cpp](#)
- [src/maxheap/operator/heapify.cpp](#)
- [src/maxheap/operator/init.cpp](#)
- [src/maxheap/operator/insert.cpp](#)
- [src/maxheap/operator/remove.cpp](#)
- [src/maxheap/operator/size.cpp](#)

7.12 minHeap Class Reference

heap class.

```
#include <minheap.hpp>
```

Public Member Functions

- [minHeap](#) (SDL_Renderer *render, std::mutex &m, TTF_Font *f, SDL_Rect v, int cap)
- [~minHeap](#) ()
- void [setmax](#) ()
- void [setmin](#) ()
- void [init](#) (std::vector< int > value)
- void [insert](#) (int value)
- void [pop](#) ()
- int [top](#) ()
- int [size](#) ()
- void [closeScript](#) ()
- bool [isReceiveEvent](#) (SDL_Event &e)
- [Button](#) * [react](#) (SDL_Event &e)
- void [rendering](#) ()
- void [setting](#) (SDL_Color c1, SDL_Color c2, SDL_Color c3, SDL_Color c4)

Protected Member Functions

- void [heapify](#) (int index)
- void [swap](#) (int index1, int index2)
- bool [swapable](#) (int index1, int index2)
- void [reallInsert](#) (int value)
- int [locating](#) (int id, int shiftDown, int shiftRight)
- void [renderLine](#) (Node *src, Node *dst)
- void [waitForStep](#) ()
- void [highlight](#) (std::vector< int > l)
- void [unhighlight](#) (std::vector< int > l)

7.12.1 Detailed Description

heap class.

Drawable heap.

Default is min heap.

Definition at line 25 of file minheap.hpp.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 minHeap()

```
minHeap::minHeap (
    SDL_Renderer * render,
    std::mutex & m,
    TTF_Font * f,
    SDL_Rect v,
    int cap )
```

Definition at line 9 of file constructor.cpp.

```
9                                     : ds_mutex(m)
10 {
11     render = r;
12     font = f;
13     viewport = v;
14     capacity = cap;
15     nodeColor = {20, 85, 185, 255};
16     fontColor = {255, 255, 255, 255};
17     edgesColor = {255, 255, 255, 255};
18     shiftX = 20;
19     shiftY = 20;
20     distanceX = 60;
21     distanceY = 80;
22     isMoving = false;
23     stepWait = 600;
24     isAnimate = false;
25     inverse = false;
26     std::string fontpath = PATH::ASSETS::FONTS_ + "nimbus-sans-1/regular.otf";
27     scriptFont = TTF_OpenFont(fontpath.c_str(), 18);
28
29     currentScript = nullptr;
30
31     Script* insert = new Script(render, scriptFont);
32     insert->linking("minheap/insert");
33     scripts[DATA_STRUCTURES_OPERATOR::INSERT] = insert;
```

```

34
35     Script* remove = new Script(render, scriptFont);
36     remove->linking("minheap/remove");
37     scripts[DATA_STRUCTURES_OPERATOR::DELETE] = remove;
38
39     Script* search = new Script(render, scriptFont);
40     search->linking("minheap/top");
41     scripts[DATA_STRUCTURES_OPERATOR::TOP] = search;
42
43     Script* init = new Script(render, scriptFont);
44     init->linking("minheap/init");
45
46     scripts[DATA_STRUCTURES_OPERATOR::INIT] = init;
47
48     Script* size = new Script(render, scriptFont);
49     size->linking("minheap/size");
50     scripts[DATA_STRUCTURES_OPERATOR::SIZE] = size;
51
52 }

```

7.12.2.2 ~minHeap()

```
minHeap::~minHeap ( )
```

Definition at line 7 of file destructor.cpp.

```

8 {
9     for(auto &i : value)
10     {
11         delete i;
12     }
13     if(scriptFont != nullptr) TTF_CloseFont(scriptFont);
14     for(auto &i : scripts)
15         delete i.second;
16     scripts.clear();
17 }

```

7.12.3 Member Function Documentation

7.12.3.1 closeScript()

```
void minHeap::closeScript ( )
```

Definition at line 74 of file event.cpp.

```

75 {
76     currentScript = nullptr;
77 }

```

7.12.3.2 heapify()

```
void minHeap::heapify (
    int index ) [protected]
```

Definition at line 13 of file heapify.cpp.

```
14 {
15     int left = 2 * index + 1;
16     int right = 2 * index + 2;
17     int largest = index;
18
19     if(left < value.size() && swappable(index, left))
20     {
21         largest = left;
22     }
23
24     if(right < value.size() && swappable(largest, right))
25     {
26         largest = right;
27     }
28
29     if(largest != index)
30     {
31         if(isAnimate)
32         {
33             animate_mutex.lock();
34             value[index]->sprite->highlight();
35             value[largest]->sprite->highlight();
36             animate_mutex.unlock();
37             waitForStep();
38         }
39         swap(index, largest);
40         if(isAnimate)
41         {
42             animate_mutex.lock();
43             value[index]->sprite->unhighlight();
44             value[largest]->sprite->unhighlight();
45             animate_mutex.unlock();
46         }
47         heapify(largest);
48     }
49 }
```

7.12.3.3 highlight()

```
void minHeap::highlight (
    std::vector< int > l ) [protected]
```

Definition at line 5 of file step.cpp.

```
6 {
7     if(isAnimate)
8     {
9         animate_mutex.lock();
10        for(int i = 0; i < l.size(); i++)
11        {
12            currentScript->highlight(l[i]);
13        }
14        animate_mutex.unlock();
15    }
16 }
```

7.12.3.4 init()

```
void minHeap::init (
    std::vector< int > value )
```

Definition at line 4 of file init.cpp.

```
5 {
6     currentScript = scripts[DATA_STRUCTURES_OPERATOR::INIT];
7     this->value.clear();
8     for(auto &i : value)
9     {
10         realInsert(i);
11     }
12 }
```

7.12.3.5 insert()

```
void minHeap::insert (
    int value )
```

Definition at line 61 of file insert.cpp.

```
62 {
63     shiftX = 20;
64     shiftY = 20;
65     currentScript = scripts[DATA_STRUCTURES_OPERATOR::INSERT];
66     isAnimate = true;
67
68     highlight({0});
69     waitForStep();
70     unhighlight({0});
71
72     realInsert(value);
73 }
```

7.12.3.6 isReceiveEvent()

```
bool minHeap::isReceiveEvent (
    SDL_Event & e )
```

Definition at line 3 of file event.cpp.

```
4 {
5     std::lock_guard<std::mutex> lk(animate_mutex);
6     switch(e.type)
7     {
8         case SDL_MOUSEBUTTONDOWN:
9             if(currentScript != nullptr && currentScript->isReceiveEvent(e)) return true;
10             if(e.motion.x < viewport.x || viewport.x + viewport.w < e.motion.x) return false;
11             if(e.motion.y < viewport.y || viewport.y + viewport.h < e.motion.y) return false;
12             if(e.button.button == SDL_BUTTON_LEFT) return false;
13             if(value.empty()) return false;
14             return true;
15             break;
16         case SDL_MOUSEMOTION:
17             if(isMoving) return true;
18             if(currentScript == nullptr) return false;
19             if(currentScript->isReceiveEvent(e)) return true;
20             return false;
21             break;
22         default:
23             return false;
24             break;
25     }
26 }
```

7.12.3.7 locating()

```
int minHeap::locating (
    int id,
    int shiftDown,
    int shiftRight ) [protected]
```

Definition at line 4 of file rendering.cpp.

```
5 {
6     if(id < 0 || id >= value.size())
7     {
8         int shift = log2(value.size()) - shiftDown + 1;
9         return std::max(0, (1 << shift) - 1);
10    }
11
12    int left = std::max(0, locating(id * 2 + 1, shiftDown + 1, shiftRight));
13    Node* node = value[id];
14    if(node->sprite != nullptr)
15    {
16        node->sprite->locatingX(shiftX + shiftRight * distanceX + left * distanceX);
17        node->sprite->locatingY(shiftY + shiftDown * distanceY);
18        node->sprite->aligning(HORIZONTAL_ALIGN::CENTER, VERTICAL_ALIGN::CENTER);
19    }
20    int right = locating(id * 2 + 2, shiftDown + 1, shiftRight + left + 1);
21
22    int shift = log2(value.size()) - shiftDown + 1;
23    return left + right + 1;
24 }
```

7.12.3.8 pop()

```
void minHeap::pop ( )
```

Definition at line 5 of file remove.cpp.

```
6 {
7     if(!value.empty()) valuesReachedZero = false;
8     if(valuesReachedZero) return ;
9     currentScript = scripts[DATA_STRUCTURES_OPERATOR::DELETE];
10    isAnimate = true;
11    highlight({0});
12    waitForStep();
13    unhighlight({0});
14
15    if(value.size() == 0)
16    {
17        valuesReachedZero = true;
18        highlight({1, 2, 3});
19        waitForStep();
20        unhighlight({1, 2, 3});
21        return;
22    }
23    if(isAnimate)
24    {
25        animate_mutex.lock();
26        value[0]->sprite->highlight();
27        value.back()->sprite->highlight();
28        animate_mutex.unlock();
29        waitForStep();
30    }
31    swap(0, value.size() - 1);
32
33    if(isAnimate)
34    {
35        animate_mutex.lock();
36        value[0]->sprite->unhighlight();
37        value.back()->sprite->unhighlight();
38        animate_mutex.unlock();
39    }
40
41    highlight({5});
42    waitForStep();
43    unhighlight({5});
44
45    value.pop_back();
```

```

46
47     highlight({6});
48     waitForStep();
49
50     heapify(0);
51
52     unhighlight({6});
53     waitForStep();
54 }

```

7.12.3.9 react()

```

Button * minHeap::react (
    SDL_Event & e )

```

Definition at line 28 of file event.cpp.

```

29 {
30     std::lock_guard<std::mutex> lk(animate_mutex);
31     switch(e.type)
32     {
33         case SDL_MOUSEBUTTONDOWN:
34             if(currentScript != nullptr && currentScript->isReceiveEvent(e))
35             {
36                 return currentScript->react(e);
37             }
38             if(isMoving)
39             {
40                 isMoving = false;
41                 int dx = e.motion.x - lastMousePressed.x;
42                 int dy = e.motion.y - lastMousePressed.y;
43                 shiftX += dx;
44                 shiftY += dy;
45             }else
46             {
47                 isMoving = true;
48                 lastMousePressed.x = e.motion.x;
49                 lastMousePressed.y = e.motion.y;
50             }
51             return nullptr;
52             break;
53         case SDL_MOUSEMOTION:
54             {
55                 if(currentScript != nullptr && currentScript->isReceiveEvent(e))
56                     return currentScript->react(e);
57                 if(!isMoving) return nullptr;
58                 int dx = e.motion.x - lastMousePressed.x;
59                 int dy = e.motion.y - lastMousePressed.y;
60                 lastMousePressed.x = e.motion.x;
61                 lastMousePressed.y = e.motion.y;
62                 shiftX += dx;
63                 shiftY += dy;
64                 return nullptr;
65                 break;
66             }
67         default:
68             return nullptr;
69             break;
70     }
71     return nullptr;
72 }

```

7.12.3.10 realInsert()

```

void minHeap::realInsert (
    int value ) [protected]

```

Definition at line 4 of file insert.cpp.

```

5 {
6     if(inverse) value *= -1;

```

```

7   Sprite* sprite = new Sprite(render);
8   sprite->setFont(font);
9   sprite->linking("AVL/node");
10  sprite->setFontColor(fontColor);
11  sprite->coloring(nodeColor);
12  sprite->setText(NUMBER::intToString(value));
13  Node* node = new Node(value, sprite);
14
15  this->value.push_back(node);
16
17  int index = this->value.size() - 1;
18  highlight({1, 2, 3});
19  waitForStep();
20  unhighlight({1, 2, 3});
21
22  if(index == 0)
23  {
24      highlight({5, 6, 7});
25      waitForStep();
26      unhighlight({5, 6, 7});
27      return ;
28  }
29  highlight({9});
30  do
31  {
32      int parent = (index - 1) / 2;
33
34      if(isAnimate)
35      {
36          animate_mutex.lock();
37          this->value[index]->sprite->highlight();
38          this->value[parent]->sprite->highlight();
39          animate_mutex.unlock();
40          waitForStep();
41      }
42      if(swappable(parent, index))
43      {
44          highlight({11, 12, 13});
45          waitForStep();
46          swap(parent, index);
47          unhighlight({11, 12, 13});
48      }
49      if(isAnimate)
50      {
51          animate_mutex.lock();
52          this->value[index]->sprite->unhighlight();
53          this->value[parent]->sprite->unhighlight();
54          animate_mutex.unlock();
55      }
56      index = parent;
57  }while(index !=0);
58  unhighlight({9});
59 }

```

7.12.3.11 rendering()

```
void minHeap::rendering ( )
```

Definition at line 39 of file rendering.cpp.

```

40 {
41     SDL_RenderSetViewport(render, &viewport);
42     std::lock_guard<std::mutex> lock(animate_mutex);
43     locating(0, 0, 0);
44
45     for(int i = 1; i < value.size(); i++)
46     {
47         Node* node = value[i];
48         if(node->sprite != nullptr)
49         {
50             Node* parent = value[(i - 1) / 2];
51             renderLine(parent, node);
52         }
53     }
54     for(auto& i : value)
55     {
56         if(i->sprite != nullptr)
57             i->sprite->rendering();
58     }
59     if(currentScript != nullptr)

```

```
60     {
61         SDL_RenderSetViewport(render, nullptr);
62         currentScript->rendering();
63     }
64
65 }
```

7.12.3.12 renderLine()

```
void minHeap::renderLine (
    Node * src,
    Node * dst ) [protected]
```

Definition at line 26 of file rendering.cpp.

```
27 {
28     SDL_Point psrc = {src->sprite->getX() + src->sprite->getW() / 2, src->sprite->getY() +
29     src->sprite->getH() / 2};
30     SDL_Point pdst = {dst->sprite->getX() + dst->sprite->getW() / 2, dst->sprite->getY() +
31     dst->sprite->getH() / 2};
32
33     SDL_SetRenderDrawColor(render, edgesColor.r, edgesColor.g, edgesColor.b, edgesColor.a);
34     for(int i = -1; i <= 1; i++)
35     {
36         for(int j = -1; j <= 1; j++)
37             SDL_RenderDrawLine(render, psrc.x + i, psrc.y + j, pdst.x + i, pdst.y + j);
38     }
39 }
```

7.12.3.13 setmax()

```
void minHeap::setmax ( )
```

Definition at line 59 of file constructor.cpp.

```
60 {
61     inverse = true;
62 }
```

7.12.3.14 setmin()

```
void minHeap::setmin ( )
```

Definition at line 54 of file constructor.cpp.

```
55 {
56     inverse = false;
57 }
```


7.12.3.15 setting()

```
void minHeap::setting (
    SDL_Color c1,
    SDL_Color c2,
    SDL_Color c3,
    SDL_Color c4 )
```

Definition at line 64 of file constructor.cpp.

```
65 {
66     bgColor = c1;
67     nodeColor = c2;
68     fontColor = c3;
69     edgesColor = c4;
70
71     for(auto i : value)
72     {
73         i->sprite->setFontColor(fontColor);
74         i->sprite->coloring(nodeColor);
75     }
76 }
```

7.12.3.16 size()

```
int minHeap::size ( )
```

Definition at line 4 of file size.cpp.

```
5 {
6     currentScript = scripts[DATA_STRUCTURES_OPERATOR::SIZE];
7     return value.size();
8 }
```

7.12.3.17 swap()

```
void minHeap::swap (
    int index1,
    int index2 ) [protected]
```

Definition at line 3 of file heapify.cpp.

```
4 {
5     std::swap(value[index1], value[index2]);
6 }
```

7.12.3.18 swapable()

```
bool minHeap::swapable (
    int index1,
    int index2 ) [protected]
```

Definition at line 8 of file heapify.cpp.

```
9 {
10     return value[index1]->value > value[index2]->value;
11 }
```

7.12.3.19 top()

```
int minHeap::top ( )
```

Definition at line 4 of file getmin.cpp.

```
5 {
6     currentScript = scripts[DATA_STRUCTURES_OPERATOR::TOP];
7     isAnimate = true;
8     highlight({0, 1, 2, 3, 4, 5});
9     waitForStep();
10    unhighlight({0, 1, 2, 3, 4, 5});
11
12    if(value.size() == 0)
13        return INT_MAX;
14    return value[0]->value;
15 }
```

7.12.3.20 unhighlight()

```
void minHeap::unhighlight (
    std::vector< int > l ) [protected]
```

Definition at line 18 of file step.cpp.

```
19 {
20     if(isAnimate)
21     {
22         animate_mutex.lock();
23         for(int i = 0; i < l.size(); i++)
24         {
25             currentScript->unhighlight(l[i]);
26         }
27         animate_mutex.unlock();
28     }
29 }
```

7.12.3.21 waitForStep()

```
void minHeap::waitForStep ( ) [protected]
```

Definition at line 31 of file step.cpp.

```
32 {
33     if(isAnimate)
34     {
35         ds_mutex.unlock();
36         std::this_thread::sleep_for(std::chrono::milliseconds(stepWait));
37         ds_mutex.lock();
38     }
39     std::lock_guard<std::mutex> pause_lock(pause_mutex);
40     if(isPause == false)
41     {
42         return ;
43     }
44
45     ds_mutex.unlock();
46     std::unique_lock<std::mutex> lk(step_mutex);
47     step_cv.wait(lk, [&]{return isQueue == true;});
48     isQueue = false;
49     ds_mutex.lock();
50 }
```

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

- [include/data_structures/minheap.hpp](#)

- [src/minheap/constructor.cpp](#)
- [src/minheap/destructor.cpp](#)
- [src/minheap/event.cpp](#)
- [src/minheap/operator/getmin.cpp](#)
- [src/minheap/operator/heapify.cpp](#)
- [src/minheap/operator/init.cpp](#)
- [src/minheap/operator/insert.cpp](#)
- [src/minheap/operator/remove.cpp](#)
- [src/minheap/operator/size.cpp](#)
- [src/minheap/rendering.cpp](#)
- [src/minheap/step.cpp](#)

7.13 MyWindow Class Reference

Window class class that create a window and manage it.

```
#include <window.hpp>
```

Public Member Functions

- [MyWindow \(\)](#)
- [~MyWindow \(\)](#)
- [bool isClosed \(\)](#)
- [bool isOpen \(\)](#)
- [void run \(\)](#)

Protected Member Functions

- [void initSDL2 \(\)](#)
- [void rendering \(\)](#)
- [void interacting \(\)](#)
- [void getEvent \(\)](#)
- [void react \(Button *but\)](#)
- [void setDisplay \(std::string name\)](#)
- [void setInputBox \(std::string name\)](#)
- [void setDataType \(DATA_STRUCTURES_TYPE type\)](#)
- [void runOperator \(\)](#)
- [void getDataFromFile \(DATA_STRUCTURES_TYPE type\)](#)
- [void getDataFromRandom \(DATA_STRUCTURES_TYPE type\)](#)

7.13.1 Detailed Description

Window class class that create a window and manage it.

Finite state machine

handle thread

Definition at line 26 of file window.hpp.

7.13.2 Constructor & Destructor Documentation

7.13.2.1 MyWindow()

MyWindow::MyWindow ()

Definition at line 28 of file constructor.cpp.

```

29 {
30     HEIGHT = 635;
31     WIDTH = 1200;
32     status = WINDOW_STATUS::IS_CLOSED;
33     FPS = 60;
34     viewport = SDL_Rect({0, 0, WIDTH, HEIGHT});
35
36     window = nullptr;
37     render = nullptr;
38     ds = nullptr;
39     inputbox = nullptr;
40
41     initSDL2();
42
43     std::string fontpath = PATH::ASSETS::FONTS_ + "nimbus-sans-l/regular.otf";
44     myfont = TTF_OpenFont(fontpath.c_str(), 24);
45
46     display_pool["nullptr"] = nullptr;
47     inputbox_pool["nullptr"] = nullptr;
48     ds_pool[DATA_STRUCTURES_TYPE::NONE] = nullptr;
49
50     setDisplay(DISPLAY::HOME_);
51 }
```

7.13.2.2 ~MyWindow()

MyWindow::~MyWindow ()

Definition at line 5 of file destructor.cpp.

```

6 {
7     if(render != nullptr)
8     {
9         SDL_DestroyRenderer(render);
10        render = nullptr;
11    }
12    if(window != nullptr)
13    {
14        SDL_DestroyWindow(window);
15        window = nullptr;
16    }
17    if(myfont != nullptr) TTF_CloseFont(myfont);
18
19    for(auto &i : thread_pool)
20    {
21        if(i.joinable())
22        {
23            i.join();
24        }
25    }
26
27    for(auto &i : display_pool)
28    {
29        delete i.second;
30    }
31
32    for(auto &i : ds_pool)
33    {
34        delete i.second;
35    }
36
37    for(auto &i : inputbox_pool)
38    {
39        delete i.second;
40    }
41
42    TTF_Quit();
43    IMG_Quit();
44    SDL_Quit();
45 }
```

7.13.3 Member Function Documentation

7.13.3.1 getDataFromFile()

```
void MyWindow::getDataFromFile (
    DATA_STRUCTURES_TYPE type ) [protected]
```

Definition at line 123 of file updating.cpp.

```
124 {
125     switch(type)
126     {
127         case DATA_STRUCTURES_TYPE::AVL:{
128             std::vector<std::string> mem =
129             FILEE::readFile(PATH::SAVING::AVL_);
130             if(mem.empty()) return ;
131             renderMutex.lock();
132             inputbox->setText(1, mem[0]);
133             renderMutex.unlock();
134             break;
135         }
136         case DATA_STRUCTURES_TYPE::TRIE:{
137             std::vector<std::string> mem =
138             FILEE::readFile(PATH::SAVING::TRIE_);
139             if(mem.empty()) return ;
140             std::string total = "";
141             for(int i = 0; i < mem.size(); i++)
142             {
143                 total += mem[i];
144                 if(i != mem.size() - 1) total += " ";
145             }
146             renderMutex.lock();
147             inputbox->setText(1, total);
148             renderMutex.unlock();
149             break;
150         }
151         case DATA_STRUCTURES_TYPE::GRAPH:{
152             std::ifstream fin(PATH::SAVING::GRAPH_);
153             renderMutex.lock();
154             fin >> ds->capacity;
155             renderMutex.unlock();
156             setInputBox("graph/edges");
157             renderMutex.lock();
158             for(int i = 0; i < ds->capacity; i++)
159                 for(int j = 0; j < ds->capacity; j++)
160                 {
161                     int x;
162                     fin >> x;
163                     inputbox->setText(i * ds->capacity + j + 1,
164                     NUMBER::intToString(x));
165                 }
166             renderMutex.unlock();
167             fin.close();
168             break;
169         }
170         case DATA_STRUCTURES_TYPE::BTREE_4TH:{
171             std::vector<std::string> mem =
172             FILEE::readFile(PATH::SAVING::BTREE_4TH_);
173             break;
174         }
175         case DATA_STRUCTURES_TYPE::MIN_HEAP:{
176             std::vector<std::string> mem =
177             FILEE::readFile(PATH::SAVING::MIN_HEAP_);
178             renderMutex.lock();
179             inputbox->setText(1, mem[0]);
180             renderMutex.unlock();
181             break;
182         }
183         case DATA_STRUCTURES_TYPE::MAX_HEAP:{
184             std::vector<std::string> mem =
185             FILEE::readFile(PATH::SAVING::MAX_HEAP_);
186             renderMutex.lock();
187             inputbox->setText(1, mem[0]);
188             renderMutex.unlock();
189             break;
190         }
191     }
```

```

187                                     renderMutex.lock();
188                                     inputbox->setText(1, mem[0]);
189                                     renderMutex.unlock();
190                                     break;
191                                 }
192     case DATA_STRUCTURES_TYPE::HASH_TABLE:{
193         std::vector<std::string> mem =
194     FILEE::readFile(PATH::SAVING::HASH_TABLE_);
195         renderMutex.lock();
196         inputbox->setText(1, mem[0]);
197         inputbox->setText(2, mem[1]);
198         renderMutex.unlock();
199         break;
200     }
201     case DATA_STRUCTURES_TYPE::NONE:{
202         break;
203     }
204 }
205 }

```

7.13.3.2 getDataFromRandom()

```

void MyWindow::getDataFromRandom (
    DATA_STRUCTURES_TYPE type ) [protected]

```

Definition at line 207 of file updating.cpp.

```

208 {
209     switch(type)
210     {
211     case DATA_STRUCTURES_TYPE::AVL:{
212         int n = RANDOM::getInt(1, 32);
213         std::string mem = RANDOM::getInt(n, 1, 999);
214         renderMutex.lock();
215         inputbox->setText(1, mem);
216         renderMutex.unlock();
217         break;
218     }
219     case DATA_STRUCTURES_TYPE::TRIE:{
220         std::vector<std::string> mem =
221     FILEE::readFile(PATH::SAVING::TRIE_);
222         break;
223     }
224     case DATA_STRUCTURES_TYPE::GRAPH:{
225         std::vector<std::string> mem =
226     FILEE::readFile(PATH::SAVING::GRAPH_);
227         break;
228     }
229     case DATA_STRUCTURES_TYPE::BTREE_4TH:{
230         std::vector<std::string> mem =
231     FILEE::readFile(PATH::SAVING::BTREE_4TH_);
232         break;
233     }
234     case DATA_STRUCTURES_TYPE::MIN_HEAP:{
235         std::vector<std::string> mem =
236     FILEE::readFile(PATH::SAVING::MIN_HEAP_);
237         break;
238     }
239     case DATA_STRUCTURES_TYPE::MAX_HEAP:{
240         std::vector<std::string> mem =
241     FILEE::readFile(PATH::SAVING::MAX_HEAP_);
242         break;
243     }
244     case DATA_STRUCTURES_TYPE::HASH_TABLE:{
245         std::vector<std::string> mem =
246     FILEE::readFile(PATH::SAVING::HASH_TABLE_);
247         break;
248     }
249     case DATA_STRUCTURES_TYPE::NONE:{
250         break;
251     }
252 }
253 }

```

7.13.3.3 getEvent()

```
void MyWindow::getEvent ( ) [protected]
```

Definition at line 43 of file event.cpp.

```

44 {
45     SDL_Event event;
46     while(isOpen())
47     {
48         while(SDL_PollEvent(&event))
49         {
50             switch(event.type)
51             {
52                 case SDL_QUIT:
53                 {
54                     event_mutex.lock();
55
56                     event_pool.push(event);
57
58                     event_mutex.unlock();
59                     break;
60                 }
61                 default:
62                 {
63
64                     bool isds = false;
65                     bool isdisplay = false;
66                     bool isinputbox = false;
67
68                     event_mutex.lock();
69
70                     renderMutex.lock();
71
72                     if(ds != nullptr && ds->isReceiveEvent(event))
73                         isds = true;
74
75                     if(current_display->isReceiveEvent(event))
76                         isdisplay = true;
77
78                     if(inputbox != nullptr && inputbox->isReceiveEvent(event))
79                         isinputbox = true;
80
81                     renderMutex.unlock();
82
83                     if(isds || isdisplay || isinputbox)
84                         event_pool.push(event);
85
86                     event_mutex.unlock();
87
88                     EVcond.notify_one();
89                     std::this_thread::sleep_for(std::chrono::milliseconds(10));
90                     break;
91                 }
92             }
93         }
94     }
95 }
```

7.13.3.4 initSDL2()

```
void MyWindow::initSDL2 ( ) [protected]
```

Definition at line 5 of file constructor.cpp.

```

6 {
7     SDL_Init(SDL_INIT_VIDEO | SDL_INIT_AUDIO);
8
9     window = SDL_CreateWindow(
10         "Dr Duck",
11         SDL_WINDOWPOS_CENTERED,
12         SDL_WINDOWPOS_CENTERED,
13         WIDTH,
14         HEIGHT,
15         SDL_WINDOW_SHOWN
16     );
17
18     render = SDL_CreateRenderer(
```

```

19         window,
20         -1,
21         SDL_RENDERER_ACCELERATED
22     );
23
24     IMG_Init(IMG_INIT_PNG | IMG_INIT_JPG);
25     TTF_Init();
26 }

```

7.13.3.5 interacting()

```
void MyWindow::interacting ( ) [protected]
```

Definition at line 4 of file event.cpp.

```

5 {
6     SDL_Event event;
7     std::unique_lock<std::mutex> ul(event_mutex);
8     while(isOpen())
9     {
10         EVcond.wait(ul, [&]() {return !event_pool.empty();});
11         event = event_pool.front();
12         event_pool.pop();
13
14         switch(event.type)
15         {
16             case SDL_QUIT:
17                 status_mutex.lock();
18                 status = WINDOW_STATUS::IS_CLOSED;
19                 status_mutex.unlock();
20                 step_cond.notify_all();
21                 break;
22             default:
23                 Button* but = nullptr;
24                 renderMutex.lock();
25
26                 if(but == nullptr && inputbox != nullptr && inputbox->isReceiveEvent(event))
27                     but = inputbox->react(event);
28
29                 if(ds != nullptr && ds->isReceiveEvent(event) && but == nullptr)
30                     but = ds->react(event);
31
32                 if(but == nullptr && but == nullptr && current_display->isReceiveEvent(event))
33                     but = current_display->react(event);
34
35                 renderMutex.unlock();
36
37                 react(but);
38                 break;
39         }
40         std::this_thread::sleep_for(std::chrono::milliseconds(5));
41     }
42 }

```

7.13.3.6 isClosed()

```
bool MyWindow::isClosed ( )
```

Definition at line 3 of file status.cpp.

```

4 {
5     std::lock_guard<std::mutex> lg(status_mutex);
6     return status == WINDOW_STATUS::IS_CLOSED;
7 }

```


7.13.3.7 isOpen()

```
bool MyWindow::isOpen ( )
```

Definition at line 9 of file status.cpp.

```
10 {
11     std::lock_guard<std::mutex> lg(status_mutex);
12     return status == WINDOW_STATUS::IS_OPEN;
13 }
```

7.13.3.8 react()

```
void MyWindow::react (
    Button * but ) [protected]
```

Definition at line 249 of file updating.cpp.

```
250 {
251     if(but == nullptr) return ;
252     switch (but->getAction())
253     {
254         case BUTTON_ACTION::CHANGE_SCREEN:
255             setDisplay(but->getNextScreen());
256             if(but->getDataType() != DATA_STRUCTURES_TYPE::NONE)
257                 setDataType(but->getDataType());
258             break;
259
260         case BUTTON_ACTION::INIT:
261             setInputBox(ds->getName() + "/init");
262             break;
263
264         case BUTTON_ACTION::SETTING:
265             setInputBox("setting");
266             break;
267
268         case BUTTON_ACTION::INSERT:
269             setInputBox(ds->getName() + "/insert");
270             break;
271
272         case BUTTON_ACTION::DELETE:
273             setInputBox(ds->getName() + "/remove");
274             break;
275
276         case BUTTON_ACTION::SEARCH:
277             setInputBox(ds->getName() + "/search");
278             break;
279
280         case BUTTON_ACTION::SIZE:
281             setInputBox(ds->getName() + "/size");
282             break;
283
284         case BUTTON_ACTION::TOP:
285             setInputBox(ds->getName() + "/top");
286             break;
287
288         case BUTTON_ACTION::CONNECTED_COMPONENTS:
289             setInputBox(ds->getName() + "/scc");
290             break;
291
292         case BUTTON_ACTION::DIJKSTRA:
293             setInputBox(ds->getName() + "/dijkstra");
294             break;
295
296         case BUTTON_ACTION::MST:
297             setInputBox(ds->getName() + "/mst");
298             break;
299
300         case BUTTON_ACTION::GO_BACK:
301             //ds->goBack();
302             break;
303
304         case BUTTON_ACTION::GO_NEXT:
305             //ds->goNext();
306             break;
307
308     }
```

```

309
310     case BUTTON_ACTION::GO_ON:
311
312         //ds->goOn();
313         break;
314
315     case BUTTON_ACTION::GO_OFF:
316
317         //ds->goOff();
318         break;
319
320     case BUTTON_ACTION::SPEED_UP:
321
322         //ds->speedUp();
323         break;
324
325     case BUTTON_ACTION::SLOW_DOWN:
326
327         //ds->slowDown();
328         break;
329
330     case BUTTON_ACTION::DONE:
331
332         step_mutex.lock();
333         isQueuingStep = true;
334         step_mutex.unlock();
335         step_cond.notify_one();
336
337         break;
338
339     case BUTTON_ACTION::EDGES:
340
341         int n;
342         renderMutex.lock();
343
344         n = NUMBER::stringToInt(inputbox->getText(1));
345         n = std::min(n, 9);
346         n = std::max(n, 2);
347         if(ds != nullptr) ds->capacity = n;
348
349         renderMutex.unlock();
350         setInputBox("graph/edges");
351         break;
352
353     case BUTTON_ACTION::CLOSE:
354
355         renderMutex.lock();
356         ds->closeScript();
357         renderMutex.unlock();
358         break;
359
360     case BUTTON_ACTION::RANDOM:{
361         int n = RANDOM::getInt(1, 32);
362         std::string mem = RANDOM::getInt(n, 1, 999);
363
364         renderMutex.lock();
365         inputbox->setText(1, mem);
366
367         renderMutex.unlock();
368         break;
369     }
370     case BUTTON_ACTION::RANDOM2: {
371
372         renderMutex.lock();
373         inputbox->setText(1, RANDOM::getInt(1, 1, 999));
374         renderMutex.unlock();
375         break;
376     }
377     case BUTTON_ACTION::RANDOM3:{
378         int n = RANDOM::getInt(1, 16);
379         int m = RANDOM::getInt(1, 16);
380         char upperbound = RANDOM::getInt(97 + 5, 97 + 25);
381         std::string mem;
382         mem = RANDOM::getString(m, 'a', upperbound);
383
384         for(int i = 1; i < n; i++)
385             mem += " " + RANDOM::getString(m, 'a', upperbound);
386
387         renderMutex.lock();
388         inputbox->setText(1, mem);
389         renderMutex.unlock();
390         break;
391     }
392     case BUTTON_ACTION::RANDOM4:
393     {
394         int m = RANDOM::getInt(1, 16);
395         renderMutex.lock();

```

```

396         inputbox->setText(1, RANDOM::getString(m, 'a', 'z'));
397         renderMutex.unlock();
398         break;
399     }
400     case BUTTON_ACTION::RANDOM5: {
401         int n = RANDOM::getInt(1, 64);
402         std::string mem = RANDOM::getInt(n, 1, 999);
403
404         renderMutex.lock();
405         inputbox->setText(2, mem);
406         renderMutex.unlock();
407         break;
408     }
409     case BUTTON_ACTION::RANDOM6: {
410         renderMutex.lock();
411         for(int i = 0; i < ds->capacity; i++)
412             for(int j = 0; j < ds->capacity; j++)
413             {
414                 if(i == j) continue;
415                 if(RANDOM::getInt(1, 100) <= 30)
416                     inputbox->setText(i * ds->capacity + j + 1,
RANDOM::getInt(1, 1, 99));
417                 else
418                     inputbox->setText(i * ds->capacity + j + 1, "0");
419             }
420         renderMutex.unlock();
421         break;
422     }
423     case BUTTON_ACTION::RANDOM7: {
424         renderMutex.lock();
425         inputbox->setText(1, NUMBER::intToString(RANDOM::getInt(0,
ds->capacity - 1)));
426         renderMutex.unlock();
427         break;
428     }
429     case BUTTON_ACTION::RANDOM8: {
430         renderMutex.lock();
431         inputbox->setText(2, NUMBER::intToString(RANDOM::getInt(0,
ds->capacity - 1)));
432         renderMutex.unlock();
433         break;
434     }
435
436     case BUTTON_ACTION::FILE :{
437         getDataFromFile(ds->getDataType());
438         break;
439     }
440
441     default:
442         break;
443 }
444 }

```

7.13.3.9 rendering()

void MyWindow::rendering () [protected]

Definition at line 4 of file rendering.cpp.

```

5 {
6     while(isOpen())
7     {
8         if(!renderMutex.try_lock())
9         {
10             std::this_thread::sleep_for(std::chrono::milliseconds(5));
11             continue;
12         }
13
14         SDL_RenderClear(render);
15
16         current_display->rendering();
17
18         if(ds != nullptr) ds->rendering();
19
20         SDL_RenderSetViewport(render, &viewport);
21
22         if(inputbox != nullptr) inputbox->rendering();
23
24         SDL_RenderPresent(render);
25

```

```

26         renderMutex.unlock();
27         std::this_thread::sleep_for(std::chrono::milliseconds(5));
28     }
29 }

```

7.13.3.10 run()

```
void MyWindow::run ( )
```

Definition at line 4 of file running.cpp.

```

5 {
6     status = WINDOW_STATUS::IS_OPEN;
7     thread_pool.push_back(std::thread(&MyWindow::rendering, this));
8     thread_pool.push_back(std::thread(&MyWindow::getEvent, this));
9     thread_pool.push_back(std::thread(&MyWindow::runOperator, this));
10
11     interacting();
12
13     for(auto& thread : thread_pool)
14     {
15         thread.join();
16     }
17 }

```

7.13.3.11 runOperator()

```
void MyWindow::runOperator ( ) [protected]
```

Definition at line 60 of file updating.cpp.

```

61 {
62     while(isOpen())
63     {
64         std::unique_lock<std::mutex> lock(step_mutex);
65         step_cond.wait(lock, [&]() {return isClosed() || isQueuingStep;});
66         if(isClosed()) break;
67
68         isQueuingStep = false;
69
70         InputBox* temp;
71
72         renderMutex.lock();
73         if(inputbox == nullptr)
74         {
75             renderMutex.unlock();
76             continue;
77         }
78
79         temp = inputbox;
80         renderMutex.unlock();
81
82         setInputBox("nullptr");
83
84         renderMutex.lock();
85
86         switch(temp->getOperator())
87         {
88             case DATA_STRUCTURES_OPERATOR::INIT:
89             {
90                 ds->init(temp);
91                 break;
92             }
93             case DATA_STRUCTURES_OPERATOR::INSERT:
94                 ds->insert(temp);
95                 break;
96             case DATA_STRUCTURES_OPERATOR::DELETE:
97                 ds->remove(temp);
98                 break;
99             case DATA_STRUCTURES_OPERATOR::SEARCH:
100                 ds->search(temp);
101                 break;

```

```

102         case DATA_STRUCTURES_OPERATOR::TOP:
103             ds->top();
104             break;
105         case DATA_STRUCTURES_OPERATOR::SIZE:
106             ds->size();
107         case DATA_STRUCTURES_OPERATOR::SETTING:
108             ds->setting(temp);
109         case DATA_STRUCTURES_OPERATOR::DIJKSTRA:
110             ds->dijkstra(temp);
111             break;
112         case DATA_STRUCTURES_OPERATOR::MST:
113             ds->mst();
114             break;
115         case DATA_STRUCTURES_OPERATOR::SCC:
116             ds->scc();
117             break;
118     }
119     renderMutex.unlock();
120 }
121 }

```

7.13.3.12 setDataType()

```

void MyWindow::setDataType (
    DATA_STRUCTURES_TYPE type ) [protected]

```

Definition at line 47 of file updating.cpp.

```

48 {
49     renderMutex.lock();
50     if(ds_pool.find(type) == ds_pool.end())
51     {
52         ds_pool[type] = new DataStructures(render, myfont, renderMutex);
53         ds_pool[type]->setDataType(type);
54     }
55     ds = ds_pool[type];
56
57     renderMutex.unlock();
58 }

```

7.13.3.13 setDisplay()

```

void MyWindow::setDisplay (
    std::string name ) [protected]

```

Definition at line 4 of file updating.cpp.

```

5 {
6     setInputBox("nullptr");
7     renderMutex.lock();
8
9     ds = nullptr;
10    if(display_pool.find(name) == display_pool.end())
11    {
12        display_pool[name] = new Display(render, viewport);
13        display_pool[name]->linking(name);
14    }
15    current_display = display_pool[name];
16
17    renderMutex.unlock();
18 }

```

7.13.3.14 setInputBox()

```
void MyWindow::setInputBox (
    std::string name ) [protected]
```

Definition at line 20 of file updating.cpp.

```
21 {
22     renderMutex.lock();
23     if (name == "graph/edges")
24     {
25         auto i = inputbox_pool.find("graph/edges");
26         if (i != inputbox_pool.end())
27         {
28             auto t = i->second;
29             inputbox_pool.erase(i);
30             delete t;
31         }
32     }
33     inputbox_pool["graph/edges"] = new InputBox(render, myfont);
34     inputbox_pool["graph/edges"]->setDuplicate(ds->capacity, ds->capacity);
35     inputbox_pool["graph/edges"]->linking("graph/edges");
36 }
37 }
38 if (inputbox_pool.find(name) == inputbox_pool.end())
39 {
40     inputbox_pool[name] = new InputBox(render, myfont);
41     inputbox_pool[name]->linking(name);
42 }
43 inputbox = inputbox_pool[name];
44 renderMutex.unlock();
45 }
```

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

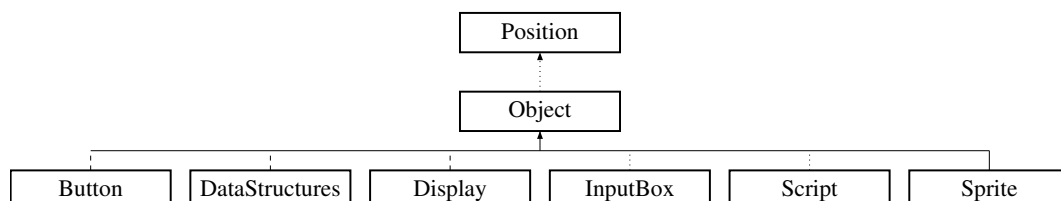
- [include/window.hpp](#)
- [src/window/constructor.cpp](#)
- [src/window/destructor.cpp](#)
- [src/window/event.cpp](#)
- [src/window/rendering.cpp](#)
- [src/window/running.cpp](#)
- [src/window/status.cpp](#)
- [src/window/updating.cpp](#)

7.14 Object Class Reference

Class that represent shape, image from files, text. Smallest drawable unit.

```
#include <object.hpp>
```

Inheritance diagram for Object:



Public Member Functions

- [Object](#) (SDL_Renderer *&r)
- [~Object](#) ()
- const SDL_Rect * [getCrop](#) ()
- void [cropping](#) (int x, int y, int w, int h)
- void [cropping](#) (SDL_Rect c)
- void [cropping](#) (const [json](#) &mem)
- void [noCropping](#) ()
- const SDL_Rect * [getLocation](#) ()
- int [getX](#) ()
- int [getY](#) ()
- int [getW](#) ()
- int [getH](#) ()
- virtual void [locating](#) (int x, int y, int w, int h)
- virtual void [locating](#) (SDL_Rect l)
- virtual void [locating](#) (const [json](#) &mem)
- virtual void [locatingX](#) (int x)
- virtual void [locatingY](#) (int y)
- virtual void [locatingW](#) (int w)
- virtual void [locatingH](#) (int h)
- virtual void [moveX](#) (int delta)
- virtual void [moveY](#) (int delta)
- virtual void [zoomW](#) (int delta)
- virtual void [zoomH](#) (int delta)
- virtual void [zoom](#) (double delta)
- virtual void [zoomInMiddle](#) (double delta)
- void [fitTheTexture](#) ()
- const SDL_Color * [getColor](#) ()
- void [coloring](#) (int r, int g, int b, int a)
- void [coloring](#) (SDL_Color c)
- void [coloring](#) (const [json](#) &mem)
- void [textureFromFile](#) (std::string dir)
- void [changeToCircle](#) ()
- void [changeToCircle](#) (SDL_Point c)
- void [changeToCircle](#) (int x, int y)
- void [changeToCircle](#) (SDL_Point c, int r)
- void [changeToCircle](#) (int x, int y, int r)
- void [changeToRectangle](#) ()
- void [setShape](#) (const [json](#) &mem)
- bool [isLiInside](#) (int x, int y)
- bool [isLiInside](#) (SDL_Point p)
- bool [isLiInside](#) (SDL_Rect r)
- bool [isLiInside](#) (int x, int y, int w, int h)
- virtual void [rendering](#) ()
- void [show](#) ()
- void [hide](#) ()
- bool [isVisible](#) ()
- void [importFromJson](#) (const [json](#) &mem)
- void [linking](#) (std::string n)
- void [setFont](#) (TTF_Font *f)
- void [setText](#) (std::string t)
- void [addText](#) (std::string t)
- void [addCharacter](#) (char c)
- void [removeCharacter](#) ()
- void [removeCharacter](#) (int n)
- std::string [getText](#) ()
- int [getSize](#) ()

Protected Member Functions

- void `fillWithColor` ()
- void `fillCircleByColor` ()
- void `fillRectangleByColor` ()
- void `textToTexture` ()

7.14.1 Detailed Description

Class that represent shape, image from files, text. Smallest drawable unit.

Definition at line 28 of file `object.hpp`.

7.14.2 Constructor & Destructor Documentation

7.14.2.1 Object()

```
Object::Object (
    SDL_Renderer *& r )
```

Definition at line 7 of file `constructor.cpp`.

```
8 {
9     render = r;
10    location = nullptr;
11    crop = nullptr;
12    color = nullptr;
13    texture = nullptr;
14    font = nullptr;
15    locating(0, 0, 0, 0);
16    shapeType = SHAPE::NONE;
17    visible = false;
18 }
```

7.14.2.2 ~Object()

```
Object::~~Object ( )
```

Definition at line 3 of file `destructor.cpp`.

```
4 {
5     render = nullptr;
6     if(crop != nullptr) delete crop;
7     if(location != nullptr) delete location;
8     if(color != nullptr) delete color;
9     if(texture != nullptr) SDL_DestroyTexture(texture);
10    texture = nullptr;
11
12    crop = nullptr;
13    location = nullptr;
14    color = nullptr;
15    texture = nullptr;
16 }
```


7.14.3 Member Function Documentation

7.14.3.1 addCharacter()

```
void Object::addCharacter (
    char c )
```

Definition at line 22 of file font.cpp.

```
23 {
24     text += c;
25     textToTexture();
26 }
```

7.14.3.2 addText()

```
void Object::addText (
    std::string t )
```

Definition at line 16 of file font.cpp.

```
17 {
18     text += t;
19     textToTexture();
20 }
```

7.14.3.3 changeToCircle() [1/5]

```
void Object::changeToCircle ( )
```

Definition at line 5 of file shape.cpp.

```
6 {
7     shapeType = SHAPE::CIRCLE;
8     radius = std::min(getW(), getH()) / 2;
9
10    center.x = getX() + getW() / 2;
11    center.y = getY() + getH() / 2;
12    fillCircleByColor();
13 }
```

7.14.3.4 changeToCircle() [2/5]

```
void Object::changeToCircle (
    int x,
    int y )
```

Definition at line 24 of file shape.cpp.

```
25 {
26     changeToCircle({x, y});
27 }
```

7.14.3.5 changeToCircle() [3/5]

```
void Object::changeToCircle (
    int x,
    int y,
    int r )
```

Definition at line 37 of file shape.cpp.

```
38 {
39     shapeType = SHAPE::CIRCLE;
40     radius = r;
41     center.x = x;
42     center.y = y;
43     fillCircleByColor();
44 }
```

7.14.3.6 changeToCircle() [4/5]

```
void Object::changeToCircle (
    SDL_Point c )
```

Definition at line 15 of file shape.cpp.

```
16 {
17     shapeType = SHAPE::CIRCLE;
18     center = c;
19     radius = std::min(getW() - c.x, c.x - getX());
20     radius = std::min(radius, std::min(getH() - c.y, c.y - getY()));
21     fillCircleByColor();
22 }
```

7.14.3.7 changeToCircle() [5/5]

```
void Object::changeToCircle (
    SDL_Point c,
    int r )
```

Definition at line 29 of file shape.cpp.

```
30 {
31     shapeType = SHAPE::CIRCLE;
32     radius = r;
33     center = c;
34     fillCircleByColor();
35 }
```

7.14.3.8 changeToRectangle()

```
void Object::changeToRectangle ( )
```

Definition at line 46 of file shape.cpp.

```
47 {
48     shapeType = SHAPE::RECTANGLE;
49     fillRectangleByColor();
50 }
```

7.14.3.9 coloring() [1/3]

```
void Object::coloring (
    const json & mem )
```

Definition at line 30 of file coloring.cpp.

```
31 {
32     if(mem.contains("r") && mem.contains("g") && mem.contains("b"))
33     {
34         if(mem.contains("a")) coloring(mem["r"], mem["g"], mem["b"], mem["a"]);
35         else coloring(mem["r"], mem["g"], mem["b"], 255);
36     }
37 }
```

7.14.3.10 coloring() [2/3]

```
void Object::coloring (
    int r,
    int g,
    int b,
    int a )
```

Definition at line 8 of file coloring.cpp.

```
9 {
10     if(color == nullptr) color = new SDL_Color;
11     color->r = r;
12     color->g = g;
13     color->b = b;
14     color->a = a;
15
16     fillWithColor();
17 }
```

7.14.3.11 coloring() [3/3]

```
void Object::coloring (
    SDL_Color c )
```

Definition at line 19 of file coloring.cpp.

```
20 {
21     if(color == nullptr) color = new SDL_Color;
22     color->r = c.r;
23     color->g = c.g;
24     color->b = c.b;
25     color->a = c.a;
26
27     fillWithColor();
28 }
```

7.14.3.12 cropping() [1/3]

```
void Object::cropping (
    const json & mem )
```

Definition at line 26 of file cropping.cpp.

```
27 {
28     if(mem.contains("x") && mem.contains("y") && mem.contains("w") && mem.contains("h"))
29         cropping(mem["x"], mem["y"], mem["w"], mem["h"]);
30 }
```

7.14.3.13 cropping() [2/3]

```
void Object::cropping (
    int x,
    int y,
    int w,
    int h )
```

Definition at line 8 of file cropping.cpp.

```
9 {
10     if(crop == nullptr) crop = new SDL_Rect;
11     crop->x = x;
12     crop->y = y;
13     crop->w = w;
14     crop->h = h;
15 }
```

7.14.3.14 cropping() [3/3]

```
void Object::cropping (
    SDL_Rect c )
```

Definition at line 17 of file cropping.cpp.

```
18 {
19     if(crop == nullptr) crop = new SDL_Rect;
20     crop->x = c.x;
21     crop->y = c.y;
22     crop->w = c.w;
23     crop->h = c.h;
24 }
```

7.14.3.15 fillCircleByColor()

```
void Object::fillCircleByColor ( ) [protected]
```

Definition at line 91 of file shape.cpp.

```
92 {
93     if(location == nullptr) locating(0, 0, 0, 0);
94
95     if(texture != nullptr) SDL_DestroyTexture(texture);
96     texture = nullptr;
97
98     Uint32 rmask, gmask, bmask, amask;
99     Uint32 pixelColor;
100     #if SDL_BYTEORDER == SDL_BIG_ENDIAN
101         rmask = 0xff000000;
102         gmask = 0x00ff0000;
103         bmask = 0x0000ff00;
104         amask = 0x000000ff;
105         pixelColor = (color->r << 24) | (color->g << 16) | (color->b << 8) | color->a;
106     #else
107         rmask = 0x000000ff;
108         gmask = 0x0000ff00;
109         bmask = 0x00ff0000;
110         amask = 0xff000000;
111         pixelColor = (color->a << 24) | (color->b << 16) | (color->g << 8) | color->r;
112     #endif
113
114     SDL_Surface *surf = SDL_CreateRGBSurface(0, getW(), getH(), 32, rmask, gmask, bmask, amask);
115     SDL_SetSurfaceBlendMode(surf, SDL_BLENDMODE_BLEND);
116
117     texture = SDL_CreateTextureFromSurface(render, surf);
118     SDL_FreeSurface(surf);
119
120     Uint32 *pixels = new Uint32[getW() * getH()];
```

```

121     memset(pixels, 0, getWidth() * getHeight() * sizeof(Uint32));
122
123     SDL_Point p = {getWidth() / 2, getHeight() / 2};
124     center = p;
125
126     if(radius > std::min(getWidth(), getHeight()) / 2) radius = std::min(getWidth(), getHeight()) / 2;
127
128     for(int i = p.x - radius; i <= p.x + radius; i++)
129         for(int j = p.y - radius; j <= p.y + radius; j++)
130             if((i - p.x) * (i - p.x) + (j - p.y) * (j - p.y) <= radius * radius)
131             {
132                 int index = i * getWidth() + j;
133                 if(index < 0 || index >= getWidth() * getHeight()) continue;
134                 pixels[index] = pixelColor;
135             }
136
137     SDL_UpdateTexture(texture, nullptr, pixels, getWidth() * sizeof(Uint32));
138     delete[] pixels;
139 }

```

7.14.3.16 fillRectangleByColor()

void Object::fillRectangleByColor () [protected]

Definition at line 74 of file shape.cpp.

```

75 {
76     if(location == nullptr) locating(0, 0, 0, 0);
77
78     if(texture != nullptr) SDL_DestroyTexture(texture);
79     texture = nullptr;
80
81     SDL_Surface* surf = SDL_CreateRGBSurfaceWithFormat(0, getWidth(), getHeight(), 32, SDL_PIXELFORMAT_RGBA32);
82
83     SDL_SetSurfaceBlendMode(surf, SDL_BLENDMODE_BLEND);
84
85     SDL_FillRect(surf, nullptr, SDL_MapRGBA(surf->format, color->r, color->g, color->b, color->a));
86
87     texture = SDL_CreateTextureFromSurface(render, surf);
88     SDL_FreeSurface(surf);
89 }

```

7.14.3.17 fillWithColor()

void Object::fillWithColor () [protected]

Definition at line 39 of file coloring.cpp.

```

40 {
41     if(shapeType == SHAPE::NONE) return fillRectangleByColor();
42     if(shapeType == SHAPE::RECTANGLE) return fillRectangleByColor();
43     if(shapeType == SHAPE::CIRCLE) return fillCircleByColor();
44 }

```

7.14.3.18 fitTheTexture()

void Object::fitTheTexture ()

Definition at line 140 of file locating.cpp.

```

141 {
142     if(texture == nullptr) return;
143     SDL_QueryTexture(texture, nullptr, nullptr, &location->w, &location->h);
144 }

```

7.14.3.19 getColor()

```
const SDL_Color * Object::getColor ( )
```

Definition at line 3 of file coloring.cpp.

```
4 {  
5     return color;  
6 }
```

7.14.3.20 getCrop()

```
const SDL_Rect * Object::getCrop ( )
```

Definition at line 3 of file cropping.cpp.

```
4 {  
5     return crop;  
6 }
```

7.14.3.21 getH()

```
int Object::getH ( )
```

Definition at line 47 of file locating.cpp.

```
48 {  
49     return location->h;  
50 }
```

7.14.3.22 getLocation()

```
const SDL_Rect * Object::getLocation ( )
```

Definition at line 27 of file locating.cpp.

```
28 {  
29     return location;  
30 }
```

7.14.3.23 getSize()

```
int Object::getSize ( )
```

Definition at line 68 of file font.cpp.

```
69 {  
70     return text.size();  
71 }
```

7.14.3.24 getText()

```
std::string Object::getText ( )
```

Definition at line 63 of file font.cpp.

```
64 {  
65     return text;  
66 }
```

7.14.3.25 getW()

```
int Object::getW ( )
```

Definition at line 42 of file locating.cpp.

```
43 {  
44     return location->w;  
45 }
```

7.14.3.26 getX()

```
int Object::getX ( )
```

Definition at line 32 of file locating.cpp.

```
33 {  
34     return location->x;  
35 }
```

7.14.3.27 getY()

```
int Object::getY ( )
```

Definition at line 37 of file locating.cpp.

```
38 {  
39     return location->y;  
40 }
```

7.14.3.28 hide()

```
void Object::hide ( )
```

Definition at line 8 of file visible.cpp.

```
9 {  
10     visible = false;  
11 }
```

7.14.3.29 importFromJson()

```
void Object::importFromJson (
    const json & mem )
```

Definition at line 21 of file constructor.cpp.

```
22 {
23     if(mem.contains("location"))
24         locating(mem["location"]);
25
26     if(mem.contains("crop"))
27         cropping(mem["crop"]);
28
29     if(mem.contains("color"))
30         coloring(mem["color"]);
31
32     if(mem.contains("shape"))
33         setShape(mem["shape"]);
34
35     if(mem.contains("visible"))
36         visible = mem["visible"];
37
38     if(mem.contains("image"))
39         textureFromFile(PATH::ASSETS::GRAPHICS_ + mem["image"].get<std::string>());
40     return ;
41 }
```

7.14.3.30 isLieInside() [1/4]

```
bool Object::isLieInside (
    int x,
    int y )
```

Definition at line 3 of file locating.cpp.

```
4 {
5     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
6     return (x >= location->x && x < location->x + location->w && y >= location->y && y < location->y +
    location->h);
7 }
```

7.14.3.31 isLieInside() [2/4]

```
bool Object::isLieInside (
    int x,
    int y,
    int w,
    int h )
```

Definition at line 21 of file locating.cpp.

```
22 {
23     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
24     return (x >= location->x && x + w <= location->x + location->w && y >= location->y && y + h <=
    location->y + location->h);
25 }
```


7.14.3.32 isLieInside() [3/4]

```
bool Object::isLieInside (
    SDL_Point p )
```

Definition at line 9 of file locating.cpp.

```
10 {
11     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
12     return (p.x >= location->x && p.x < location->x + location->w && p.y >= location->y && p.y <
        location->y + location->h);
13 }
```

7.14.3.33 isLieInside() [4/4]

```
bool Object::isLieInside (
    SDL_Rect r )
```

Definition at line 15 of file locating.cpp.

```
16 {
17     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
18     return (r.x >= location->x && r.x + r.w <= location->x + location->w && r.y >= location->y && r.y +
        r.h <= location->y + location->h);
19 }
```

7.14.3.34 isVisible()

```
bool Object::isVisible ( )
```

Definition at line 13 of file visible.cpp.

```
14 {
15     return visible;
16 }
```

7.14.3.35 linking()

```
void Object::linking (
    std::string n )
```

Definition at line 43 of file constructor.cpp.

```
44 {
45     name = n;
46     json* mem = JSON::readFile(PATH::ATB::OBJECT_ + name + ".json");
47
48     importFromJson(*mem);
49
50     delete mem;
51 }
```

7.14.3.36 locating() [1/3]

```
void Object::locating (
    const json & mem ) [virtual]
```

Definition at line 70 of file locating.cpp.

```
71 {
72     if(mem.contains("x") && mem.contains("y") && mem.contains("w") && mem.contains("h"))
73         locating(mem["x"], mem["y"], mem["w"], mem["h"]);
74 }
```

7.14.3.37 locating() [2/3]

```
void Object::locating (
    int x,
    int y,
    int w,
    int h ) [virtual]
```

Reimplemented in [Sprite](#).

Definition at line 52 of file locating.cpp.

```
53 {
54     if(location == nullptr) location = new SDL_Rect;
55     location->x = x;
56     location->y = y;
57     location->w = w;
58     location->h = h;
59 }
```

7.14.3.38 locating() [3/3]

```
void Object::locating (
    SDL_Rect l ) [virtual]
```

Reimplemented in [Sprite](#).

Definition at line 61 of file locating.cpp.

```
62 {
63     if(location == nullptr) location = new SDL_Rect;
64     location->x = l.x;
65     location->y = l.y;
66     location->w = l.w;
67     location->h = l.h;
68 }
```

7.14.3.39 locatingH()

```
void Object::locatingH (
    int h ) [virtual]
```

Definition at line 94 of file locating.cpp.

```
95 {
96     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
97     location->h = h;
98 }
```

7.14.3.40 locatingW()

```
void Object::locatingW (
    int w ) [virtual]
```

Definition at line 88 of file locating.cpp.

```
89 {
90     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
91     location->w = w;
92 }
```

7.14.3.41 locatingX()

```
void Object::locatingX (
    int x ) [virtual]
```

Reimplemented in [Sprite](#).

Definition at line 76 of file locating.cpp.

```
77 {
78     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
79     location->x = x;
80 }
```

7.14.3.42 locatingY()

```
void Object::locatingY (
    int y ) [virtual]
```

Reimplemented in [Sprite](#).

Definition at line 82 of file locating.cpp.

```
83 {
84     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
85     location->y = y;
86 }
```

7.14.3.43 moveX()

```
void Object::moveX (
    int delta ) [virtual]
```

Reimplemented in [Sprite](#).

Definition at line 100 of file locating.cpp.

```
101 {
102     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
103     location->x += delta;
104 }
```

7.14.3.44 moveY()

```
void Object::moveY (
    int delta ) [virtual]
```

Reimplemented in [Sprite](#).

Definition at line 106 of file locating.cpp.

```
107 {
108     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
109     location->y += delta;
110 }
```

7.14.3.45 noCropping()

```
void Object::noCropping ( )
```

Definition at line 32 of file cropping.cpp.

```
33 {
34     if(crop != nullptr) delete crop;
35     crop = nullptr;
36 }
```

7.14.3.46 removeCharacter() [1/2]

```
void Object::removeCharacter ( )
```

Definition at line 28 of file font.cpp.

```
29 {
30     if (text.size() > 0)
31         text.pop_back();
32     textToTexture();
33 }
```

7.14.3.47 removeCharacter() [2/2]

```
void Object::removeCharacter (
    int n )
```

Definition at line 35 of file font.cpp.

```
36 {
37     if(n == 0) return ;
38     if(text.size() <= n) text.clear();
39     else text.erase(text.end() - n, text.end());
40     textToTexture();
41 }
```

7.14.3.48 rendering()

```
void Object::rendering ( ) [virtual]
```

Reimplemented in [Sprite](#), [Script](#), [InputBox](#), [Display](#), [DataStructures](#), and [Button](#).

Definition at line 3 of file `rendering.cpp`.

```
4 {
5     if(!visible) return ;
6     if(texture == nullptr) return ;
7     SDL_RenderCopy(render, texture, crop, location);
8 }
```

7.14.3.49 setFont()

```
void Object::setFont (
    TTF_Font * f )
```

Definition at line 4 of file `font.cpp`.

```
5 {
6     font = f;
7     textToTexture();
8 }
```

7.14.3.50 setShape()

```
void Object::setShape (
    const json & mem )
```

Definition at line 52 of file `shape.cpp`.

```
53 {
54     if(mem["type"].get<std::string>() == "CIRCLE")
55     {
56         if(mem.contains("center"))
57         {
58             if(mem.contains("radius"))
59                 changeToCircle(mem["center"]["x"], mem["center"]["y"], mem["radius"]);
60             else changeToCircle(mem["center"]["x"], mem["center"]["y"]);
61         }else changeToCircle();
62     }
63     return ;
64 }
65
66 if(mem["type"].get<std::string>() == "NONE" || mem["type"].get<std::string>() == "RECTANGLE")
67 {
68     changeToRectangle();
69     return ;
70 }
71
72 }
```

7.14.3.51 setText()

```
void Object::setText (
    std::string t )
```

Definition at line 10 of file `font.cpp`.

```
11 {
12     text = t;
13     textToTexture();
14 }
```

7.14.3.52 show()

```
void Object::show ( )
```

Definition at line 3 of file visible.cpp.

```
4 {  
5     visible = true;  
6 }
```

7.14.3.53 textToTexture()

```
void Object::textToTexture ( ) [protected]
```

Definition at line 43 of file font.cpp.

```
44 {  
45     if(font == nullptr) return ;  
46     if(color == nullptr) return ;  
47     if(render == nullptr) return ;  
48     if(texture != nullptr)  
49     {  
50         SDL_DestroyTexture(texture);  
51     }  
52     texture = nullptr;  
53  
54     SDL_Surface* surface = TTF_RenderText_Blended(font, text.c_str(), *color);  
55  
56     if(surface == nullptr) return ;  
57  
58     texture = SDL_CreateTextureFromSurface(render, surface);  
59     SDL_FreeSurface(surface);  
60     fitTheTexture();  
61 }
```

7.14.3.54 textureFromFile()

```
void Object::textureFromFile (   
    std::string dir )
```

Definition at line 4 of file external_storage.cpp.

```
5 {  
6     SDL_Surface *surface = IMG_Load(dir.c_str());  
7  
8     texture = SDL_CreateTextureFromSurface(render, surface);  
9     SDL_FreeSurface(surface);  
10 }
```

7.14.3.55 zoom()

```
void Object::zoom (   
    double delta ) [virtual]
```

Definition at line 123 of file locating.cpp.

```
124 {  
125     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});  
126     location->w *= delta;  
127     location->h *= delta;  
128 }
```

7.14.3.56 zoomH()

```
void Object::zoomH (
    int delta ) [virtual]
```

Definition at line 118 of file locating.cpp.

```
119 {
120     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
121     location->h += delta;
122 }
```

7.14.3.57 zoomInMiddle()

```
void Object::zoomInMiddle (
    double delta ) [virtual]
```

Definition at line 130 of file locating.cpp.

```
131 {
132     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
133     SDL_Point center = {location->x + location->w / 2, location->y + location->h / 2};
134     location->w *= delta;
135     location->h *= delta;
136     location->x = center.x - location->w / 2;
137     location->y = center.y - location->h / 2;
138 }
```

7.14.3.58 zoomW()

```
void Object::zoomW (
    int delta ) [virtual]
```

Definition at line 112 of file locating.cpp.

```
113 {
114     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
115     location->w += delta;
116 }
```

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

- include/object.hpp
- src/object/coloring.cpp
- src/object/constructor.cpp
- src/object/cropping.cpp
- src/object/destructor.cpp
- src/object/external_storage.cpp
- src/object/font.cpp
- src/object/locating.cpp
- src/object/rendering.cpp
- src/object/shape.cpp
- src/object/visible.cpp

7.15 Point Struct Reference

Public Attributes

- double [x](#)
- double [y](#)

7.15.1 Detailed Description

Definition at line 3 of file `rendering.cpp`.

7.15.2 Member Data Documentation

7.15.2.1 [x](#)

```
double Point::x
```

Definition at line 5 of file `rendering.cpp`.

7.15.2.2 [y](#)

```
double Point::y
```

Definition at line 5 of file `rendering.cpp`.

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

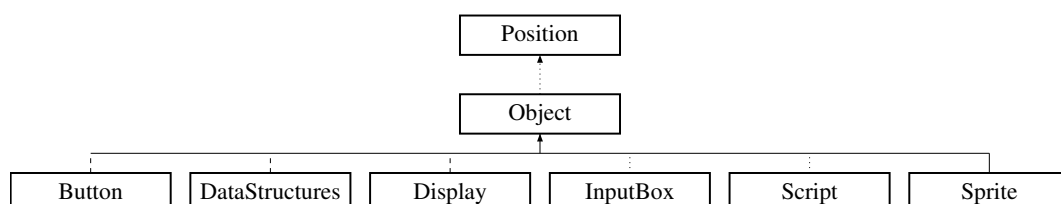
- [src/graph/rendering.cpp](#)

7.16 Position Struct Reference

location of an object in 2D coordinate

```
#include <object.hpp>
```

Inheritance diagram for Position:



Public Attributes

- SDL_Rect * [location](#)
- SDL_Rect * [crop](#)
- int [radius](#)
- SDL_Point [center](#)
- bool [visible](#)

7.16.1 Detailed Description

location of an object in 2D coordinate

Definition at line 16 of file object.hpp.

7.16.2 Member Data Documentation

7.16.2.1 center

```
SDL_Point Position::center
```

Definition at line 21 of file object.hpp.

7.16.2.2 crop

```
SDL_Rect* Position::crop
```

Definition at line 19 of file object.hpp.

7.16.2.3 location

```
SDL_Rect* Position::location
```

Definition at line 18 of file object.hpp.

7.16.2.4 radius

```
int Position::radius
```

Definition at line 20 of file object.hpp.

7.16.2.5 visible

```
bool Position::visible
```

Definition at line 22 of file object.hpp.

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

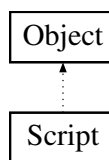
- include/object.hpp

7.17 Script Class Reference

Container that contains a pseudo-code.

```
#include <script.hpp>
```

Inheritance diagram for Script:



Public Member Functions

- [Script](#) (SDL_Renderer *render, TTF_Font *f)
- [~Script](#) ()
- void [linking](#) (std::string name)
- void [highlight](#) (int index)
- void [unhighlight](#) (int index)
- bool [isReceiveEvent](#) (SDL_Event &event)
- [Button](#) * [react](#) (SDL_Event &event)
- void [rendering](#) ()

Protected Member Functions

- void [initBackground](#) (const [json](#) &mem)
- void [initButtons](#) (const [json](#) &mem)
- void [importFromJson](#) ()

Private Member Functions

- void [importFromJson](#) (const [json](#) &mem)

7.17.1 Detailed Description

Container that contains a pseudo-code.

Drawable

Definition at line 21 of file script.hpp.

7.17.2 Constructor & Destructor Documentation

7.17.2.1 Script()

```
Script::Script (
    SDL_Renderer * render,
    TTF_Font * f )
```

Definition at line 4 of file constructor.cpp.

```
4                                     : Object(render)
5 {
6     this->render = render;
7     this->font = f;
8     isMoving = false;
9 }
```

7.17.2.2 ~Script()

```
Script::~~Script ( )
```

Definition at line 3 of file destructor.cpp.

```
4 {
5     render = nullptr;
6     font = nullptr;
7     for(int i = 0; i < sprites.size(); i++)
8         if(sprites[i] != nullptr) delete sprites[i];
9     sprites.clear();
10 }
```

7.17.3 Member Function Documentation

7.17.3.1 highlight()

```
void Script::highlight (
    int index )
```

Definition at line 3 of file highlight.cpp.

```
4 {
5     if(index < 0 || index >= sprites.size()) return ;
6     sprites[index]->setTextBoxTransparent(180);
7 }
```

7.17.3.2 importFromJson()

```
void Script::importFromJson ( ) [protected]
```

Definition at line 17 of file constructor.cpp.

```
18 {
19     json * mem = JSON::readFile(PATH::ATB::SCRIPT_ + name + ".json");
20     if(mem == nullptr) return;
21
22     if(mem->contains("background"))
23         initBackground((*mem)["background"]);
24     if(mem->contains("sprite"))
25         spriteName = (*mem)["sprite"].get<std::string>();
26     if(mem->contains("buttons"))
27         initButtons((*mem)["buttons"]);
28
29     if(mem->contains("script"))
30     {
31         auto lines = FILEE::readFile(PATH::ASSETS::SCRIPT_ + (*mem)["script"].get<std::string>());
32         int j = 0;
33         for(auto i : lines)
34         {
35             sprites.push_back(new Sprite(render));
36             sprites.back()->setFont(font);
37             sprites.back()->linking(spriteName);
38             sprites.back()->setText(i);
39             sprites.back()->locatingX(getX());
40             sprites.back()->locatingY(getY() + j * 18);
41             sprites.back()->aligning(HORIZONTAL_ALIGN::LEFT, VERTICAL_ALIGN::CENTER);
42             j++;
43         }
44     }
45 }
46 }
```

7.17.3.3 initBackground()

```
void Script::initBackground (
    const json & mem ) [protected]
```

Definition at line 48 of file constructor.cpp.

```
49 {
50     Object::importFromJson(mem);
51 }
```

7.17.3.4 initButtons()

```
void Script::initButtons (
    const json & mem ) [protected]
```

Definition at line 53 of file constructor.cpp.

```
54 {
55     for(auto &i : mem)
56     {
57         Button* b = new Button(render);
58         b->linking(i["name"].get<std::string>());
59         buts.push_back(b);
60     }
61 }
62 }
```

7.17.3.5 isReceiveEvent()

```
bool Script::isReceiveEvent (
    SDL_Event & event )
```

Definition at line 4 of file event.cpp.

```
5 {
6     SDL_Rect viewport = {getX(), getY(), getW(), getH()};
7
8     switch(e.type)
9     {
10         case SDL_MOUSEBUTTONDOWN:
11             if(e.motion.x < viewport.x || viewport.x + viewport.w < e.motion.x) return false;
12             if(e.motion.y < viewport.y || viewport.y + viewport.h < e.motion.y) return false;
13             if(e.button.button == SDL_BUTTON_LEFT)
14             {
15                 for(auto i : buts)
16                 {
17                     if(i->isReceiveEvent(e))
18                     {
19                         return true;
20                     }
21                 }
22                 return false;
23             }
24             return true;
25             break;
26         case SDL_MOUSEMOTION:
27             if(isMoving) return true;
28             for(auto i : buts)
29             {
30                 if(i->isReceiveEvent(e))
31                 {
32                     return true;
33                 }
34             }
35             return false;
36             break;
37         default:
38             return false;
39             break;
40     }
41 }
```

7.17.3.6 linking()

```
void Script::linking (
    std::string name )
```

Definition at line 11 of file constructor.cpp.

```
12 {
13     this->name = name;
14     importFromJson();
15 }
```

7.17.3.7 react()

```
Button * Script::react (
    SDL_Event & event )
```

Definition at line 43 of file event.cpp.

```
44 {
45     switch(e.type)
46     {
47         case SDL_MOUSEBUTTONDOWN:
48             if(isMoving)
```

```

49         {
50             isMoving = false;
51             int dx = e.motion.x - lastMousePressed.x;
52             int dy = e.motion.y - lastMousePressed.y;
53             moveX(dx);
54             moveY(dy);
55             for(auto i : buts)
56                 i->move(dx, dy);
57             for(auto i : sprites)
58             {
59                 i->moveX(dx);
60                 i->moveY(dy);
61             }
62         }
63     }else
64     {
65         for(auto i : buts)
66         {
67             if(i->isReceiveEvent(e))
68             {
69                 return i;
70             }
71         }
72         isMoving = true;
73         lastMousePressed.x = e.motion.x;
74         lastMousePressed.y = e.motion.y;
75     }
76     return nullptr;
77     break;
78 case SDL_MOUSEMOTION:
79     {
80         if(!isMoving)
81         {
82             for(auto i : buts)
83                 i->isHover(e.motion.x, e.motion.y);
84             return nullptr;
85         }
86         int dx = e.motion.x - lastMousePressed.x;
87         int dy = e.motion.y - lastMousePressed.y;
88         lastMousePressed.x = e.motion.x;
89         lastMousePressed.y = e.motion.y;
90         moveX(dx);
91         moveY(dy);
92         for(auto i : buts)
93             i->move(dx, dy);
94         for(auto i : sprites)
95         {
96             i->moveX(dx);
97             i->moveY(dy);
98         }
99     }
100     return nullptr;
101     break;
102 }
103 default:
104     return nullptr;
105     break;
106 }
107 }
108 }

```

7.17.3.8 rendering()

void Script::rendering () [virtual]

Reimplemented from [Object](#).

Definition at line 3 of file rendering.cpp.

```

4 {
5     Object::rendering();
6     for(auto& i : sprites)
7         i->rendering();
8     for(auto& i : buts)
9         i->rendering();
10    return ;
11 }

```

7.17.3.9 unhighlight()

```
void Script::unhighlight (
    int index )
```

Definition at line 9 of file highlight.cpp.

```
10 {
11     if(index < 0 || index >= sprites.size()) return ;
12     sprites[index]->setTextBoxTransparent(0);
13 }
```

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

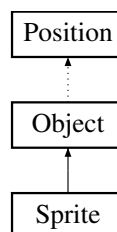
- include/script.hpp
- src/script/constructor.cpp
- src/script/destructor.cpp
- src/script/event.cpp
- src/script/highlight.cpp
- src/script/rendering.cpp

7.18 Sprite Class Reference

[Object](#) container [Drawable](#).

```
#include <sprite.hpp>
```

Inheritance diagram for Sprite:



Public Member Functions

- [Sprite](#) (SDL_Renderer *r)
- [~Sprite](#) ()
- void [linking](#) (std::string n)
- void [setFont](#) (TTF_Font *f)
- void [setFontColor](#) (SDL_Color c)
- void [setTextBox](#) (TTF_Font *f)
- void [setTextBoxTransparent](#) (int a)
- void [setText](#) (std::string s)
- void [typing](#) (char c)
- void [typing](#) (std::string s)
- void [backspace](#) ()
- void [backspace](#) (int n)
- std::string [getText](#) ()
- void [locating](#) (int x, int y, int w, int h) override

- void [locating](#) (SDL_Rect r) override
- void [locatingX](#) (int x) override
- void [locatingY](#) (int y) override
- void [moveX](#) (int x) override
- void [moveY](#) (int y) override
- void [aligning](#) (HORIZONTAL_ALIGN h, VERTICAL_ALIGN v)
- void [aligning](#) (HORIZONTAL_ALIGN h)
- void [aligning](#) (VERTICAL_ALIGN v)
- void [rendering](#) () override
- void [highlight](#) ()
- void [unhighlight](#) ()
- const SDL_Rect * [getCrop](#) ()
- void [cropping](#) (int x, int y, int w, int h)
- void [cropping](#) (SDL_Rect c)
- void [cropping](#) (const json &mem)
- void [noCropping](#) ()
- const SDL_Rect * [getLocation](#) ()
- int [getX](#) ()
- int [getY](#) ()
- int [getW](#) ()
- int [getH](#) ()
- virtual void [locating](#) (const json &mem)
- virtual void [locatingW](#) (int w)
- virtual void [locatingH](#) (int h)
- virtual void [zoomW](#) (int delta)
- virtual void [zoomH](#) (int delta)
- virtual void [zoom](#) (double delta)
- virtual void [zoomInMiddle](#) (double delta)
- void [fitTheTexture](#) ()
- const SDL_Color * [getColor](#) ()
- void [coloring](#) (int r, int g, int b, int a)
- void [coloring](#) (SDL_Color c)
- void [coloring](#) (const json &mem)
- void [textureFromFile](#) (std::string dir)
- void [changeToCircle](#) ()
- void [changeToCircle](#) (SDL_Point c)
- void [changeToCircle](#) (int x, int y)
- void [changeToCircle](#) (SDL_Point c, int r)
- void [changeToCircle](#) (int x, int y, int r)
- void [changeToRectangle](#) ()
- void [setShape](#) (const json &mem)
- bool [isLieInside](#) (int x, int y)
- bool [isLieInside](#) (SDL_Point p)
- bool [isLieInside](#) (SDL_Rect r)
- bool [isLieInside](#) (int x, int y, int w, int h)
- void [show](#) ()
- void [hide](#) ()
- bool [isVisible](#) ()
- void [importFromJson](#) (const json &mem)
- void [addText](#) (std::string t)
- void [addCharacter](#) (char c)
- void [removeCharacter](#) ()
- void [removeCharacter](#) (int n)
- int [getSize](#) ()

Protected Member Functions

- void `initBackground` (const `json` &mem)
- void `initObjects` (const `json` &mem)
- void `initTextBox` (const `json` &mem)
- void `initInput` (const `json` &mem)
- void `importFromJson` ()
- void `aligning` ()
- void `fillWithColor` ()
- void `fillCircleByColor` ()
- void `fillRectangleByColor` ()
- void `textToTexture` ()

7.18.1 Detailed Description

`Object` container Drawable.

Definition at line 20 of file `sprite.hpp`.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 `Sprite()`

```
Sprite::Sprite (  
    SDL_Renderer * r )
```

Definition at line 9 of file `constructor.cpp`.

```
9             : Object(r)  
10 {  
11     render = r;  
12     textBox = nullptr;  
13     alignH = HORIZONTAL_ALIGN::CENTER;  
14     alignV = VERTICAL_ALIGN::CENTER;  
15     receiveDigit = true;  
16     receiveLetter = true;  
17     receiveSymbol = true;  
18     maxSize = 50;  
19 }
```

7.18.2.2 `~Sprite()`

```
Sprite::~~Sprite ( )
```

Definition at line 3 of file `destructor.cpp`.

```
4 {  
5     for (auto& object : objects)  
6     {  
7         delete object;  
8     }  
9 }
```

7.18.3 Member Function Documentation

7.18.3.1 addCharacter()

```
void Object::addCharacter (
    char c ) [inherited]
```

Definition at line 22 of file font.cpp.

```
23 {
24     text += c;
25     textToTexture();
26 }
```

7.18.3.2 addText()

```
void Object::addText (
    std::string t ) [inherited]
```

Definition at line 16 of file font.cpp.

```
17 {
18     text += t;
19     textToTexture();
20 }
```

7.18.3.3 aligning() [1/4]

```
void Sprite::aligning ( ) [protected]
```

Definition at line 4 of file aligning.cpp.

```
5 {
6     if(textBox == nullptr) return ;
7     int w = std::min(textBox->getW(), getW());
8     int h = std::min(textBox->getH(), getH());
9
10    textBox->cropping(textBox->getW() - w, textBox->getH() - h, w, h);
11    textBox->locatingW(w);
12    textBox->locatingH(h);
13    switch(alignH)
14    {
15        case HORIZONTAL_ALIGN::LEFT:
16            textBox->locatingX(getX());
17            break;
18        case HORIZONTAL_ALIGN::CENTER:
19            textBox->locatingX(getX() + (getW() - w) / 2);
20            break;
21        case HORIZONTAL_ALIGN::RIGHT:
22            textBox->locatingX(getX() + getW() - w);
23            break;
24    }
25    switch(alignV)
26    {
27        case VERTICAL_ALIGN::TOP:
28            textBox->locatingY(getY());
29            break;
30        case VERTICAL_ALIGN::CENTER:
31            textBox->locatingY(getY() + (getH() - h) / 2);
32            break;
33        case VERTICAL_ALIGN::BOTTOM:
34            textBox->locatingY(getY() + getH() - h);
35            break;
36    }
37 }
```

7.18.3.4 aligning() [2/4]

```
void Sprite::aligning (
    HORIZONTAL_ALIGN h )
```

Definition at line 46 of file aligning.cpp.

```
47 {
48     alignH = h;
49     aligning();
50 }
```

7.18.3.5 aligning() [3/4]

```
void Sprite::aligning (
    HORIZONTAL_ALIGN h,
    VERTICAL_ALIGN v )
```

Definition at line 39 of file aligning.cpp.

```
40 {
41     alignH = h;
42     alignV = v;
43     aligning();
44 }
```

7.18.3.6 aligning() [4/4]

```
void Sprite::aligning (
    VERTICAL_ALIGN v )
```

Definition at line 52 of file aligning.cpp.

```
53 {
54     alignV = v;
55     aligning();
56 }
```

7.18.3.7 backspace() [1/2]

```
void Sprite::backspace ( )
```

Definition at line 52 of file textbox.cpp.

```
53 {
54     if(textBox == nullptr) return ;
55     textBox->removeCharacter();
56     aligning();
57 }
```

7.18.3.8 backspace() [2/2]

```
void Sprite::backspace (
    int n )
```

Definition at line 59 of file textbox.cpp.

```
60 {
61     if(textBox == nullptr) return ;
62     textBox->removeCharacter(n);
63     aligning();
64 }
```

7.18.3.9 changeToCircle() [1/5]

```
void Object::changeToCircle ( ) [inherited]
```

Definition at line 5 of file shape.cpp.

```
6 {
7     shapeType = SHAPE::CIRCLE;
8     radius = std::min(getW(), getH()) / 2;
9
10    center.x = getX() + getW() / 2;
11    center.y = getY() + getH() / 2;
12    fillCircleByColor();
13 }
```

7.18.3.10 changeToCircle() [2/5]

```
void Object::changeToCircle (
    int x,
    int y ) [inherited]
```

Definition at line 24 of file shape.cpp.

```
25 {
26     changeToCircle({x, y});
27 }
```

7.18.3.11 changeToCircle() [3/5]

```
void Object::changeToCircle (
    int x,
    int y,
    int r ) [inherited]
```

Definition at line 37 of file shape.cpp.

```
38 {
39     shapeType = SHAPE::CIRCLE;
40     radius = r;
41     center.x = x;
42     center.y = y;
43     fillCircleByColor();
44 }
```

7.18.3.12 changeToCircle() [4/5]

```
void Object::changeToCircle (
    SDL_Point c ) [inherited]
```

Definition at line 15 of file shape.cpp.

```
16 {
17     shapeType = SHAPE::CIRCLE;
18     center = c;
19     radius = std::min(getW() - c.x, c.x - getX());
20     radius = std::min(radius, std::min(getH() - c.y, c.y - getY()));
21     fillCircleByColor();
22 }
```

7.18.3.13 changeToCircle() [5/5]

```
void Object::changeToCircle (
    SDL_Point c,
    int r ) [inherited]
```

Definition at line 29 of file shape.cpp.

```
30 {
31     shapeType = SHAPE::CIRCLE;
32     radius = r;
33     center = c;
34     fillCircleByColor();
35 }
```

7.18.3.14 changeToRectangle()

```
void Object::changeToRectangle ( ) [inherited]
```

Definition at line 46 of file shape.cpp.

```
47 {
48     shapeType = SHAPE::RECTANGLE;
49     fillRectangleByColor();
50 }
```

7.18.3.15 coloring() [1/3]

```
void Object::coloring (
    const json & mem ) [inherited]
```

Definition at line 30 of file coloring.cpp.

```
31 {
32     if(mem.contains("r") && mem.contains("g") && mem.contains("b"))
33     {
34         if(mem.contains("a")) coloring(mem["r"], mem["g"], mem["b"], mem["a"]);
35         else coloring(mem["r"], mem["g"], mem["b"], 255);
36     }
37 }
```

7.18.3.16 coloring() [2/3]

```
void Object::coloring (
    int r,
    int g,
    int b,
    int a ) [inherited]
```

Definition at line 8 of file coloring.cpp.

```
9 {
10     if(color == nullptr) color = new SDL_Color;
11     color->r = r;
12     color->g = g;
13     color->b = b;
14     color->a = a;
15
16     fillWithColor();
17 }
```

7.18.3.17 coloring() [3/3]

```
void Object::coloring (
    SDL_Color c ) [inherited]
```

Definition at line 19 of file coloring.cpp.

```
20 {
21     if(color == nullptr) color = new SDL_Color;
22     color->r = c.r;
23     color->g = c.g;
24     color->b = c.b;
25     color->a = c.a;
26
27     fillWithColor();
28 }
```

7.18.3.18 cropping() [1/3]

```
void Object::cropping (
    const json & mem ) [inherited]
```

Definition at line 26 of file cropping.cpp.

```
27 {
28     if(mem.contains("x") && mem.contains("y") && mem.contains("w") && mem.contains("h"))
29         cropping(mem["x"], mem["y"], mem["w"], mem["h"]);
30 }
```

7.18.3.19 cropping() [2/3]

```
void Object::cropping (
    int x,
    int y,
    int w,
    int h ) [inherited]
```

Definition at line 8 of file cropping.cpp.

```
9 {
10     if(crop == nullptr) crop = new SDL_Rect;
11     crop->x = x;
12     crop->y = y;
13     crop->w = w;
14     crop->h = h;
15 }
```

7.18.3.20 cropping() [3/3]

```
void Object::cropping (
    SDL_Rect c ) [inherited]
```

Definition at line 17 of file cropping.cpp.

```
18 {
19     if(crop == nullptr) crop = new SDL_Rect;
20     crop->x = c.x;
21     crop->y = c.y;
22     crop->w = c.w;
23     crop->h = c.h;
24 }
```

7.18.3.21 fillCircleByColor()

```
void Object::fillCircleByColor ( ) [protected], [inherited]
```

Definition at line 91 of file shape.cpp.

```
92 {
93     if(location == nullptr) locating(0, 0, 0, 0);
94
95     if(texture != nullptr) SDL_DestroyTexture(texture);
96     texture = nullptr;
97
98     Uint32 rmask, gmask, bmask, amask;
99     Uint32 pixelColor;
100     #if SDL_BYTEORDER == SDL_BIG_ENDIAN
101         rmask = 0xff000000;
102         gmask = 0x00ff0000;
103         bmask = 0x0000ff00;
104         amask = 0x000000ff;
105         pixelColor = (color->r << 24) | (color->g << 16) | (color->b << 8) | color->a;
106     #else
107         rmask = 0x000000ff;
108         gmask = 0x0000ff00;
109         bmask = 0x00ff0000;
110         amask = 0xff000000;
111         pixelColor = (color->a << 24) | (color->b << 16) | (color->g << 8) | color->r;
112     #endif
113
114     SDL_Surface *surf = SDL_CreateRGBSurface(0, getW(), getH(), 32, rmask, gmask, bmask, amask);
115     SDL_SetSurfaceBlendMode(surf, SDL_BLENDMODE_BLEND);
116
117     texture = SDL_CreateTextureFromSurface(render, surf);
118     SDL_FreeSurface(surf);
119
120     Uint32 *pixels = new Uint32[getW() * getH()];
121     memset(pixels, 0, getW() * getH() * sizeof(Uint32));
122
123     SDL_Point p = {getW() / 2, getH() / 2};
124     center = p;
125
126     if(radius > std::min(getW(), getH()) / 2) radius = std::min(getW(), getH()) / 2;
127
128     for(int i = p.x - radius; i <= p.x + radius; i++)
129         for(int j = p.y - radius; j <= p.y + radius; j++)
130             if((i - p.x) * (i - p.x) + (j - p.y) * (j - p.y) <= radius * radius)
131             {
132                 int index = i * getW() + j;
133                 if(index < 0 || index >= getW() * getH()) continue;
134                 pixels[index] = pixelColor;
135             }
136
137     SDL_UpdateTexture(texture, nullptr, pixels, getW() * sizeof(Uint32));
138     delete[] pixels;
139 }
```

7.18.3.22 fillRectangleByColor()

void Object::fillRectangleByColor () [protected], [inherited]

Definition at line 74 of file shape.cpp.

```

75 {
76     if(location == nullptr) locating(0, 0, 0, 0);
77
78     if(texture != nullptr) SDL_DestroyTexture(texture);
79     texture = nullptr;
80
81     SDL_Surface* surf = SDL_CreateRGBSurfaceWithFormat(0, getW(), getH(), 32, SDL_PIXELFORMAT_RGBA32);
82
83     SDL_SetSurfaceBlendMode(surf, SDL_BLENDMODE_BLEND);
84
85     SDL_FillRect(surf, nullptr, SDL_MapRGBA(surf->format, color->r, color->g, color->b, color->a));
86
87     texture = SDL_CreateTextureFromSurface(render, surf);
88
89     SDL_FreeSurface(surf);
90 }
```

7.18.3.23 fillWithColor()

void Object::fillWithColor () [protected], [inherited]

Definition at line 39 of file coloring.cpp.

```

40 {
41     if(shapeType == SHAPE::NONE) return fillRectangleByColor();
42     if(shapeType == SHAPE::RECTANGLE) return fillRectangleByColor();
43     if(shapeType == SHAPE::CIRCLE) return fillCircleByColor();
44 }
```

7.18.3.24 fitTheTexture()

void Object::fitTheTexture () [inherited]

Definition at line 140 of file locating.cpp.

```

141 {
142     if(texture == nullptr) return;
143     SDL_QueryTexture(texture, nullptr, nullptr, &location->w, &location->h);
144 }
```

7.18.3.25 getColor()

const SDL_Color * Object::getColor () [inherited]

Definition at line 3 of file coloring.cpp.

```

4 {
5     return color;
6 }
```


7.18.3.26 getCrop()

```
const SDL_Rect * Object::getCrop ( ) [inherited]
```

Definition at line 3 of file cropping.cpp.

```
4 {  
5     return crop;  
6 }
```

7.18.3.27 getH()

```
int Object::getH ( ) [inherited]
```

Definition at line 47 of file locating.cpp.

```
48 {  
49     return location->h;  
50 }
```

7.18.3.28 getLocation()

```
const SDL_Rect * Object::getLocation ( ) [inherited]
```

Definition at line 27 of file locating.cpp.

```
28 {  
29     return location;  
30 }
```

7.18.3.29 getSize()

```
int Object::getSize ( ) [inherited]
```

Definition at line 68 of file font.cpp.

```
69 {  
70     return text.size();  
71 }
```

7.18.3.30 getText()

```
std::string Sprite::getText ( )
```

Definition at line 49 of file constructor.cpp.

```
50 {  
51     if(textBox == nullptr) return "";  
52     return textBox->getText();  
53 }
```

7.18.3.31 getW()

int Object::getW () [inherited]

Definition at line 42 of file locating.cpp.

```
43 {  
44     return location->w;  
45 }
```

7.18.3.32 getX()

int Object::getX () [inherited]

Definition at line 32 of file locating.cpp.

```
33 {  
34     return location->x;  
35 }
```

7.18.3.33 getY()

int Object::getY () [inherited]

Definition at line 37 of file locating.cpp.

```
38 {  
39     return location->y;  
40 }
```

7.18.3.34 hide()

void Object::hide () [inherited]

Definition at line 8 of file visible.cpp.

```
9 {  
10     visible = false;  
11 }
```

7.18.3.35 highlight()

void Sprite::highlight ()

Definition at line 4 of file coloring.cpp.

```
5 {  
6     const SDL_Color* c = Object::getColor();  
7  
8     if(c == nullptr) return ;  
9  
10    cacheColor = {c->r, c->g, c->b, c->a};  
11    int maxChannel = std::max(std::max(c->r, c->g), c->b);  
12    Object::coloring(maxChannel * 0.4, maxChannel * 0.4, maxChannel * 0.4, c->a);  
13 }
```

7.18.3.36 importFromJson() [1/2]

```
void Sprite::importFromJson ( ) [protected]
```

Definition at line 105 of file constructor.cpp.

```
106 {
107     json* mem = JSON::readFile(PATH::ATB::SPRITE_ + name + ".json");
108
109     initBackground ((*mem) ["background"]);
110
111     if (mem->contains("objects")) initObjects ((*mem) ["objects"]);
112     if (mem->contains("text-box")) initTextBox ((*mem) ["text-box"]);
113     if (mem->contains("input")) initInput ((*mem) ["input"]);
114     delete mem;
115 }
```

7.18.3.37 importFromJson() [2/2]

```
void Object::importFromJson (
    const json & mem ) [inherited]
```

Definition at line 21 of file constructor.cpp.

```
22 {
23     if (mem.contains("location"))
24         locating(mem["location"]);
25
26     if (mem.contains("crop"))
27         cropping(mem["crop"]);
28
29     if (mem.contains("color"))
30         coloring(mem["color"]);
31
32     if (mem.contains("shape"))
33         setShape(mem["shape"]);
34
35     if (mem.contains("visible"))
36         visible = mem["visible"];
37
38     if (mem.contains("image"))
39         textureFromFile(PATH::ASSETS::GRAPHICS_ + mem["image"].get<std::string>());
40     return ;
41 }
```

7.18.3.38 initBackground()

```
void Sprite::initBackground (
    const json & mem ) [protected]
```

Definition at line 55 of file constructor.cpp.

```
56 {
57     Object::importFromJson(mem);
58     if (Object::getColor() == nullptr) return ;
59     cacheColor = *Object::getColor();
60 }
```

7.18.3.39 initInput()

```
void Sprite::initInput (
    const json & mem ) [protected]
```

Definition at line 94 of file constructor.cpp.

```
95 {
96     if(textBox == nullptr) return;
97     if(mem.contains("digit")) receiveDigit = mem["digit"].get<bool>();
98     if(mem.contains("letter")) receiveLetter = mem["letter"].get<bool>();
99     if(mem.contains("symbol")) receiveSymbol = mem["symbol"].get<bool>();
100    if(mem.contains("lower")) numberLower = mem["lower"].get<std::string>();
101    if(mem.contains("upper")) numberUpper = mem["upper"].get<std::string>();
102    if(mem.contains("maxsize")) maxSize = mem["maxsize"].get<int>();
103 }
```

7.18.3.40 initObjects()

```
void Sprite::initObjects (
    const json & mem ) [protected]
```

Definition at line 62 of file constructor.cpp.

```
63 {
64     objects.clear();
65     for(auto& i : mem)
66     {
67         Object* obj = new Object(render);
68         if(i.contains("name")) obj->linking(i["name"]);
69
70         obj->importFromJson(i);
71
72         obj->moveX(getX());
73         obj->moveY(getY());
74
75         objects.push_back(obj);
76     }
77 }
```

7.18.3.41 initTextBox()

```
void Sprite::initTextBox (
    const json & mem ) [protected]
```

Definition at line 79 of file constructor.cpp.

```
80 {
81     if(textBox == nullptr)
82     {
83         objects.push_back(new Object(render));
84         textBox = objects.back();
85     }
86     textBox->importFromJson(mem);
87     textBox->setFont(this->font);
88     textBox->moveX(getX());
89     textBox->moveY(getY());
90     if(mem.contains("text")) textBox->setText(mem["text"].get<std::string>());
91     aligning();
92 }
```

7.18.3.42 isLieInside() [1/4]

```
bool Object::isLieInside (
    int x,
    int y ) [inherited]
```

Definition at line 3 of file locating.cpp.

```
4 {
5     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
6     return (x >= location->x && x < location->x + location->w && y >= location->y && y < location->y +
        location->h);
7 }
```

7.18.3.43 isLieInside() [2/4]

```
bool Object::isLieInside (
    int x,
    int y,
    int w,
    int h ) [inherited]
```

Definition at line 21 of file locating.cpp.

```
22 {
23     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
24     return (x >= location->x && x + w <= location->x + location->w && y >= location->y && y + h <=
        location->y + location->h);
25 }
```

7.18.3.44 isLieInside() [3/4]

```
bool Object::isLieInside (
    SDL_Point p ) [inherited]
```

Definition at line 9 of file locating.cpp.

```
10 {
11     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
12     return (p.x >= location->x && p.x < location->x + location->w && p.y >= location->y && p.y <
        location->y + location->h);
13 }
```

7.18.3.45 isLieInside() [4/4]

```
bool Object::isLieInside (
    SDL_Rect r ) [inherited]
```

Definition at line 15 of file locating.cpp.

```
16 {
17     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
18     return (r.x >= location->x && r.x + r.w <= location->x + location->w && r.y >= location->y && r.y +
        r.h <= location->y + location->h);
19 }
```

7.18.3.46 isVisible()

```
bool Object::isVisible ( ) [inherited]
```

Definition at line 13 of file visible.cpp.

```
14 {
15     return visible;
16 }
```

7.18.3.47 linking()

```
void Sprite::linking (
    std::string n )
```

Definition at line 21 of file constructor.cpp.

```
22 {
23     name = n;
24     importFromJson();
25 }
```

7.18.3.48 locating() [1/3]

```
void Object::locating (
    const json & mem ) [virtual], [inherited]
```

Definition at line 70 of file locating.cpp.

```
71 {
72     if(mem.contains("x") && mem.contains("y") && mem.contains("w") && mem.contains("h"))
73         locating(mem["x"], mem["y"], mem["w"], mem["h"]);
74 }
```

7.18.3.49 locating() [2/3]

```
void Sprite::locating (
    int x,
    int y,
    int w,
    int h ) [override], [virtual]
```

Reimplemented from [Object](#).

Definition at line 3 of file locating.cpp.

```
4 {
5     Object::locating(x, y, w, h);
6     for (auto &i : objects)
7     {
8         i->locating(x, y, w, h);
9     }
10 }
```

7.18.3.50 locating() [3/3]

```
void Sprite::locating (
    SDL_Rect r ) [override], [virtual]
```

Reimplemented from [Object](#).

Definition at line 12 of file locating.cpp.

```
13 {
14     Object::locating(r);
15     for (auto &i : objects)
16     {
17         i->locating(r);
18     }
19 }
```

7.18.3.51 locatingH()

```
void Object::locatingH (
    int h ) [virtual], [inherited]
```

Definition at line 94 of file locating.cpp.

```
95 {
96     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
97     location->h = h;
98 }
```

7.18.3.52 locatingW()

```
void Object::locatingW (
    int w ) [virtual], [inherited]
```

Definition at line 88 of file locating.cpp.

```
89 {
90     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
91     location->w = w;
92 }
```

7.18.3.53 locatingX()

```
void Sprite::locatingX (
    int x ) [override], [virtual]
```

Reimplemented from [Object](#).

Definition at line 20 of file locating.cpp.

```
21 {
22     Object::locatingX(x);
23     for (auto &i : objects)
24     {
25         i->locatingX(x);
26     }
27 }
```

7.18.3.54 locatingY()

```
void Sprite::locatingY (
    int y ) [override], [virtual]
```

Reimplemented from [Object](#).

Definition at line 29 of file locating.cpp.

```
30 {
31     Object::locatingY(y);
32     for (auto &i : objects)
33     {
34         i->locatingY(y);
35     }
36 }
```

7.18.3.55 moveX()

```
void Sprite::moveX (
    int x ) [override], [virtual]
```

Reimplemented from [Object](#).

Definition at line 38 of file locating.cpp.

```
39 {
40     Object::moveX(x);
41     for (auto &i : objects)
42     {
43         i->moveX(x);
44     }
45 }
```

7.18.3.56 moveY()

```
void Sprite::moveY (
    int y ) [override], [virtual]
```

Reimplemented from [Object](#).

Definition at line 47 of file locating.cpp.

```
48 {
49     Object::moveY(y);
50     for (auto &i : objects)
51     {
52         i->moveY(y);
53     }
54 }
```

7.18.3.57 noCropping()

```
void Object::noCropping ( ) [inherited]
```

Definition at line 32 of file cropping.cpp.

```
33 {
34     if(crop != nullptr) delete crop;
35     crop = nullptr;
36 }
```


7.18.3.58 removeCharacter() [1/2]

```
void Object::removeCharacter ( ) [inherited]
```

Definition at line 28 of file font.cpp.

```
29 {  
30     if (text.size() > 0)  
31         text.pop_back();  
32     textToTexture();  
33 }
```

7.18.3.59 removeCharacter() [2/2]

```
void Object::removeCharacter (  
    int n ) [inherited]
```

Definition at line 35 of file font.cpp.

```
36 {  
37     if(n == 0) return ;  
38     if(text.size() <= n) text.clear();  
39     else text.erase(text.end() - n, text.end());  
40     textToTexture();  
41 }
```

7.18.3.60 rendering()

```
void Sprite::rendering ( ) [override], [virtual]
```

Reimplemented from [Object](#).

Definition at line 3 of file rendering.cpp.

```
4 {  
5     Object::rendering();  
6     for(auto& obj : objects)  
7     {  
8         obj->rendering();  
9     }  
10 }
```

7.18.3.61 setFont()

```
void Sprite::setFont (  
    TTF_Font * f )
```

Definition at line 26 of file constructor.cpp.

```
27 {  
28     font = f;  
29     Object::setFont(f);  
30 }
```

7.18.3.62 setFontColor()

```
void Sprite::setFontColor (
    SDL_Color c )
```

Definition at line 31 of file constructor.cpp.

```
32 {
33     if (textBox != nullptr)
34     {
35         textBox->coloring(c);
36         textBox->setText(textBox->getText());
37     }
38 }
```

7.18.3.63 setShape()

```
void Object::setShape (
    const json & mem ) [inherited]
```

Definition at line 52 of file shape.cpp.

```
53 {
54     if (mem["type"].get<std::string>() == "CIRCLE")
55     {
56         if (mem.contains("center"))
57         {
58             if (mem.contains("radius"))
59                 changeToCircle(mem["center"]["x"], mem["center"]["y"], mem["radius"]);
60             else changeToCircle(mem["center"]["x"], mem["center"]["y"]);
61         } else changeToCircle();
62     }
63     return ;
64 }
65
66 if (mem["type"].get<std::string>() == "NONE" || mem["type"].get<std::string>() == "RECTANGLE")
67 {
68     changeToRectangle();
69     return ;
70 }
71
72 }
```

7.18.3.64 setText()

```
void Sprite::setText (
    std::string s )
```

Definition at line 12 of file textbox.cpp.

```
13 {
14     if (textBox == nullptr) return ;
15     for (char c : s)
16     {
17         if (!receiveDigit && NUMBER::isDigit(c)) return ;
18         if (!receiveLetter && NUMBER::isLetter(c)) return ;
19         if (!receiveSymbol && NUMBER::isSymbol(c)) return ;
20     }
21     if (s.size() > maxSize) s = s.substr(0, maxSize);
22     textBox->setText(s);
23     aligning();
24 }
```

7.18.3.65 setTextBox()

```
void Sprite::setTextBox (
    TTF_Font * f )
```

Definition at line 39 of file constructor.cpp.

```
40 {
41     if(textBox == nullptr)
42     {
43         objects.push_back(new Object(render));
44         textBox = objects.back();
45     }
46     textBox->setFont(f);
47 }
```

7.18.3.66 setTextBoxTransparent()

```
void Sprite::setTextBoxTransparent (
    int a )
```

Definition at line 4 of file textbox.cpp.

```
5 {
6     if(textBox == nullptr) return ;
7     SDL_Color c = *getColor();
8     c.a = a;
9     coloring(c);
10 }
```

7.18.3.67 show()

```
void Object::show ( ) [inherited]
```

Definition at line 3 of file visible.cpp.

```
4 {
5     visible = true;
6 }
```

7.18.3.68 textToTexture()

```
void Object::textToTexture ( ) [protected], [inherited]
```

Definition at line 43 of file font.cpp.

```
44 {
45     if(font == nullptr) return ;
46     if(color == nullptr) return ;
47     if(render == nullptr) return ;
48     if(texture != nullptr)
49     {
50         SDL_DestroyTexture(texture);
51     }
52     texture = nullptr;
53     SDL_Surface* surface = TTF_RenderText_Blended(font, text.c_str(), *color);
54     if(surface == nullptr) return ;
55     texture = SDL_CreateTextureFromSurface(render, surface);
56     SDL_FreeSurface(surface);
57     fitTheTexture();
58 }
```

7.18.3.69 textureFromFile()

```
void Object::textureFromFile (
    std::string dir ) [inherited]
```

Definition at line 4 of file external_storage.cpp.

```
5 {
6     SDL_Surface *surface = IMG_Load(dir.c_str());
7
8     texture = SDL_CreateTextureFromSurface(render, surface);
9     SDL_FreeSurface(surface);
10 }
```

7.18.3.70 typing() [1/2]

```
void Sprite::typing (
    char c )
```

Definition at line 26 of file textbox.cpp.

```
27 {
28     if(textBox == nullptr) return ;
29     if(!receiveDigit && NUMBER::isDigit(c)) return ;
30     if(!receiveLetter && NUMBER::isLetter(c)) return ;
31     if(!receiveSymbol && NUMBER::isSymbol(c)) return ;
32     if(textBox->getSize() >= maxSize) return ;
33     textBox->addCharacter(c);
34     aligning();
35 }
```

7.18.3.71 typing() [2/2]

```
void Sprite::typing (
    std::string s )
```

Definition at line 37 of file textbox.cpp.

```
38 {
39     if(textBox == nullptr) return ;
40     if(textBox->getText().size() + s.size() >= maxSize) return ;
41     for(char c : s)
42     {
43         if(!receiveDigit && NUMBER::isDigit(c)) return ;
44         if(!receiveLetter && NUMBER::isLetter(c)) return ;
45         if(!receiveSymbol && NUMBER::isSymbol(c)) return ;
46     }
47     if(textBox->getSize() + s.size() > maxSize) s = s.substr(0, maxSize);
48     textBox->addText(s);
49     aligning();
50 }
```

7.18.3.72 unhighlight()

```
void Sprite::unhighlight ( )
```

Definition at line 15 of file coloring.cpp.

```
16 {
17     Object::coloring(cacheColor);
18 }
```

7.18.3.73 zoom()

```
void Object::zoom (
    double delta ) [virtual], [inherited]
```

Definition at line 123 of file locating.cpp.

```
124 {
125     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
126     location->w *= delta;
127     location->h *= delta;
128 }
```

7.18.3.74 zoomH()

```
void Object::zoomH (
    int delta ) [virtual], [inherited]
```

Definition at line 118 of file locating.cpp.

```
119 {
120     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
121     location->h += delta;
122 }
```

7.18.3.75 zoomInMiddle()

```
void Object::zoomInMiddle (
    double delta ) [virtual], [inherited]
```

Definition at line 130 of file locating.cpp.

```
131 {
132     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
133     SDL_Point center = {location->x + location->w / 2, location->y + location->h / 2};
134     location->w *= delta;
135     location->h *= delta;
136     location->x = center.x - location->w / 2;
137     location->y = center.y - location->h / 2;
138 }
```

7.18.3.76 zoomW()

```
void Object::zoomW (
    int delta ) [virtual], [inherited]
```

Definition at line 112 of file locating.cpp.

```
113 {
114     if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
115     location->w += delta;
116 }
```

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

- [include/sprite.hpp](#)
- [src/sprite/aligning.cpp](#)
- [src/sprite/coloring.cpp](#)
- [src/sprite/constructor.cpp](#)
- [src/sprite/destructor.cpp](#)
- [src/sprite/locating.cpp](#)
- [src/sprite/rendering.cpp](#)
- [src/sprite/textbox.cpp](#)

7.19 Trie Class Reference

[Trie](#) data structure.

```
#include <trie.hpp>
```

Public Member Functions

- [Trie](#) (SDL_Renderer *r, std::mutex &m, TTF_Font *f, SDL_Rect v, int capacity)
- [~Trie](#) ()
- void [init](#) (std::vector< std::string > words)
- bool [insert](#) (std::string word)
- bool [search](#) (std::string word)
- bool [remove](#) (std::string word)
- void [setEdgesColor](#) (SDL_Color c)
- void [setNodeColor](#) (SDL_Color bg, SDL_Color fg)
- void [rendering](#) ()
- bool [isReceiveEvent](#) (SDL_Event &e)
- [Button](#) * [react](#) (SDL_Event &e)
- void [closeScript](#) ()
- void [setting](#) (SDL_Color c1, SDL_Color c2, SDL_Color c3, SDL_Color c4)

Protected Member Functions

- Node * [insert](#) (Node *node, std::string word, int index)
- bool [search](#) (Node *node, std::string word, int index)
- Node * [remove](#) (Node *node, std::string word, int index)
- int [locating](#) (Node *node, int shiftDown, int shiftRight)
- void [drawEdges](#) (Node *u, Node *v)
- void [waitForStep](#) ()
- void [highlight](#) (std::vector< int > l)
- void [unhighlight](#) (std::vector< int > l)

7.19.1 Detailed Description

[Trie](#) data structure.

Definition at line 20 of file `trie.hpp`.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 Trie()

```
Trie::Trie (
    SDL_Renderer * r,
    std::mutex & m,
    TTF_Font * f,
    SDL_Rect v,
    int capacity )
```

Definition at line 14 of file constructor.cpp.

```
14                                     : ds_mutex(m)
15 {
16     render = r;
17     font = f;
18     viewport = v;
19     capacity = cap;
20     size = 0;
21     root = nullptr;
22     edgesColor = {255, 255, 255, 255};
23     fontColor = {255, 255, 255, 255};
24     nodeColor = {20, 85, 185, 255};
25     shiftX = 20;
26     shiftX = 20;
27     distanceX = 60;
28     distanceY = 80;
29     isMoving = false;
30
31     isQueue = false;
32     isPause = false;
33     stepWait = 600;
34     isAnimate = false;
35
36     std::string fontpath = PATH::ASSETS::FONTS_ + "nimbus-sans-l/regular.otf";
37     scriptFont = TTF_OpenFont(fontpath.c_str(), 18);
38
39     currentScript = nullptr;
40     Script* insert = new Script(render, scriptFont);
41     insert->linking("trie/insert");
42     scripts[DATA_STRUCTURES_OPERATOR::INSERT] = insert;
43
44     Script* remove = new Script(render, scriptFont);
45     remove->linking("trie/remove");
46     scripts[DATA_STRUCTURES_OPERATOR::DELETE] = remove;
47
48     Script* search = new Script(render, scriptFont);
49     search->linking("trie/search");
50     scripts[DATA_STRUCTURES_OPERATOR::SEARCH] = search;
51
52     Script* init = new Script(render, scriptFont);
53     init->linking("trie/init");
54     scripts[DATA_STRUCTURES_OPERATOR::INIT] = init;
55 }
```

7.19.2.2 ~Trie()

```
Trie::~Trie ( )
```

Definition at line 9 of file destructor.cpp.

```
10 {
11     if(root != nullptr) delete root;
12
13     for(auto i : scripts)
14         delete i.second;
15     TTF_CloseFont(scriptFont);
16 }
```

7.19.3 Member Function Documentation

7.19.3.1 closeScript()

```
void Trie::closeScript ( )
```

Definition at line 72 of file event.cpp.

```
73 {
74     currentScript = nullptr;
75 }
```

7.19.3.2 drawEgdes()

```
void Trie::drawEgdes (
    Node * u,
    Node * v ) [protected]
```

Definition at line 4 of file rendering.cpp.

```
5 {
6     SDL_Point psrc = {u->sprite->getX() + u->sprite->getW() / 2, u->sprite->getY() + u->sprite->getH() /
7     2};
8     SDL_Point pdst = {v->sprite->getX() + v->sprite->getW() / 2, v->sprite->getY() + v->sprite->getH() /
9     2};
10    SDL_SetRenderDrawColor(render, edgesColor.r, edgesColor.g, edgesColor.b, edgesColor.a);
11    for(int i = -1; i <= 1; i++)
12    {
13        for(int j = -1; j <= 1; j++)
14            SDL_RenderDrawLine(render, psrc.x + i, psrc.y + j, pdst.x + i, pdst.y + j);
15 }
```

7.19.3.3 highlight()

```
void Trie::highlight (
    std::vector< int > l ) [protected]
```

Definition at line 26 of file step.cpp.

```
27 {
28     if(isAnimate)
29     {
30         animate_mutex.lock();
31         for(int i = 0; i < l.size(); i++)
32         {
33             currentScript->highlight(l[i]);
34         }
35         animate_mutex.unlock();
36     }
37 }
```

7.19.3.4 init()

```
void Trie::init (
    std::vector< std::string > words )
```

Definition at line 4 of file init.cpp.

```
5 {
6     if(root != nullptr) delete root;
7     root = nullptr;
8     currentScript = scripts[DATA_STRUCTURES_OPERATOR::INIT];
9     isAnimate = false;
10    for(auto word : words)
11    {
12        root = insert(root, word, 0);
13    }
14 }
```


7.19.3.5 insert() [1/2]

```
Trie::Node * Trie::insert (
    Node * node,
    std::string word,
    int index ) [protected]
```

Definition at line 3 of file insert.cpp.

```
4 {
5     if(node == nullptr)
6     {
7         Sprite* spr = new Sprite(render);
8         spr->setFont(font);
9         spr->linking("trie/node");
10        spr->typing(' ');
11        spr->setFontColor(fontColor);
12        spr->coloring(nodeColor);
13
14        node = new Node('\0', spr);
15        if(root == nullptr) root = node;
16        node->sprite->coloring(nodeColor);
17
18        if(isAnimate)
19        {
20            highlight({1, 2, 3});
21            waitForStep();
22            unhighlight({1, 2, 3});
23        }
24    }
25    if(isAnimate)
26    {
27
28        animate_mutex.lock();
29        node->sprite->highlight();
30        animate_mutex.unlock();
31
32        waitForStep();
33
34        animate_mutex.lock();
35        node->sprite->unhighlight();
36        animate_mutex.unlock();
37    }
38
39    highlight({4, 5, 6, 7, 8});
40    waitForStep();
41    unhighlight({4, 5, 6, 7, 8});
42
43    node->numberOfWords++;
44    if(index == (int) word.size())
45    {
46        node->endOfWords++;
47        return node;
48    }
49
50    highlight({9, 10, 11, 12, 13, 14});
51    waitForStep();
52    unhighlight({9, 10, 11, 12, 13, 14});
53
54    int key = word[index] - 'a';
55
56    Node *& currentChild = node->childs[key];
57    if(currentChild == nullptr)
58    {
59        Sprite* spr = new Sprite(render);
60        spr->setFont(font);
61        spr->linking("trie/node");
62        spr->typing(word[index]);
63        spr->setFontColor(fontColor);
64        spr->coloring(nodeColor);
65
66        currentChild = new Node(key, spr);
67    }
68    currentChild = insert(currentChild, word, index + 1);
69
70    if(isAnimate)
71    {
72
73        animate_mutex.lock();
74        node->sprite->highlight();
75        animate_mutex.unlock();
76
77        waitForStep();
78
```

```

79         animate_mutex.lock();
80         node->sprite->unhighlight();
81         animate_mutex.unlock();
82     }
83     highlight({15});
84     waitForStep();
85     unhighlight({15});
86     return node;
87 }

```

7.19.3.6 insert() [2/2]

```

bool Trie::insert (
    std::string word )

```

Definition at line 89 of file insert.cpp.

```

90 {
91
92     currentScript = scripts[DATA_STRUCTURES_OPERATOR::INSERT];
93     isAnimate = true;
94
95
96     highlight({0});
97     waitForStep();
98     unhighlight({0});
99
100     root = insert(root, word, 0);
101     return true;
102 }

```

7.19.3.7 isReceiveEvent()

```

bool Trie::isReceiveEvent (
    SDL_Event & e )

```

Definition at line 3 of file event.cpp.

```

4 {
5     switch(e.type)
6     {
7         case SDL_MOUSEBUTTONDOWN:
8             if(currentScript != nullptr && currentScript->isReceiveEvent(e)) return true;
9             if(e.motion.x < viewport.x || viewport.x + viewport.w < e.motion.x) return false;
10            if(e.motion.y < viewport.y || viewport.y + viewport.h < e.motion.y) return false;
11            if(e.button.button == SDL_BUTTON_LEFT) return false;
12            if(root == nullptr) return false;
13            return true;
14            break;
15        case SDL_MOUSEMOTION:
16            if(isMoving) return true;
17            if(currentScript == nullptr) return false;
18            if(currentScript->isReceiveEvent(e)) return true;
19            return false;
20            break;
21        default:
22            return false;
23            break;
24    }
25 }

```

7.19.3.8 locating()

```
int Trie::locating (
    Node * node,
    int shiftDown,
    int shiftRight ) [protected]
```

Definition at line 57 of file constructor.cpp.

```
58 {
59     if(node == nullptr) return 0;
60
61     int j = 0;
62     for(int i = 0; i < 26; i++)
63         if(node->childs[i] != nullptr) j++;
64     j /= 2;
65
66     int i = 0;
67
68     for(; j > 0; i++)
69     {
70         if(node->childs[i] == nullptr)
71             continue;
72         j--;
73
74         shiftRight = locating(node->childs[i], shiftDown + 1, shiftRight);
75     }
76
77     node->sprite->locatingX(shiftX + shiftRight * distanceX);
78     node->sprite->locatingY(shiftY + shiftDown * distanceY);
79     node->sprite->aligning(HORIZONTAL_ALIGN::CENTER, VERTICAL_ALIGN::CENTER);
80     shiftRight++;
81
82     for(; i < 26; i++)
83     {
84         if(node->childs[i] == nullptr)
85             continue;
86         shiftRight = locating(node->childs[i], shiftDown + 1, shiftRight);
87     }
88
89     return shiftRight;
90 }
```

7.19.3.9 react()

```
Button * Trie::react (
    SDL_Event & e )
```

Definition at line 27 of file event.cpp.

```
28 {
29     switch(e.type)
30     {
31         case SDL_MOUSEBUTTONDOWN:
32             if(currentScript != nullptr && currentScript->isReceiveEvent(e))
33             {
34                 return currentScript->react(e);
35             }
36             if(isMoving)
37             {
38                 isMoving = false;
39                 int dx = e.motion.x - lastMousePressed.x;
40                 int dy = e.motion.y - lastMousePressed.y;
41                 shiftX += dx;
42                 shiftY += dy;
43             }else
44             {
45                 isMoving = true;
46                 lastMousePressed.x = e.motion.x;
47                 lastMousePressed.y = e.motion.y;
48             }
49             return nullptr;
50             break;
51         case SDL_MOUSEMOTION:
52             {
53                 if(currentScript != nullptr && currentScript->isReceiveEvent(e))
```

```

54         return currentScript->react(e);
55         if(!isMoving) return nullptr;
56         int dx = e.motion.x - lastMousePressed.x;
57         int dy = e.motion.y - lastMousePressed.y;
58         lastMousePressed.x = e.motion.x;
59         lastMousePressed.y = e.motion.y;
60         shiftX += dx;
61         shiftY += dy;
62         return nullptr;
63         break;
64     }
65     default:
66         return nullptr;
67         break;
68     }
69     return nullptr;
70 }

```

7.19.3.10 remove() [1/2]

```

Trie::Node * Trie::remove (
    Node * node,
    std::string word,
    int index ) [protected]

```

Definition at line 3 of file remove.cpp.

```

4 {
5     if(node == nullptr)
6     {
7         highlight({1, 2, 3});
8         waitForStep();
9         unhighlight({1, 2, 3});
10        return nullptr;
11    }
12    node->numberOfWords--;
13
14    if(isAnimate)
15    {
16        animate_mutex.lock();
17        node->sprite->highlight();
18        animate_mutex.unlock();
19
20        waitForStep();
21    }
22
23    if(index == (int) word.size())
24    {
25        highlight({4, 5});
26        waitForStep();
27        unhighlight({4, 5});
28
29        node->endOfWords--;
30    }else
31    {
32        highlight({6, 7, 8, 9, 10, 11, 12, 13});
33        waitForStep();
34        unhighlight({6, 7, 8, 9, 10, 11, 12, 13});
35
36        if(isAnimate)
37        {
38            animate_mutex.lock();
39            node->sprite->unhighlight();
40            animate_mutex.unlock();
41        }
42
43        int key = word[index] - 'a';
44        node->childs[key] = remove(node->childs[key], word, index + 1);
45    }
46    if(isAnimate)
47    {
48        animate_mutex.lock();
49        node->sprite->highlight();
50        animate_mutex.unlock();
51
52        waitForStep();
53        animate_mutex.lock();
54        node->sprite->unhighlight();

```

```

55         animate_mutex.unlock();
56     }
57
58     if (node->numberOfWords == 0)
59     {
60         highlight({14});
61         waitForStep();
62         unhighlight({14});
63         delete node;
64         node = nullptr;
65     }
66
67     return node;
68 }

```

7.19.3.11 remove() [2/2]

```

bool Trie::remove (
    std::string word )

```

Definition at line 70 of file remove.cpp.

```

71 {
72     currentScript = scripts[DATA_STRUCTURES_OPERATOR::DELETE];
73
74     isAnimate = false;
75     if (!search(root, word, 0))
76     {
77         isAnimate = true;
78         highlight({1, 2, 3});
79         waitForStep();
80         unhighlight({1, 2, 3});
81
82         return false;
83     }
84     isAnimate = true;
85
86     highlight({0});
87     waitForStep();
88     unhighlight({0});
89
90     root = remove(root, word, 0);
91     return true;
92 }

```

7.19.3.12 rendering()

```

void Trie::rendering ( )

```

Definition at line 17 of file rendering.cpp.

```

18 {
19     if (root == nullptr) return ;
20     SDL_RenderSetViewport(render, &viewport);
21     locating(root, 0, 0);
22
23     std::queue<Node*> q;
24     q.push(root);
25
26     while (!q.empty())
27     {
28         Node* u = q.front();
29         q.pop();
30
31         for (int i = 0; i < 26; i++)
32         {
33             if (u->childs[i] != nullptr)
34             {
35                 q.push(u->childs[i]);
36                 drawEdges(u, u->childs[i]);
37             }
38         }
39     }

```

```

39         u->sprite->rendering();
40     }
41     if(currentScript != nullptr)
42     {
43         SDL_RenderSetViewport(render, nullptr);
44         currentScript->rendering();
45     }
46 }

```

7.19.3.13 search() [1/2]

```

bool Trie::search (
    Node * node,
    std::string word,
    int index ) [protected]

```

Definition at line 3 of file search.cpp.

```

4 {
5     if(node == nullptr)
6     {
7         highlight({1, 2, 3});
8         waitForStep();
9         unhighlight({1, 2, 3});
10        return false;
11    }
12    if(isAnimate)
13    {
14        animate_mutex.lock();
15        node->sprite->highlight();
16        animate_mutex.unlock();
17
18        waitForStep();
19    }
20 }
21 if(index == (int) word.size())
22 {
23     highlight({4, 5, 6});
24     waitForStep();
25     unhighlight({4, 5, 6});
26     if(isAnimate)
27     {
28         animate_mutex.lock();
29         node->sprite->unhighlight();
30         animate_mutex.unlock();
31     }
32     return node->endOfWords > 0;
33 }
34 int key = word[index] - 'a';
35 if(isAnimate)
36 {
37     animate_mutex.lock();
38     node->sprite->unhighlight();
39     animate_mutex.unlock();
40 }
41
42 highlight({7, 8, 9, 10, 11, 12});
43 waitForStep();
44 unhighlight({7, 8, 9, 10, 11, 12});
45
46 return search(node->childs[key], word, index + 1);
47 }

```

7.19.3.14 search() [2/2]

```

bool Trie::search (
    std::string word )

```

Definition at line 49 of file search.cpp.

```

50 {
51     currentScript = scripts[DATA_STRUCTURES_OPERATOR::SEARCH];
52     isAnimate = true;
53
54     highlight({0});
55     waitForStep();
56     unhighlight({0});
57
58     return search(root, word, 0);
59 }

```

7.19.3.15 setEdgesColor()

```

void Trie::setEdgesColor (
    SDL_Color c )

```

7.19.3.16 setNodeColor()

```

void Trie::setNodeColor (
    SDL_Color bg,
    SDL_Color fg )

```

7.19.3.17 setting()

```

void Trie::setting (
    SDL_Color c1,
    SDL_Color c2,
    SDL_Color c3,
    SDL_Color c4 )

```

Definition at line 92 of file constructor.cpp.

```

93 {
94     bgColor = c1;
95     nodeColor = c2;
96     fontColor = c3;
97     edgesColor = c4;
98
99     std::queue<Node*> q;
100     if(root != nullptr)
101         q.push(root);
102
103     while(!q.empty())
104     {
105         Node* node = q.front();
106         q.pop();
107
108         node->sprite->coloring(nodeColor);
109         node->sprite->setFontColor(fontColor);
110         node->sprite->coloring(nodeColor);
111
112         for(int i = 0; i < 26; i++)
113             if(node->childs[i] != nullptr)
114                 q.push(node->childs[i]);
115     }
116 }

```

7.19.3.18 unhighlight()

```
void Trie::unhighlight (
    std::vector< int > l ) [protected]
```

Definition at line 39 of file step.cpp.

```
40 {
41     if (isAnimate)
42     {
43         animate_mutex.lock();
44         for (int i = 0; i < l.size(); i++)
45         {
46             currentScript->unhighlight (l[i]);
47         }
48         animate_mutex.unlock();
49     }
50 }
```

7.19.3.19 waitForStep()

```
void Trie::waitForStep ( ) [protected]
```

Definition at line 5 of file step.cpp.

```
6 {
7     if (isAnimate)
8     {
9         ds_mutex.unlock();
10        std::this_thread::sleep_for (std::chrono::milliseconds (stepWait));
11        ds_mutex.lock();
12    }
13    std::lock_guard<std::mutex> pause_lock (pause_mutex);
14    if (isPause == false)
15    {
16        return ;
17    }
18
19    ds_mutex.unlock();
20    std::unique_lock<std::mutex> lk (step_mutex);
21    step_cv.wait (lk, [&]{return isQueue == true;});
22    isQueue = false;
23    ds_mutex.lock();
24 }
```

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

- [include/data_structures/trie.hpp](#)
- [src/trie/constructor.cpp](#)
- [src/trie/destructor.cpp](#)
- [src/trie/event.cpp](#)
- [src/trie/operator/init.cpp](#)
- [src/trie/operator/insert.cpp](#)
- [src/trie/operator/remove.cpp](#)
- [src/trie/operator/search.cpp](#)
- [src/trie/rendering.cpp](#)
- [src/trie/step.cpp](#)

Chapter 8

File Documentation

8.1 include/button.hpp File Reference

```
#include <vector>
#include <string>
#include <object.hpp>
#include <sprite.hpp>
#include <GLOBAL.hpp>
```

Classes

- class [Button](#)
[Button](#) class that interact with user input.

8.2 include/data_structures.hpp File Reference

```
#include <iostream>
#include <vector>
#include <string>
#include <mutex>
#include <SDL2/SDL.h>
#include <GLOBAL.hpp>
#include <object.hpp>
#include <sprite.hpp>
#include <services.hpp>
#include <display.hpp>
#include <inputbox.hpp>
#include <data_structures/AVL.hpp>
#include <data_structures/trie.hpp>
#include <data_structures/hash_table.hpp>
#include <data_structures/minheap.hpp>
#include <data_structures/maxheap.hpp>
#include <data_structures/graph.hpp>
```

Classes

- class [DataStructures](#)
Container that contains all data structures.

8.3 include/data_structures/AVL.hpp File Reference

```
#include <iostream>
#include <cmath>
#include <mutex>
#include <condition_variable>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <sprite.hpp>
#include <script.hpp>
```

Classes

- class [AVL](#)
AVL class.

8.4 include/data_structures/btree4th.hpp File Reference

```
#include <vector>
#include <string>
#include <iostream>
```

Classes

- class [BTree4th](#)

8.5 include/data_structures/graph.hpp File Reference

```
#include <vector>
#include <string>
#include <queue>
#include <stack>
#include <iostream>
#include <mutex>
#include <thread>
#include <condition_variable>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <sprite.hpp>
#include <button.hpp>
```

Classes

- class [Graph](#)
[Graph](#) class.

8.6 include/data_structures/hash_table.hpp File Reference

```
#include <iostream>
#include <vector>
#include <map>
#include <mutex>
#include <condition_variable>
#include <thread>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <sprite.hpp>
#include <GLOBAL.hpp>
#include <script.hpp>
```

Classes

- class [HashTable](#)
[HashTable](#) class.

8.7 include/data_structures/maxheap.hpp File Reference

```
#include <iostream>
#include <vector>
#include <string>
```

Classes

- class [maxHeap](#)

8.8 include/data_structures/minheap.hpp File Reference

```
#include <iostream>
#include <vector>
#include <string>
#include <mutex>
#include <thread>
#include <condition_variable>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <sprite.hpp>
#include <button.hpp>
#include <script.hpp>
```

Classes

- class [minHeap](#)
heap class.

8.9 include/data_structures/trie.hpp File Reference

```
#include <iostream>
#include <string>
#include <vector>
#include <mutex>
#include <condition_variable>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <script.hpp>
#include <sprite.hpp>
```

Classes

- class [Trie](#)
Trie data structure.

8.10 include/display.hpp File Reference

```
#include <vector>
#include <string>
#include <SDL2/SDL.h>
#include <services.hpp>
#include <GLOBAL.hpp>
#include <object.hpp>
#include <button.hpp>
```

Classes

- class [Display](#)
container of button intermediate between button and user input, window

8.11 include/GLOBAL.hpp File Reference

```
#include <string>
#include <random>
#include <fstream>
#include <nlohmann/json.hpp>
```

Namespaces

- [DISPLAY](#)
Name of display.
- [PATH](#)
Path to assets, attributes, and saving files.
- [PATH::ASSETS](#)
- [PATH::ATB](#)
- [PATH::SAVING](#)

Typedefs

- using [json](#) = [nlohmann::json](#)

Enumerations

- enum class [WINDOW_STATUS](#) { [IS_OPEN](#) , [IS_CLOSED](#) }
Status of window.
- enum class [SHAPE](#) {
[NONE](#) , [RECTANGLE](#) , [CIRCLE](#) , [TRIANGLE](#) ,
[LINE](#) , [POLYGON](#) }
Kind of simple shape.
- enum class [BUTTON_STATUS](#) { [NONE](#) , [HOVER](#) , [RELEASED](#) , [CLICKED](#) }
Status of button.
- enum class [BUTTON_ACTION](#) {
[CHANGE_SCREEN](#) , [INSERT](#) , [DELETE](#) , [INIT](#) ,
[SEARCH](#) , [SETTING](#) , [DONE](#) , [EDGES](#) ,
[GO_BACK](#) , [GO_NEXT](#) , [GO_ON](#) , [GO_OFF](#) ,
[SPEED_UP](#) , [SLOW_DOWN](#) , [CLOSE](#) , [TOP](#) ,
[SIZE](#) , [CONNECTED_COMPONENTS](#) , [MST](#) , [DIJKSTRA](#) ,
[RANDOM](#) , [RANDOM2](#) , [RANDOM3](#) , [RANDOM4](#) ,
[RANDOM5](#) , [RANDOM6](#) , [RANDOM7](#) , [RANDOM8](#) ,
[RANDOM9](#) , [RANDOM10](#) , [RANDOM11](#) , [RANDOM12](#) ,
[RANDOM13](#) , [RANDOM14](#) , [RANDOM15](#) , [RANDOM16](#) ,
[FILE](#) , [NONE](#) }
Type of button.
- enum class [DATA_STRUCTURES_TYPE](#) {
[NONE](#) , [AVL](#) , [HASH_TABLE](#) , [GRAPH](#) ,
[TRIE](#) , [MIN_HEAP](#) , [MAX_HEAP](#) , [BTREE_4TH](#) }
Type of data structures.
- enum class [DATA_STRUCTURES_OPERATOR](#) {
[INIT](#) , [INSERT](#) , [DELETE](#) , [SEARCH](#) ,
[TOP](#) , [SIZE](#) , [SCC](#) , [MST](#) ,
[DIJKSTRA](#) , [SETTING](#) }
Type of data structures operator.
- enum class [HORIZONTAL_ALIGN](#) { [LEFT](#) , [CENTER](#) , [RIGHT](#) }
Align in horizon axis.
- enum class [VERTICAL_ALIGN](#) { [TOP](#) , [CENTER](#) , [BOTTOM](#) }
Align in vertical axis.
- enum class [INPUT_TYPE](#) {
[NONE](#) , [INT](#) , [ARRAY](#) , [STRING](#) ,
[STRINGS](#) }
Type of input.

Variables

- `const std::string DISPLAY::HOME_ = "home"`
- `const std::string DISPLAY::WORKING_ = "working"`
- `const std::string PATH::ASSETS_ = "assets/"`
- `const std::string PATH::ASSETS::GRAPHICS_ = "assets/graphics/"`
- `const std::string PATH::ASSETS::FONTS_ = "assets/fonts/"`
- `const std::string PATH::ASSETS::SCRIPT_ = "assets/script/"`
- `const std::string PATH::ATTRIBUTE_ = "atb/"`
- `const std::string PATH::ATB::SPRITE_ = "atb/sprite/"`
- `const std::string PATH::ATB::OBJECT_ = "atb/object/"`
- `const std::string PATH::ATB::DISPLAY_ = "atb/display/"`
- `const std::string PATH::ATB::BUTTON_ = "atb/button/"`
- `const std::string PATH::ATB::DATA_STRUCTURES_ = "atb/data_structures/"`
- `const std::string PATH::ATB::INPUTBOX_ = "atb/input/"`
- `const std::string PATH::ATB::SCRIPT_ = "atb/script/"`
- `const std::string PATH::SAVING_ = "saving/"`
- `const std::string PATH::SAVING::AVL_ = "saving/AVL.txt"`
- `const std::string PATH::SAVING::HASH_TABLE_ = "saving/HASH_TABLE.txt"`
- `const std::string PATH::SAVING::GRAPH_ = "saving/GRAPH.txt"`
- `const std::string PATH::SAVING::TRIE_ = "saving/TRIE.txt"`
- `const std::string PATH::SAVING::MIN_HEAP_ = "saving/MIN_HEAP.txt"`
- `const std::string PATH::SAVING::MAX_HEAP_ = "saving/MAX_HEAP.txt"`
- `const std::string PATH::SAVING::BTREE_4TH_ = "saving/BTREE_4TH.txt"`

8.11.1 Typedef Documentation

8.11.1.1 json

```
using json = nlohmann::json
```

Definition at line 10 of file GLOBAL.hpp.

8.11.2 Enumeration Type Documentation

Enumerator

8.11.2.1 BUTTON_ACTION

```
enum BUTTON_ACTION [strong]
```

Type of button.

Enumerator

CHANGE_SCREEN	
INSERT	
DELETE	
INIT	
SEARCH	
SETTING	
DONE	
EDGES	
GO_BACK	
GO_NEXT	
GO_ON	
GO_OFF	
SPEED_UP	
SLOW_DOWN	
CLOSE	
TOP	
SIZE	
CONNECTED_COMPONENTS	
MST	
DIJKSTRA	
RANDOM	
RANDOM2	
RANDOM3	
RANDOM4	
RANDOM5	
RANDOM6	
RANDOM7	
RANDOM8	
RANDOM9	
RANDOM10	
RANDOM11	
RANDOM12	
RANDOM13	
RANDOM14	
RANDOM15	
RANDOM16	
FILE	
NONE	

Definition at line 44 of file GLOBAL.hpp.

```

45 {
46     CHANGE_SCREEN,
47     INSERT,
48     DELETE,
49     INIT,
50     SEARCH,
51     SETTING,
52     DONE,
53     EDGES,
54     GO_BACK,
55     GO_NEXT,
56     GO_ON,
57     GO_OFF,
58     SPEED_UP,
59     SLOW_DOWN,
60     CLOSE,
61     TOP,
62     SIZE,
63     CONNECTED_COMPONENTS,
64     MST,
65     DIJKSTRA,
66     RANDOM,
67     RANDOM2,
68     RANDOM3,
69     RANDOM4,
70     RANDOM5,
71     RANDOM6,
72     RANDOM7,
73     RANDOM8,
74     RANDOM9,
75     RANDOM10,
76     RANDOM11,
77     RANDOM12,
78     RANDOM13,
79     RANDOM14,
80     RANDOM15,
81     RANDOM16,
82     FILE,
83     NONE,
84 };

```

8.11.2.2 BUTTON_STATUS

```
enum BUTTON_STATUS [strong]
```

Status of button.

Enumerator

NONE	
HOVER	
RELEASED	
CLICKED	

Definition at line 34 of file GLOBAL.hpp.

```

35 {
36     NONE,
37     HOVER,
38     RELEASED,
39     CLICKED
40 };

```

8.11.2.3 DATA_STRUCTURES_OPERATOR

```
enum DATA_STRUCTURES_OPERATOR [strong]
```


Type of data structures operator.

Enumerator

INIT	
INSERT	
DELETE	
SEARCH	
TOP	
SIZE	
SCC	
MST	
DIJKSTRA	
SETTING	

Definition at line 103 of file GLOBAL.hpp.

```
104 {  
105     INIT,  
106     INSERT,  
107     DELETE,  
108     SEARCH,  
109     TOP,  
110     SIZE,  
111     SCC,  
112     MST,  
113     DIJKSTRA,  
114     SETTING  
115 };
```

8.11.2.4 DATA_STRUCTURES_TYPE

```
enum DATA_STRUCTURES_TYPE [strong]
```

Type of data structures.

Enumerator

NONE	
AVL	
HASH_TABLE	
GRAPH	
TRIE	
MIN_HEAP	
MAX_HEAP	
BTREE_4TH	

Definition at line 88 of file GLOBAL.hpp.

```
89 {  
90     NONE,  
91     AVL,  
92     HASH_TABLE,  
93     GRAPH,  
94     TRIE,  
95     MIN_HEAP,  
96     MAX_HEAP,  
97     BTREE_4TH  
98 };
```

8.11.2.5 HORIZONTAL_ALIGN

enum [HORIZONTAL_ALIGN](#) [strong]

Align in horizon axis.

Enumerator

LEFT	
CENTER	
RIGHT	

Definition at line 119 of file GLOBAL.hpp.

```
120 {  
121     LEFT,  
122     CENTER,  
123     RIGHT  
124 };
```

8.11.2.6 INPUT_TYPE

enum [INPUT_TYPE](#) [strong]

Type of input.

Enumerator

NONE	
INT	
ARRAY	
STRING	
STRINGS	

Definition at line 138 of file GLOBAL.hpp.

```
139 {  
140     NONE,  
141     INT,  
142     ARRAY,  
143     STRING,  
144     STRINGS  
145 };
```

8.11.2.7 SHAPE

enum [SHAPE](#) [strong]

Kind of simple shape.

Enumerator

NONE	
------	--

Enumerator

RECTANGLE	
CIRCLE	
TRIANGLE	
LINE	
POLYGON	

Definition at line 22 of file GLOBAL.hpp.

```

23 {
24     NONE,
25     RECTANGLE,
26     CIRCLE,
27     TRIANGLE,
28     LINE,
29     POLYGON
30 };
```

8.11.2.8 VERTICAL_ALIGN

```
enum VERTICAL_ALIGN [strong]
```

Align in vertical axis.

Enumerator

TOP	
CENTER	
BOTTOM	

Definition at line 128 of file GLOBAL.hpp.

```

129 {
130     TOP,
131     CENTER,
132     BOTTOM
133 };
```

8.11.2.9 WINDOW_STATUS

```
enum WINDOW_STATUS [strong]
```

Status of window.

Enumerator

IS_OPEN	
IS_CLOSED	

Definition at line 14 of file GLOBAL.hpp.

```

15 {
```

```
16     IS_OPEN,  
17     IS_CLOSED  
18 };
```

8.12 include/inputbox.hpp File Reference

```
#include <string>  
#include <vector>  
#include <SDL2/SDL.h>  
#include <object.hpp>  
#include <sprite.hpp>  
#include <button.hpp>
```

Classes

- class [InputBox](#)
Register for user keyboard input.

8.13 include/object.hpp File Reference

```
#include <string>  
#include <SDL2/SDL.h>  
#include <SDL2/SDL_ttf.h>  
#include <nlohmann/json.hpp>  
#include <GLOBAL.hpp>
```

Classes

- struct [Position](#)
location of an object in 2D coordinate
- class [Object](#)
Class that represent shape, image from files, text. Smallest drawable unit.

Typedefs

- using [json](#) = nlohmann::json

8.13.1 Typedef Documentation

8.13.1.1 json

```
using json = nlohmann::json
```

Definition at line 12 of file object.hpp.

8.14 include/script.hpp File Reference

```
#include <iostream>
#include <string>
#include <vector>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <object.hpp>
#include <sprite.hpp>
#include <button.hpp>
```

Classes

- class [Script](#)
Container that contains a pseudo-code.

8.15 include/services.hpp File Reference

```
#include <atomic>
#include <string>
#include <math.h>
#include <climits>
#include <random>
#include <chrono>
#include <nlohmann/json.hpp>
```

Namespaces

- [JSON](#)
*Interact with *.json files.*
- [FILEE](#)
Interact with text files.
- [NUMBER](#)
Convert between string and interger.
- [SIUSTRING](#)
Features for std::string.
- [RANDOM](#)
Random intergers, doubles, strings generator.

Typedefs

- using `json` = `nlohmann::json`

Functions

- `json * JSON::readFile` (`std::string path`)
- `void JSON::saveFile` (`std::string path`, `json *data`)
- `std::vector< std::string > FILEE::readFile` (`std::string path`)
- `int64_t NUMBER::stringToInt` (`std::string s`)
- `std::string NUMBER::intToString` (`int64_t n`)
- `std::vector< int > NUMBER::stringToArray` (`std::string s`)
- `bool NUMBER::isDigit` (`char c`)
- `bool NUMBER::isLetter` (`char c`)
- `bool NUMBER::isSymbol` (`char c`)
- `bool NUMBER::isSign` (`char c`)
- `bool NUMBER::isOperator` (`char c`)
- `std::string NUMBER::removeLeadingZero` (`std::string s`)
- `bool NUMBER::isNumber` (`std::string s`)
- `bool NUMBER::isInInterval` (`std::string s`, `int64_t a`, `int64_t b`)
- `bool SIUSTRING::isSeparator` (`char c`)
- `std::vector< std::string > SIUSTRING::split` (`std::string s`)
- `int RANDOM::getInt` (`int a`, `int b`)
- `std::string RANDOM::getInt` (`int length`, `int a`, `int b`)
- `long long RANDOM::getLongLong` (`long long a`, `long long b`)
- `float RANDOM::getFloat` (`float a`, `float b`)
- `double RANDOM::getDouble` (`double a`, `double b`)
- `char RANDOM::getChar` (`char a`, `char b`)
- `char RANDOM::getChar` ()
- `std::string RANDOM::getString` (`int length`)
- `std::string RANDOM::getString` (`int length`, `char a`, `char b`)
- `bool RANDOM::flipCoin` ()

Variables

- `const int64_t NUMBER::INF` = `LLONG_MAX`
- `std::mt19937 RANDOM::rng` = `std::mt19937(std::chrono::steady_clock::now().time_since_epoch().count())`

8.15.1 Typedef Documentation

8.15.1.1 json

```
using json = nlohmann::json
```

Definition at line 13 of file `services.hpp`.

8.16 include/sprite.hpp File Reference

```
#include <vector>
#include <SDL2/SDL.h>
#include <nlohmann/json.hpp>
#include <SDL2/SDL_ttf.h>
#include <object.hpp>
```

Classes

- class [Sprite](#)
Object container Drawable.

Typedefs

- using [json](#) = nlohmann::json

8.16.1 Typedef Documentation

8.16.1.1 json

```
using json = nlohmann::json
```

Definition at line 15 of file sprite.hpp.

8.17 include/window.hpp File Reference

```
#include <vector>
#include <thread>
#include <mutex>
#include <condition_variable>
#include <map>
#include <queue>
#include <SDL2/SDL.h>
#include <GLOBAL.hpp>
#include <display.hpp>
#include <data_structures.hpp>
#include <inputbox.hpp>
```

Classes

- class [MyWindow](#)
Window class class that create a window and manage it.

8.18 README.md File Reference

8.19 src/AVL/constructor.cpp File Reference

```
#include <algorithm>
#include <queue>
#include <data_structures/AVL.hpp>
```

8.20 src/btree4th/constructor.cpp File Reference

```
#include <data_structures/btree4th.hpp>
```

8.21 src/button/constructor.cpp File Reference

```
#include <iostream>
#include <button.hpp>
#include <GLOBAL.hpp>
#include <services.hpp>
```

8.22 src/data_structures/constructor.cpp File Reference

```
#include <data_structures.hpp>
```

8.23 src/display/constructor.cpp File Reference

```
#include <display.hpp>
#include <iostream>
```

8.24 src/graph/constructor.cpp File Reference

```
#include <data_structures/graph.hpp>
```

8.25 src/hash_table/constructor.cpp File Reference

```
#include <data_structures/hash_table.hpp>
#include <iostream>
#include <services.hpp>
```

8.26 src/inputbox/constructor.cpp File Reference

```
#include "GLOBAL.hpp"
#include <iostream>
#include <SDL2/SDL.h>
#include <inputbox.hpp>
#include <services.hpp>
```

8.27 src/maxheap/constructor.cpp File Reference

```
#include <data_structures/maxheap.hpp>
```

8.28 src/minheap/constructor.cpp File Reference

```
#include <data_structures/minheap.hpp>
```

8.29 src/object/constructor.cpp File Reference

```
#include <iostream>
#include <object.hpp>
#include <GLOBAL.hpp>
#include <services.hpp>
```

8.30 src/script/constructor.cpp File Reference

```
#include <script.hpp>
#include <services.hpp>
```

8.31 src/sprite/constructor.cpp File Reference

```
#include <fstream>
#include <iostream>
#include <sprite.hpp>
#include <GLOBAL.hpp>
#include <services.hpp>
```

8.32 src/trie/constructor.cpp File Reference

```
#include <data_structures/trie.hpp>
#include <queue>
```

8.33 src/window/constructor.cpp File Reference

```
#include <window.hpp>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_ttf.h>
```

8.34 src/AVL/destructor.cpp File Reference

```
#include <data_structures/AVL.hpp>
```

8.35 src/btree4th/destructor.cpp File Reference

```
#include <data_structures/btree4th.hpp>
```

8.36 src/button/destructor.cpp File Reference

```
#include <button.hpp>
```

8.37 src/data_structures/destructor.cpp File Reference

```
#include <data_structures.hpp>
```

8.38 src/display/destructor.cpp File Reference

```
#include <display.hpp>
```

8.39 src/graph/destructor.cpp File Reference

```
#include <data_structures/graph.hpp>
```

8.40 src/hash_table/destructor.cpp File Reference

```
#include <data_structures/hash_table.hpp>
```

8.41 src/inputbox/destructor.cpp File Reference

```
#include <inputbox.hpp>
```

8.42 src/maxheap/destructor.cpp File Reference

```
#include <data_structures/maxheap.hpp>
```

8.43 src/minheap/destructor.cpp File Reference

```
#include <data_structures/minheap.hpp>
```

8.44 src/object/destructor.cpp File Reference

```
#include <object.hpp>
```

8.45 src/script/destructor.cpp File Reference

```
#include <script.hpp>
```

8.46 src/sprite/destructor.cpp File Reference

```
#include <sprite.hpp>
```

8.47 src/trie/destructor.cpp File Reference

```
#include <data_structures/trie.hpp>
```

8.48 src/window/destructor.cpp File Reference

```
#include <window.hpp>  
#include <SDL2/SDL_ttf.h>  
#include <SDL2/SDL_image.h>
```

8.49 src/AVL/event.cpp File Reference

```
#include <data_structures/AVL.hpp>
```

8.50 src/button/event.cpp File Reference

```
#include <button.hpp>
```

8.51 src/data_structures/event.cpp File Reference

```
#include "GLOBAL.hpp"  
#include <data_structures.hpp>
```

8.52 src/display/event.cpp File Reference

```
#include <display.hpp>
```

8.53 src/graph/event.cpp File Reference

```
#include <data_structures/graph.hpp>
```

8.54 src/hash_table/event.cpp File Reference

```
#include <data_structures/hash_table.hpp>
```

8.55 src/inputbox/event.cpp File Reference

```
#include <inputbox.hpp>  
#include <iostream>
```

8.56 src/minheap/event.cpp File Reference

```
#include <data_structures/minheap.hpp>
```

8.57 src/script/event.cpp File Reference

```
#include <script.hpp>
```

8.58 src/trie/event.cpp File Reference

```
#include <data_structures/trie.hpp>
```

8.59 src/window/event.cpp File Reference

```
#include <window.hpp>  
#include <services.hpp>
```

8.60 src/AVL/operator/delete.cpp File Reference

```
#include <data_structures/AVL.hpp>  
#include <services.hpp>
```

8.61 src/AVL/operator/init.cpp File Reference

```
#include <data_structures/AVL.hpp>  
#include <services.hpp>
```

8.62 src/btree4th/operator/init.cpp File Reference

```
#include <data_structures/btree4th.hpp>
```

8.63 src/data_structures/operator/AVL/init.cpp File Reference

```
#include <data_structures.hpp>
```

8.64 src/data_structures/operator/graph/init.cpp File Reference

```
#include <data_structures.hpp>
```

8.65 src/data_structures/operator/hash_table/init.cpp File Reference

```
#include <data_structures.hpp>
```

8.66 src/data_structures/operator/minheap/init.cpp File Reference

```
#include <data_structures.hpp>
```

8.67 src/data_structures/operator/trie/init.cpp File Reference

```
#include <data_structures.hpp>
```

8.68 src/graph/operator/init.cpp File Reference

```
#include <data_structures/graph.hpp>
#include <services.hpp>
```

Functions

- double [sqr](#) (double x)

8.68.1 Function Documentation

8.68.1.1 [sqr\(\)](#)

```
double sqr (
    double x )
```

Definition at line 4 of file init.cpp.

```
5 {
6     return x * x;
7 }
```

8.69 src/hash_table/operator/init.cpp File Reference

```
#include <iostream>
#include <data_structures/hash_table.hpp>
#include <services.hpp>
```

8.70 src/maxheap/operator/init.cpp File Reference

```
#include <data_structures/maxheap.hpp>
```

8.71 src/minheap/operator/init.cpp File Reference

```
#include <data_structures/minheap.hpp>
```

8.72 src/trie/operator/init.cpp File Reference

```
#include <data_structures/trie.hpp>
```


8.73 src/AVL/operator/insert.cpp File Reference

```
#include <data_structures.hpp>  
#include <services.hpp>
```

8.74 src/btree4th/operator/insert.cpp File Reference

```
#include <data_structures/btree4th.hpp>
```

8.75 src/data_structures/operator/AVL/insert.cpp File Reference

```
#include <data_structures.hpp>
```

8.76 src/data_structures/operator/hash_table/insert.cpp File Reference

```
#include <data_structures.hpp>
```

8.77 src/data_structures/operator/minheap/insert.cpp File Reference

```
#include <data_structures.hpp>
```

8.78 src/data_structures/operator/trie/insert.cpp File Reference

```
#include <data_structures.hpp>
```

8.79 src/hash_table/operator/insert.cpp File Reference

```
#include <data_structures/hash_table.hpp>  
#include <services.hpp>
```

8.80 src/maxheap/operator/insert.cpp File Reference

```
#include <data_structures/maxheap.hpp>
```

8.81 **src/minheap/operator/insert.cpp** File Reference

```
#include <data_structures/minheap.hpp>
#include <services.hpp>
```

8.82 **src/trie/operator/insert.cpp** File Reference

```
#include <data_structures/trie.hpp>
```

8.83 **src/AVL/operator/search.cpp** File Reference

```
#include <data_structures/AVL.hpp>
#include <services.hpp>
```

8.84 **src/btree4th/operator/search.cpp** File Reference

```
#include <data_structures/btree4th.hpp>
```

8.85 **src/data_structures/operator/AVL/search.cpp** File Reference

```
#include <data_structures.hpp>
```

8.86 **src/data_structures/operator/hash_table/search.cpp** File Reference

```
#include <data_structures.hpp>
```

8.87 **src/data_structures/operator/trie/search.cpp** File Reference

```
#include <data_structures.hpp>
```

8.88 **src/hash_table/operator/search.cpp** File Reference

```
#include <data_structures/hash_table.hpp>
```

8.89 src/trie/operator/search.cpp File Reference

```
#include <data_structures/trie.hpp>
```

8.90 src/AVL/rendering.cpp File Reference

```
#include <data_structures/AVL.hpp>
#include <queue>
```

8.91 src/button/rendering.cpp File Reference

```
#include <button.hpp>
#include <iostream>
```

8.92 src/data_structures/rendering.cpp File Reference

```
#include <data_structures.hpp>
```

8.93 src/display/rendering.cpp File Reference

```
#include <display.hpp>
#include <SDL2/SDL.h>
```

8.94 src/graph/rendering.cpp File Reference

```
#include <data_structures/graph.hpp>
```

Classes

- struct [Point](#)

8.95 src/hash_table/rendering.cpp File Reference

```
#include <data_structures/hash_table.hpp>
```

8.96 src/inputbox/rendering.cpp File Reference

```
#include <inputbox.hpp>
```

8.97 src/maxheap/rendering.cpp File Reference

8.98 src/minheap/rendering.cpp File Reference

```
#include <data_structures/minheap.hpp>  
#include <cmath>
```

8.99 src/object/rendering.cpp File Reference

```
#include <object.hpp>
```

8.100 src/script/rendering.cpp File Reference

```
#include <script.hpp>
```

8.101 src/sprite/rendering.cpp File Reference

```
#include <sprite.hpp>
```

8.102 src/trie/rendering.cpp File Reference

```
#include <data_structures/trie.hpp>  
#include <queue>
```

8.103 src/window/rendering.cpp File Reference

```
#include <window.hpp>  
#include <services.hpp>
```

8.104 src/AVL/rotate.cpp File Reference

```
#include <data_structures/AVL.hpp>
```

8.105 src/AVL/step.cpp File Reference

```
#include <data_structures/AVL.hpp>  
#include <chrono>  
#include <thread>
```

8.106 src/data_structures/step.cpp File Reference

```
#include <data_structures.hpp>
```

8.107 src/hash_table/step.cpp File Reference

```
#include <data_structures/hash_table.hpp>
```

8.108 src/minheap/step.cpp File Reference

```
#include <data_structures/minheap.hpp>  
#include <chrono>  
#include <thread>
```

8.109 src/trie/step.cpp File Reference

```
#include <data_structures/trie.hpp>  
#include <thread>
```

8.110 src/btree4th/node.cpp File Reference

```
#include <data_structures/btree4th.hpp>
```

8.111 `src/btree4th/operator/addRecord.cpp` File Reference

```
#include <data_structures/btree4th.hpp>
```

8.112 `src/btree4th/operator/split.cpp` File Reference

```
#include <data_structures/btree4th.hpp>
```

8.113 `src/button/action.cpp` File Reference

```
#include <button.hpp>
```

8.114 `src/button/mouse_action.cpp` File Reference

```
#include <button.hpp>  
#include <iostream>
```

8.115 `src/data_structures/AVL.cpp` File Reference

8.116 `src/data_structures/btree4th.cpp` File Reference

8.117 `src/data_structures/graph.cpp` File Reference

8.118 `src/data_structures/hash_table.cpp` File Reference

8.119 `src/data_structures/heap.cpp` File Reference

8.120 `src/data_structures/operator.cpp` File Reference

```
#include "GLOBAL.hpp"  
#include <data_structures.hpp>
```

8.121 src/inputbox/operator.cpp File Reference

```
#include <inputbox.hpp>
```

8.122 src/data_structures/operator/AVL/remove.cpp File Reference

```
#include <data_structures.hpp>
```

8.123 src/data_structures/operator/hash_table/remove.cpp File Reference

```
#include <data_structures.hpp>
```

8.124 src/data_structures/operator/trie/remove.cpp File Reference

```
#include <data_structures.hpp>
```

8.125 src/hash_table/operator/remove.cpp File Reference

```
#include <data_structures/hash_table.hpp>
```

8.126 src/maxheap/operator/remove.cpp File Reference

```
#include <data_structures/maxheap.hpp>
```

8.127 src/minheap/operator/remove.cpp File Reference

```
#include <data_structures/minheap.hpp>
```

Variables

- bool `valuesReachedZero` = false

8.127.1 Variable Documentation

8.127.1.1 valuesReachedZero

```
bool valuesReachedZero = false
```

Definition at line 3 of file remove.cpp.

8.128 src/trie/operator/remove.cpp File Reference

```
#include <data_structures/trie.hpp>
```

8.129 src/data_structures/operator/AVL/setting.cpp File Reference

```
#include <data_structures.hpp>  
#include <services.hpp>
```

8.130 src/data_structures/operator/graph/setting.cpp File Reference

```
#include <data_structures.hpp>  
#include <services.hpp>
```

8.131 src/data_structures/operator/hash_table/setting.cpp File Reference

```
#include <data_structures.hpp>  
#include <services.hpp>
```

8.132 src/data_structures/operator/minheap/setting.cpp File Reference

```
#include <data_structures.hpp>  
#include <services.hpp>
```


8.133 src/data_structures/operator/trie/setting.cpp File Reference

```
#include <data_structures.hpp>
#include <services.hpp>
```

8.134 src/data_structures/operator/graph/dijkstra.cpp File Reference

```
#include <data_structures.hpp>
```

8.135 src/graph/operator/dijkstra.cpp File Reference

```
#include <data_structures/graph.hpp>
```

Classes

- struct [distanceHeap](#)

8.136 src/data_structures/operator/graph/mst.cpp File Reference

```
#include <data_structures.hpp>
```

8.137 src/graph/operator/mst.cpp File Reference

```
#include <data_structures/graph.hpp>
#include <algorithm>
```

Classes

- struct [DSU](#)

8.138 src/data_structures/operator/graph/scc.cpp File Reference

```
#include <data_structures.hpp>
```

8.139 src/graph/operator/scc.cpp File Reference

```
#include <data_structures/graph.hpp>
#include <services.hpp>
```

8.140 src/data_structures/operator/minheap/pop.cpp File Reference

```
#include <data_structures.hpp>
```

8.141 src/data_structures/operator/minheap/size.cpp File Reference

8.142 src/maxheap/operator/size.cpp File Reference

```
#include <data_structures/maxheap.hpp>
```

8.143 src/minheap/operator/size.cpp File Reference

```
#include <data_structures/minheap.hpp>
```

8.144 src/data_structures/operator/minheap/top.cpp File Reference

8.145 src/data_structures/trie.cpp File Reference

8.146 src/graph/operator/repair.cpp File Reference

```
#include <data_structures/graph.hpp>
```

8.147 src/inputbox/typing.cpp File Reference

```
#include <inputbox.hpp>
```

8.148 src/main.cpp File Reference

```
#include <iostream>
#include <window.hpp>
```

Functions

- signed `main()`

8.148.1 Function Documentation

8.148.1.1 `main()`

```
signed main ( )
```

Definition at line 4 of file main.cpp.

```
5 {
6     MyWindow window;
7     window.run();
8
9     return 0;
10 }
```

8.149 src/maxheap/operator/getmax.cpp File Reference

```
#include <data_structures/maxheap.hpp>
```

8.150 src/maxheap/operator/heapify.cpp File Reference

```
#include <data_structures/maxheap.hpp>
```

8.151 src/minheap/operator/heapify.cpp File Reference

```
#include <data_structures/minheap.hpp>
```

8.152 src/minheap/operator/getmin.cpp File Reference

```
#include <climits>
#include <data_structures/minheap.hpp>
```

8.153 src/object/coloring.cpp File Reference

```
#include <object.hpp>
```

8.154 src/sprite/coloring.cpp File Reference

```
#include <sprite.hpp>  
#include <iostream>
```

8.155 src/object/cropping.cpp File Reference

```
#include <object.hpp>
```

8.156 src/object/external_storage.cpp File Reference

```
#include <object.hpp>  
#include <SDL2/SDL_image.h>
```

8.157 src/object/font.cpp File Reference

```
#include <object.hpp>  
#include <iostream>
```

8.158 src/object/locating.cpp File Reference

```
#include <object.hpp>
```

8.159 src/sprite/locating.cpp File Reference

```
#include <sprite.hpp>
```

8.160 src/object/shape.cpp File Reference

```
#include <object.hpp>
#include <algorithm>
#include <iostream>
```

8.161 src/object/visible.cpp File Reference

```
#include <object.hpp>
```

8.162 src/script/highlight.cpp File Reference

```
#include <script.hpp>
```

8.163 src/services/file.cpp File Reference

```
#include <services.hpp>
#include <fstream>
#include <iostream>
```

8.164 src/services/json.cpp File Reference

```
#include <services.hpp>
#include <fstream>
#include <iostream>
```

8.165 src/services/number.cpp File Reference

```
#include <services.hpp>
#include <iostream>
```

Functions

- `std::string removeLeadingZero (std::string s)`

8.165.1 Function Documentation

8.165.1.1 removeLeadingZero()

```
std::string removeLeadingZero (  
    std::string s )
```

Definition at line 115 of file number.cpp.

```
116 {  
117     if(!NUMBER::isNumber(s)) return "0";  
118  
119     bool isNegative = (s[0] == '-');  
120     bool isSign = NUMBER::isSign(s[0]);  
121  
122     int i = isSign;  
123     while(s[i] == '0') i++;  
124  
125     if(i == (int) s.size()) return "0";  
126  
127     std::string result = "";  
128     if(isNegative) result += '-';  
129     for(; i < (int) s.size(); i++) result += s[i];  
130  
131     return result;  
132 }
```

8.166 src/services/random.cpp File Reference

```
#include <services.hpp>
```

8.167 src/services/string.cpp File Reference

```
#include <services.hpp>
```

8.168 src/sprite/aligning.cpp File Reference

```
#include <sprite.hpp>  
#include <iostream>
```

8.169 src/sprite/textbox.cpp File Reference

```
#include <sprite.hpp>  
#include <services.hpp>
```

8.170 src/window/running.cpp File Reference

```
#include <window.hpp>  
#include <services.hpp>
```

8.171 src/window/status.cpp File Reference

```
#include <window.hpp>
```

8.172 src/window/updating.cpp File Reference

```
#include <window.hpp>  
#include <sstream>
```


Index

- ~AVL
 - AVL, [31](#)
- ~BTree4th
 - BTree4th, [46](#)
- ~Button
 - Button, [53](#)
- ~DSU
 - DSU, [131](#)
- ~DataStructures
 - DataStructures, [76](#)
- ~Display
 - Display, [110](#)
- ~Graph
 - Graph, [135](#)
- ~HashTable
 - HashTable, [144](#)
- ~InputBox
 - InputBox, [156](#)
- ~MyWindow
 - MyWindow, [178](#)
- ~Object
 - Object, [190](#)
- ~Script
 - Script, [209](#)
- ~Sprite
 - Sprite, [215](#)
- ~Trie
 - Trie, [237](#)
- ~distanceHeap
 - distanceHeap, [128](#)
- ~maxHeap
 - maxHeap, [163](#)
- ~minHeap
 - minHeap, [168](#)
- addCharacter
 - Button, [53](#)
 - DataStructures, [76](#)
 - Display, [111](#)
 - Object, [191](#)
 - Sprite, [216](#)
- addRecordToLeaf
 - BTree4th, [46](#)
- addText
 - Button, [53](#)
 - DataStructures, [76](#)
 - Display, [111](#)
 - Object, [191](#)
 - Sprite, [216](#)
- aligning
 - Sprite, [216](#), [217](#)
- ARRAY
 - GLOBAL.hpp, [257](#)
- ASSETS_
 - PATH, [20](#)
- ATTRIBUTE_
 - PATH, [20](#)
- AVL, [29](#)
 - ~AVL, [31](#)
 - AVL, [30](#)
 - balanceFactor, [31](#)
 - balancing, [31](#)
 - closeScript, [32](#)
 - getHigh, [32](#)
 - GLOBAL.hpp, [256](#)
 - goBack, [32](#)
 - goNext, [33](#)
 - goOff, [33](#)
 - goOn, [33](#)
 - highlight, [33](#)
 - init, [34](#)
 - insert, [34](#), [35](#)
 - isReceiveEvent, [36](#)
 - locating, [36](#)
 - maxDepth, [36](#), [37](#)
 - react, [37](#)
 - remove, [38](#)
 - rendering, [40](#)
 - renderLine, [40](#)
 - rotateLeft, [40](#)
 - rotateRight, [41](#)
 - search, [41](#), [42](#)
 - setEdgesColor, [43](#)
 - setNodeColor, [43](#)
 - setting, [43](#)
 - slowDown, [43](#)
 - speedUp, [44](#)
 - unhighlight, [44](#)
 - unplugSmallest, [44](#)
 - waitForStep, [44](#)
- AVL_
 - PATH::SAVING, [23](#)
- backspace
 - Sprite, [217](#)
- balanceFactor
 - AVL, [31](#)
- balancing
 - AVL, [31](#)
- BOTTOM

- GLOBAL.hpp, 258
- BTree4th, 45
 - ~BTree4th, 46
 - addRecordToLeaf, 46
 - BTree4th, 46
 - init, 47
 - insert, 47, 48
 - leftMost, 48
 - mergeChild, 48
 - remove, 49
 - rightMost, 49
 - search, 49
 - split, 50
- BTREE_4TH
 - GLOBAL.hpp, 256
- BTREE_4TH_
 - PATH::SAVING, 23
- Button, 51
 - ~Button, 53
 - addCharacter, 53
 - addText, 53
 - Button, 53
 - changeToCircle, 54, 55
 - changeToRectangle, 55
 - coloring, 55, 56
 - cropping, 56
 - fillCircleByColor, 57
 - fillRectangleByColor, 57
 - fillWithColor, 58
 - fitTheTexture, 58
 - getAction, 58
 - getColor, 58
 - getCrop, 59
 - getDataType, 59
 - getH, 59
 - getInputType, 59
 - getLocation, 60
 - getNextScreen, 60
 - getSize, 60
 - getText, 60
 - getW, 60
 - getX, 61
 - getY, 61
 - hide, 61
 - importFromJson, 61
 - initAction, 62
 - initBackground, 63
 - initSprites, 63
 - isChooosed, 63
 - isClicked, 64
 - isHover, 64
 - isLieInside, 64, 65
 - isReceiveEvent, 65
 - isVisible, 66
 - linking, 66
 - locating, 66, 67
 - locatingH, 67
 - locatingW, 67
 - locatingX, 67
 - locatingY, 68
 - move, 68
 - moveX, 68
 - moveY, 68
 - noCropping, 69
 - removeCharacter, 69
 - rendering, 69
 - setFont, 70
 - setShape, 70
 - setText, 70
 - show, 71
 - textToTexture, 71
 - textureFromFile, 71
 - zoom, 71
 - zoomH, 72
 - zoomInMiddle, 72
 - zoomW, 72
- BUTTON_
 - PATH::ATB, 21
- BUTTON_ACTION
 - GLOBAL.hpp, 252
- BUTTON_STATUS
 - GLOBAL.hpp, 254
- capacity
 - DataStructures, 108
- CENTER
 - GLOBAL.hpp, 257, 258
- center
 - Position, 207
- CHANGE_SCREEN
 - GLOBAL.hpp, 253
- changeToCircle
 - Button, 54, 55
 - DataStructures, 77, 78
 - Display, 111, 112
 - Object, 191, 192
 - Sprite, 218, 219
- changeToRectangle
 - Button, 55
 - DataStructures, 78
 - Display, 112
 - Object, 192
 - Sprite, 219
- CIRCLE
 - GLOBAL.hpp, 258
- CLICKED
 - GLOBAL.hpp, 254
- CLOSE
 - GLOBAL.hpp, 253
- closeScript
 - AVL, 32
 - DataStructures, 78
 - HashTable, 144
 - minHeap, 168
 - Trie, 237
- coloring
 - Button, 55, 56

- DataStructures, 79
- Display, 112, 113
- Object, 192, 193
- Sprite, 219, 220
- CONNECTED_COMPONENTS
 - GLOBAL.hpp, 253
- crop
 - Position, 207
- cropping
 - Button, 56
 - DataStructures, 79, 80
 - Display, 113, 114
 - Object, 193, 194
 - Sprite, 220
- DATA_STRUCTURES_
 - PATH::ATB, 21
- DATA_STRUCTURES_OPERATOR
 - GLOBAL.hpp, 254
- DATA_STRUCTURES_TYPE
 - GLOBAL.hpp, 256
- DataStructures, 73
 - ~DataStructures, 76
 - addCharacter, 76
 - addText, 76
 - capacity, 108
 - changeToCircle, 77, 78
 - changeToRectangle, 78
 - closeScript, 78
 - coloring, 79
 - cropping, 79, 80
 - DataStructures, 75
 - Dijkstra, 80
 - dijkstra, 80
 - fillCircleByColor, 81
 - fillRectangleByColor, 81
 - fillWithColor, 82
 - fitTheTexture, 82
 - getColor, 82
 - getCrop, 82
 - getDataType, 83
 - getH, 83
 - getLocation, 83
 - getName, 83
 - getSize, 83
 - getText, 84
 - getW, 84
 - getX, 84
 - getY, 84
 - goBack, 84
 - goNext, 85
 - goOff, 85
 - goOn, 85
 - hide, 85
 - importFromJson, 86
 - init, 86
 - initAVL, 87
 - initBackground, 87
 - initDisplay, 87
 - initGraph, 88
 - initHashTable, 88
 - initLinker, 88
 - initMinHeap, 89
 - initTrie, 89
 - insert, 89
 - insertAVL, 89
 - insertHashTable, 90
 - insertMinHeap, 90
 - insertTrie, 90
 - isLieInside, 90, 91
 - isReceiveEvent, 91
 - isVisible, 92
 - linking, 92
 - locating, 92, 93
 - locatingH, 93
 - locatingW, 93
 - locatingX, 93
 - locatingY, 94
 - moveX, 94
 - moveY, 94
 - MST, 94
 - mst, 95
 - noCropping, 95
 - react, 95
 - remove, 95
 - removeAVL, 96
 - removeCharacter, 96
 - removeHashTable, 97
 - removeMinHeap, 97
 - removeTrie, 97
 - rendering, 97
 - SCC, 98
 - scc, 98
 - search, 98
 - searchAVL, 98
 - searchHashTable, 99
 - searchMinHeap, 99
 - searchTrie, 99
 - setDataType, 99
 - setFont, 100
 - setShape, 100
 - setText, 101
 - setting, 101
 - settingAVL, 101
 - settingGraph, 102
 - settingHashTable, 103
 - settingMinHeap, 103
 - settingTrie, 104
 - show, 105
 - size, 105
 - slowDown, 105
 - speedUp, 105
 - textToTexture, 106
 - textureFromFile, 106
 - top, 106
 - zoom, 107
 - zoomH, 107

- zoomInMiddle, 107
 - zoomW, 107
- defaultSetting
 - HashTable, 144
- DELETE
 - GLOBAL.hpp, 253, 256
- DIJKSTRA
 - GLOBAL.hpp, 253, 256
- Dijkstra
 - DataStructures, 80
 - Graph, 135
- dijkstra
 - DataStructures, 80
- DISPLAY, 13
 - HOME_, 13
 - WORKING_, 13
- Display, 108
 - ~Display, 110
 - addCharacter, 111
 - addText, 111
 - changeToCircle, 111, 112
 - changeToRectangle, 112
 - coloring, 112, 113
 - cropping, 113, 114
 - Display, 110
 - fillCircleByColor, 114
 - fillRectangleByColor, 115
 - fillWithColor, 115
 - fitTheTexture, 115
 - getColor, 115
 - getCrop, 116
 - getH, 116
 - getLocation, 116
 - getSize, 116
 - getText, 116
 - getW, 117
 - getX, 117
 - getY, 117
 - hide, 117
 - importFromJson, 117, 118
 - initBackground, 118
 - initButtons, 118
 - isButtonReceiveEvent, 119
 - isLieInside, 119, 120
 - isReceiveEvent, 120
 - isVisible, 120
 - linking, 120
 - locating, 121
 - locatingH, 121
 - locatingW, 122
 - locatingX, 122
 - locatingY, 122
 - moveX, 122
 - moveY, 123
 - noCropping, 123
 - react, 123
 - removeCharacter, 123, 124
 - rendering, 124
 - setFont, 124
 - setShape, 124
 - setText, 125
 - show, 125
 - textToTexture, 125
 - textureFromFile, 126
 - zoom, 126
 - zoomH, 126
 - zoomInMiddle, 126
 - zoomW, 127
- DISPLAY_
 - PATH::ATB, 22
- distanceHeap, 127
 - ~distanceHeap, 128
- distanceHeap, 128
- empty, 128
- g, 130
- Graph, 142
- insert, 128
- pop, 129
- swap, 129
- swappable, 129
- value, 130
- DONE
 - GLOBAL.hpp, 253
- drawEdge
 - HashTable, 145
- drawEdges
 - Trie, 238
- DSU, 130
 - ~DSU, 131
- DSU, 131
- find, 131
- Graph, 142
- isUnionized, 132
- parent, 133
- unionEdge, 132, 133
- EDGES
 - GLOBAL.hpp, 253
- empty
 - distanceHeap, 128
- FILE
 - GLOBAL.hpp, 253
- FILEE, 14
 - readFile, 14
- fillCircleByColor
 - Button, 57
 - DataStructures, 81
 - Display, 114
 - Object, 194
 - Sprite, 221
- fillRectangleByColor
 - Button, 57
 - DataStructures, 81
 - Display, 115
 - Object, 195
 - Sprite, 221

- fillWithColor
 - Button, 58
 - DataStructures, 82
 - Display, 115
 - Object, 195
 - Sprite, 222
- find
 - DSU, 131
- fitTheTexture
 - Button, 58
 - DataStructures, 82
 - Display, 115
 - Object, 195
 - Sprite, 222
- flipCoin
 - RANDOM, 25
- FONTS_
 - PATH::ASSETS, 20
- g
 - distanceHeap, 130
- getAction
 - Button, 58
- getChar
 - RANDOM, 25
- getColor
 - Button, 58
 - DataStructures, 82
 - Display, 115
 - Object, 195
 - Sprite, 222
- getCrop
 - Button, 59
 - DataStructures, 82
 - Display, 116
 - Object, 196
 - Sprite, 222
- getDataFromFile
 - MyWindow, 179
- getDataFromRandom
 - MyWindow, 180
- getDataType
 - Button, 59
 - DataStructures, 83
- getDouble
 - RANDOM, 25
- getEvent
 - MyWindow, 180
- getFloat
 - RANDOM, 25
- getH
 - Button, 59
 - DataStructures, 83
 - Display, 116
 - Object, 196
 - Sprite, 223
- getHigh
 - AVL, 32
- getInputType
 - Button, 59
- getInt
 - RANDOM, 26
- getLocation
 - Button, 60
 - DataStructures, 83
 - Display, 116
 - Object, 196
 - Sprite, 223
- getLongLong
 - RANDOM, 26
- getName
 - DataStructures, 83
- getNextScreen
 - Button, 60
- getOperator
 - InputBox, 157
- getSize
 - Button, 60
 - DataStructures, 83
 - Display, 116
 - Object, 196
 - Sprite, 223
- getString
 - RANDOM, 26, 27
- getText
 - Button, 60
 - DataStructures, 84
 - Display, 116
 - InputBox, 157
 - Object, 196
 - Sprite, 223
- getW
 - Button, 60
 - DataStructures, 84
 - Display, 117
 - Object, 197
 - Sprite, 223
- getX
 - Button, 61
 - DataStructures, 84
 - Display, 117
 - Object, 197
 - Sprite, 224
- getY
 - Button, 61
 - DataStructures, 84
 - Display, 117
 - Object, 197
 - Sprite, 224
- GLOBAL.hpp
 - ARRAY, 257
 - AVL, 256
 - BOTTOM, 258
 - BTREE_4TH, 256
 - BUTTON_ACTION, 252
 - BUTTON_STATUS, 254
 - CENTER, 257, 258

CHANGE_SCREEN, 253
 CIRCLE, 258
 CLICKED, 254
 CLOSE, 253
 CONNECTED_COMPONENTS, 253
 DATA_STRUCTURES_OPERATOR, 254
 DATA_STRUCTURES_TYPE, 256
 DELETE, 253, 256
 DIJKSTRA, 253, 256
 DONE, 253
 EDGES, 253
 FILE, 253
 GO_BACK, 253
 GO_NEXT, 253
 GO_OFF, 253
 GO_ON, 253
 GRAPH, 256
 HASH_TABLE, 256
 HORIZONTAL_ALIGN, 256
 HOVER, 254
 INIT, 253, 256
 INPUT_TYPE, 257
 INSERT, 253, 256
 INT, 257
 IS_CLOSED, 258
 IS_OPEN, 258
 json, 252
 LEFT, 257
 LINE, 258
 MAX_HEAP, 256
 MIN_HEAP, 256
 MST, 253, 256
 NONE, 253, 254, 256, 257
 POLYGON, 258
 RANDOM, 253
 RANDOM10, 253
 RANDOM11, 253
 RANDOM12, 253
 RANDOM13, 253
 RANDOM14, 253
 RANDOM15, 253
 RANDOM16, 253
 RANDOM2, 253
 RANDOM3, 253
 RANDOM4, 253
 RANDOM5, 253
 RANDOM6, 253
 RANDOM7, 253
 RANDOM8, 253
 RANDOM9, 253
 RECTANGLE, 258
 RELEASED, 254
 RIGHT, 257
 SCC, 256
 SEARCH, 253, 256
 SETTING, 253, 256
 SHAPE, 257
 SIZE, 253, 256
 SLOW_DOWN, 253
 SPEED_UP, 253
 STRING, 257
 STRINGS, 257
 TOP, 253, 256, 258
 TRIANGLE, 258
 TRIE, 256
 VERTICAL_ALIGN, 258
 WINDOW_STATUS, 258
 GO_BACK
 GLOBAL.hpp, 253
 GO_NEXT
 GLOBAL.hpp, 253
 GO_OFF
 GLOBAL.hpp, 253
 GO_ON
 GLOBAL.hpp, 253
 goBack
 AVL, 32
 DataStructures, 84
 HashTable, 145
 goNext
 AVL, 33
 DataStructures, 85
 HashTable, 145
 goOff
 AVL, 33
 DataStructures, 85
 HashTable, 145
 goOn
 AVL, 33
 DataStructures, 85
 HashTable, 146
 GRAPH
 GLOBAL.hpp, 256
 Graph, 134
 ~Graph, 135
 Dijkstra, 135
 distanceHeap, 142
 DSU, 142
 Graph, 134
 init, 136
 isReceiveEvent, 137
 MST, 137
 react, 137
 renderEdge, 138
 rendering, 139
 repair, 139
 SCC, 140
 setting, 140
 Tarjan, 140
 unionEdges, 141
 waitForStep, 141
 GRAPH_
 PATH::SAVING, 23
 GRAPHICS_
 PATH::ASSETS, 21
 HASH_TABLE

- GLOBAL.hpp, 256
- HASH_TABLE_
 - PATH::SAVING, 23
- HashTable, 142
 - ~HashTable, 144
 - closeScript, 144
 - defaultSetting, 144
 - drawEdge, 145
 - goBack, 145
 - goNext, 145
 - goOff, 145
 - goOn, 146
 - HashTable, 143
 - highlight, 146
 - init, 146
 - insert, 146, 147
 - isReceiveEvent, 148
 - locating, 148, 149
 - react, 149
 - remove, 150, 151
 - renderEdges, 151
 - rendering, 151
 - search, 152
 - setEdgesColor, 153
 - setNodeColor, 153
 - setting, 153
 - slowDown, 154
 - speedUp, 154
 - unhighlight, 154
 - waitForStep, 154
- heapify
 - maxHeap, 164
 - minHeap, 168
- hide
 - Button, 61
 - DataStructures, 85
 - Display, 117
 - Object, 197
 - Sprite, 224
- highlight
 - AVL, 33
 - HashTable, 146
 - minHeap, 169
 - Script, 209
 - Sprite, 224
 - Trie, 238
- HOME_
 - DISPLAY, 13
- HORIZONTAL_ALIGN
 - GLOBAL.hpp, 256
- HOVER
 - GLOBAL.hpp, 254
- importFromJson
 - Button, 61
 - DataStructures, 86
 - Display, 117, 118
 - InputBox, 157
 - Object, 197
 - Script, 209
 - Sprite, 224, 225
- include/button.hpp, 247
- include/data_structures.hpp, 247
- include/data_structures/AVL.hpp, 248
- include/data_structures/btree4th.hpp, 248
- include/data_structures/graph.hpp, 248
- include/data_structures/hash_table.hpp, 249
- include/data_structures/maxheap.hpp, 249
- include/data_structures/minheap.hpp, 249
- include/data_structures/trie.hpp, 250
- include/display.hpp, 250
- include/GLOBAL.hpp, 250
- include/inputbox.hpp, 259
- include/object.hpp, 259
- include/script.hpp, 260
- include/services.hpp, 260
- include/sprite.hpp, 262
- include/window.hpp, 262
- INF
 - NUMBER, 19
- INIT
 - GLOBAL.hpp, 253, 256
- init
 - AVL, 34
 - BTree4th, 47
 - DataStructures, 86
 - Graph, 136
 - HashTable, 146
 - maxHeap, 164
 - minHeap, 169
 - Trie, 238
- init.cpp
 - sqr, 270
- initAction
 - Button, 62
- initAVL
 - DataStructures, 87
- initBackground
 - Button, 63
 - DataStructures, 87
 - Display, 118
 - InputBox, 157
 - Script, 210
 - Sprite, 225
- initButtons
 - Display, 118
 - InputBox, 158
 - Script, 210
- initDisplay
 - DataStructures, 87
- initGraph
 - DataStructures, 88
- initHashTable
 - DataStructures, 88
- initInput
 - Sprite, 225
- initLinker

- DataStructures, 88
- initMinHeap
 - DataStructures, 89
- initObjects
 - Sprite, 226
- initOperator
 - InputBox, 158
- initSDL2
 - MyWindow, 181
- initSprites
 - Button, 63
 - InputBox, 158
- initTextBox
 - Sprite, 226
- initTrie
 - DataStructures, 89
- INPUT_TYPE
 - GLOBAL.hpp, 257
- InputBox, 155
 - ~InputBox, 156
 - getOperator, 157
 - getText, 157
 - importFromJson, 157
 - initBackground, 157
 - initButtons, 158
 - initOperator, 158
 - initSprites, 158
 - InputBox, 156
 - isButtonReceiveEvent, 159
 - isInputReceiveEvent, 159
 - isReceiveEvent, 160
 - linking, 160
 - m, 162
 - n, 162
 - react, 160
 - rendering, 161
 - setDuplicate, 161
 - setText, 161, 162
- INPUTBOX_
 - PATH::ATB, 22
- INSERT
 - GLOBAL.hpp, 253, 256
- insert
 - AVL, 34, 35
 - BTree4th, 47, 48
 - DataStructures, 89
 - distanceHeap, 128
 - HashTable, 146, 147
 - maxHeap, 164
 - minHeap, 170
 - Trie, 238, 240
- insertAVL
 - DataStructures, 89
- insertHashTable
 - DataStructures, 90
- insertMinHeap
 - DataStructures, 90
- insertTrie
 - DataStructures, 90
- INT
 - GLOBAL.hpp, 257
- interacting
 - MyWindow, 182
- intToString
 - NUMBER, 16
- IS_CLOSED
 - GLOBAL.hpp, 258
- IS_OPEN
 - GLOBAL.hpp, 258
- isButtonReceiveEvent
 - Display, 119
 - InputBox, 159
- isChooosed
 - Button, 63
- isClicked
 - Button, 64
- isClosed
 - MyWindow, 182
- isDigit
 - NUMBER, 16
- isHover
 - Button, 64
- isInInterval
 - NUMBER, 16
- isInputReceiveEvent
 - InputBox, 159
- isLetter
 - NUMBER, 17
- isLieInside
 - Button, 64, 65
 - DataStructures, 90, 91
 - Display, 119, 120
 - Object, 198, 199
 - Sprite, 226, 227
- isNumber
 - NUMBER, 17
- isOpen
 - MyWindow, 182
- isOperator
 - NUMBER, 17
- isReceiveEvent
 - AVL, 36
 - Button, 65
 - DataStructures, 91
 - Display, 120
 - Graph, 137
 - HashTable, 148
 - InputBox, 160
 - minHeap, 170
 - Script, 210
 - Trie, 240
- isSeparator
 - SIUSTRING, 28
- isSign
 - NUMBER, 17
- isSymbol

- NUMBER, 18
- isUnionized
 - DSU, 132
- isVisible
 - Button, 66
 - DataStructures, 92
 - Display, 120
 - Object, 199
 - Sprite, 227
- JSON, 14
 - readFile, 15
 - saveFile, 15
- json
 - GLOBAL.hpp, 252
 - object.hpp, 259
 - services.hpp, 261
 - sprite.hpp, 262
- LEFT
 - GLOBAL.hpp, 257
- leftMost
 - BTree4th, 48
- LINE
 - GLOBAL.hpp, 258
- linking
 - Button, 66
 - DataStructures, 92
 - Display, 120
 - InputDialog, 160
 - Object, 199
 - Script, 211
 - Sprite, 228
- locating
 - AVL, 36
 - Button, 66, 67
 - DataStructures, 92, 93
 - Display, 121
 - HashTable, 148, 149
 - minHeap, 170
 - Object, 199, 200
 - Sprite, 228
 - Trie, 240
- locatingH
 - Button, 67
 - DataStructures, 93
 - Display, 121
 - Object, 200
 - Sprite, 229
- locatingW
 - Button, 67
 - DataStructures, 93
 - Display, 122
 - Object, 200
 - Sprite, 229
- locatingX
 - Button, 67
 - DataStructures, 93
 - Display, 122
- Object, 201
- Sprite, 229
- locatingY
 - Button, 68
 - DataStructures, 94
 - Display, 122
 - Object, 201
 - Sprite, 229
- location
 - Position, 207
- m
 - InputDialog, 162
- main
 - main.cpp, 281
- main.cpp
 - main, 281
- MAX_HEAP
 - GLOBAL.hpp, 256
- MAX_HEAP_
 - PATH::SAVING, 23
- maxDepth
 - AVL, 36, 37
- maxHeap, 163
 - ~maxHeap, 163
 - heapify, 164
 - init, 164
 - insert, 164
 - maxHeap, 163
 - pop, 165
 - size, 165
 - swap, 165
 - swapable, 165
 - top, 166
- mergeChild
 - BTree4th, 48
- MIN_HEAP
 - GLOBAL.hpp, 256
- MIN_HEAP_
 - PATH::SAVING, 24
- minHeap, 166
 - ~minHeap, 168
 - closeScript, 168
 - heapify, 168
 - highlight, 169
 - init, 169
 - insert, 170
 - isReceiveEvent, 170
 - locating, 170
 - minHeap, 167
 - pop, 171
 - react, 172
 - realInsert, 172
 - rendering, 173
 - renderLine, 174
 - setMax, 174
 - setmin, 174
 - setting, 174
 - size, 175

- swap, [175](#)
 - swapable, [175](#)
 - top, [175](#)
 - unhighlight, [176](#)
 - waitForStep, [176](#)
- move
 - Button, [68](#)
- moveX
 - Button, [68](#)
 - DataStructures, [94](#)
 - Display, [122](#)
 - Object, [201](#)
 - Sprite, [230](#)
- moveY
 - Button, [68](#)
 - DataStructures, [94](#)
 - Display, [123](#)
 - Object, [201](#)
 - Sprite, [230](#)
- MST
 - DataStructures, [94](#)
 - GLOBAL.hpp, [253](#), [256](#)
 - Graph, [137](#)
- mst
 - DataStructures, [95](#)
- MyWindow, [177](#)
 - ~MyWindow, [178](#)
 - getDataFromFile, [179](#)
 - getDataFromRandom, [180](#)
 - getEvent, [180](#)
 - initSDL2, [181](#)
 - interacting, [182](#)
 - isClosed, [182](#)
 - isOpen, [182](#)
 - MyWindow, [178](#)
 - react, [183](#)
 - rendering, [185](#)
 - run, [186](#)
 - runOperator, [186](#)
 - setDataType, [187](#)
 - setDisplay, [187](#)
 - setInputBox, [187](#)
- n
 - InputBox, [162](#)
- noCropping
 - Button, [69](#)
 - DataStructures, [95](#)
 - Display, [123](#)
 - Object, [202](#)
 - Sprite, [230](#)
- NONE
 - GLOBAL.hpp, [253](#), [254](#), [256](#), [257](#)
- NUMBER, [15](#)
 - INF, [19](#)
 - intToString, [16](#)
 - isDigit, [16](#)
 - isInInterval, [16](#)
 - isLetter, [17](#)
 - isNumber, [17](#)
 - isOperator, [17](#)
 - isSign, [17](#)
 - isSymbol, [18](#)
 - removeLeadingZero, [18](#)
 - stringToArray, [18](#)
 - stringToInt, [18](#)
- number.cpp
 - removeLeadingZero, [284](#)
- Object, [188](#)
 - ~Object, [190](#)
 - addCharacter, [191](#)
 - addText, [191](#)
 - changeToCircle, [191](#), [192](#)
 - changeToRectangle, [192](#)
 - coloring, [192](#), [193](#)
 - cropping, [193](#), [194](#)
 - fillCircleByColor, [194](#)
 - fillRectangleByColor, [195](#)
 - fillWithColor, [195](#)
 - fitTheTexture, [195](#)
 - getColor, [195](#)
 - getCrop, [196](#)
 - getH, [196](#)
 - getLocation, [196](#)
 - getSize, [196](#)
 - getText, [196](#)
 - getW, [197](#)
 - getX, [197](#)
 - getY, [197](#)
 - hide, [197](#)
 - importFromJson, [197](#)
 - isLieInside, [198](#), [199](#)
 - isVisible, [199](#)
 - linking, [199](#)
 - locating, [199](#), [200](#)
 - locatingH, [200](#)
 - locatingW, [200](#)
 - locatingX, [201](#)
 - locatingY, [201](#)
 - moveX, [201](#)
 - moveY, [201](#)
 - noCropping, [202](#)
 - Object, [190](#)
 - removeCharacter, [202](#)
 - rendering, [202](#)
 - setFont, [203](#)
 - setShape, [203](#)
 - setText, [203](#)
 - show, [203](#)
 - textToTexture, [204](#)
 - textureFromFile, [204](#)
 - zoom, [204](#)
 - zoomH, [204](#)
 - zoomInMiddle, [205](#)
 - zoomW, [205](#)
- object.hpp
 - json, [259](#)

OBJECT_
 PATH::ATB, 22

parent
 DSU, 133

PATH, 19
 ASSETS_, 20
 ATTRIBUTE_, 20
 SAVING_, 20

PATH::ASSETS, 20
 FONTS_, 20
 GRAPHICS_, 21
 SCRIPT_, 21

PATH::ATB, 21
 BUTTON_, 21
 DATA_STRUCTURES_, 21
 DISPLAY_, 22
 INPUTBOX_, 22
 OBJECT_, 22
 SCRIPT_, 22
 SPRITE_, 22

PATH::SAVING, 23
 AVL_, 23
 BTREE_4TH_, 23
 GRAPH_, 23
 HASH_TABLE_, 23
 MAX_HEAP_, 23
 MIN_HEAP_, 24
 TRIE_, 24

Point, 206
 x, 206
 y, 206

POLYGON
 GLOBAL.hpp, 258

pop
 distanceHeap, 129
 maxHeap, 165
 minHeap, 171

Position, 206
 center, 207
 crop, 207
 location, 207
 radius, 207
 visible, 207

radius
 Position, 207

RANDOM, 24
 flipCoin, 25
 getChar, 25
 getDouble, 25
 getFloat, 25
 getInt, 26
 getLongLong, 26
 getString, 26, 27
 GLOBAL.hpp, 253
 rng, 27

RANDOM10
 GLOBAL.hpp, 253

RANDOM11
 GLOBAL.hpp, 253

RANDOM12
 GLOBAL.hpp, 253

RANDOM13
 GLOBAL.hpp, 253

RANDOM14
 GLOBAL.hpp, 253

RANDOM15
 GLOBAL.hpp, 253

RANDOM16
 GLOBAL.hpp, 253

RANDOM2
 GLOBAL.hpp, 253

RANDOM3
 GLOBAL.hpp, 253

RANDOM4
 GLOBAL.hpp, 253

RANDOM5
 GLOBAL.hpp, 253

RANDOM6
 GLOBAL.hpp, 253

RANDOM7
 GLOBAL.hpp, 253

RANDOM8
 GLOBAL.hpp, 253

RANDOM9
 GLOBAL.hpp, 253

react
 AVL, 37
 DataStructures, 95
 Display, 123
 Graph, 137
 HashTable, 149
 InputBox, 160
 minHeap, 172
 MyWindow, 183
 Script, 211
 Trie, 241

readFile
 FILEE, 14
 JSON, 15

README.md, 263

realInsert
 minHeap, 172

RECTANGLE
 GLOBAL.hpp, 258

RELEASED
 GLOBAL.hpp, 254

remove
 AVL, 38
 BTree4th, 49
 DataStructures, 95
 HashTable, 150, 151
 Trie, 242, 243

remove.cpp
 valuesReachedZero, 278

removeAVL

- DataStructures, 96
- removeCharacter
 - Button, 69
 - DataStructures, 96
 - Display, 123, 124
 - Object, 202
 - Sprite, 230, 231
- removeHashTable
 - DataStructures, 97
- removeLeadingZero
 - NUMBER, 18
 - number.cpp, 284
- removeMinHeap
 - DataStructures, 97
- removeTrie
 - DataStructures, 97
- renderEdge
 - Graph, 138
- renderEdges
 - HashTable, 151
- rendering
 - AVL, 40
 - Button, 69
 - DataStructures, 97
 - Display, 124
 - Graph, 139
 - HashTable, 151
 - InputBox, 161
 - minHeap, 173
 - MyWindow, 185
 - Object, 202
 - Script, 212
 - Sprite, 231
 - Trie, 243
- renderLine
 - AVL, 40
 - minHeap, 174
- repair
 - Graph, 139
- RIGHT
 - GLOBAL.hpp, 257
- rightMost
 - BTree4th, 49
- rng
 - RANDOM, 27
- rotateLeft
 - AVL, 40
- rotateRight
 - AVL, 41
- run
 - MyWindow, 186
- runOperator
 - MyWindow, 186
- saveFile
 - JSON, 15
- SAVING_
 - PATH, 20
- SCC
 - DataStructures, 98
 - GLOBAL.hpp, 256
 - Graph, 140
- scc
 - DataStructures, 98
- Script, 208
 - ~Script, 209
 - highlight, 209
 - importFromJson, 209
 - initBackground, 210
 - initButtons, 210
 - isReceiveEvent, 210
 - linking, 211
 - react, 211
 - rendering, 212
 - Script, 209
 - unhighlight, 212
- SCRIPT_
 - PATH::ASSETS, 21
 - PATH::ATB, 22
- SEARCH
 - GLOBAL.hpp, 253, 256
- search
 - AVL, 41, 42
 - BTree4th, 49
 - DataStructures, 98
 - HashTable, 152
 - Trie, 244
- searchAVL
 - DataStructures, 98
- searchHashTable
 - DataStructures, 99
- searchMinHeap
 - DataStructures, 99
- searchTrie
 - DataStructures, 99
- services.hpp
 - json, 261
- setDataType
 - DataStructures, 99
 - MyWindow, 187
- setDisplay
 - MyWindow, 187
- setDuplicate
 - InputBox, 161
- setEdgesColor
 - AVL, 43
 - HashTable, 153
 - Trie, 245
- setFont
 - Button, 70
 - DataStructures, 100
 - Display, 124
 - Object, 203
 - Sprite, 231
- setFontColor
 - Sprite, 231
- setInputBox

- MyWindow, 187
- setMax
 - minHeap, 174
- setmin
 - minHeap, 174
- setNodeColor
 - AVL, 43
 - HashTable, 153
 - Trie, 245
- setShape
 - Button, 70
 - DataStructures, 100
 - Display, 124
 - Object, 203
 - Sprite, 232
- setText
 - Button, 70
 - DataStructures, 101
 - Display, 125
 - InputBox, 161, 162
 - Object, 203
 - Sprite, 232
- setTextBox
 - Sprite, 232
- setTextBoxTransparent
 - Sprite, 233
- SETTING
 - GLOBAL.hpp, 253, 256
- setting
 - AVL, 43
 - DataStructures, 101
 - Graph, 140
 - HashTable, 153
 - minHeap, 174
 - Trie, 245
- settingAVL
 - DataStructures, 101
- settingGraph
 - DataStructures, 102
- settingHashTable
 - DataStructures, 103
- settingMinHeap
 - DataStructures, 103
- settingTrie
 - DataStructures, 104
- SHAPE
 - GLOBAL.hpp, 257
- show
 - Button, 71
 - DataStructures, 105
 - Display, 125
 - Object, 203
 - Sprite, 233
- SIUSTRING, 27
 - isSeparator, 28
 - split, 28
- SIZE
 - GLOBAL.hpp, 253, 256
- size
 - DataStructures, 105
 - maxHeap, 165
 - minHeap, 175
- SLOW_DOWN
 - GLOBAL.hpp, 253
- slowDown
 - AVL, 43
 - DataStructures, 105
 - HashTable, 154
- SPEED_UP
 - GLOBAL.hpp, 253
- speedUp
 - AVL, 44
 - DataStructures, 105
 - HashTable, 154
- split
 - BTree4th, 50
 - SIUSTRING, 28
- Sprite, 213
 - ~Sprite, 215
 - addCharacter, 216
 - addText, 216
 - aligning, 216, 217
 - backspace, 217
 - changeToCircle, 218, 219
 - changeToRectangle, 219
 - coloring, 219, 220
 - cropping, 220
 - fillCircleByColor, 221
 - fillRectangleByColor, 221
 - fillWithColor, 222
 - fitTheTexture, 222
 - getColor, 222
 - getCrop, 222
 - getH, 223
 - getLocation, 223
 - getSize, 223
 - getText, 223
 - getW, 223
 - getX, 224
 - getY, 224
 - hide, 224
 - highlight, 224
 - importFromJson, 224, 225
 - initBackground, 225
 - initInput, 225
 - initObjects, 226
 - initTextBox, 226
 - isLieInside, 226, 227
 - isVisible, 227
 - linking, 228
 - locating, 228
 - locatingH, 229
 - locatingW, 229
 - locatingX, 229
 - locatingY, 229
 - moveX, 230

- moveY, [230](#)
- noCropping, [230](#)
- removeCharacter, [230](#), [231](#)
- rendering, [231](#)
- setFont, [231](#)
- setFontColor, [231](#)
- setShape, [232](#)
- setText, [232](#)
- setTextBox, [232](#)
- setTextBoxTransparent, [233](#)
- show, [233](#)
- Sprite, [215](#)
- textToTexture, [233](#)
- textureFromFile, [233](#)
- typing, [234](#)
- unhighlight, [234](#)
- zoom, [234](#)
- zoomH, [235](#)
- zoomInMiddle, [235](#)
- zoomW, [235](#)
- sprite.hpp
 - json, [262](#)
- SPRITE_
 - PATH::ATB, [22](#)
- sqr
 - init.cpp, [270](#)
- src/AVL/constructor.cpp, [263](#)
- src/AVL/destructor.cpp, [265](#)
- src/AVL/event.cpp, [267](#)
- src/AVL/operator/delete.cpp, [269](#)
- src/AVL/operator/init.cpp, [269](#)
- src/AVL/operator/insert.cpp, [271](#)
- src/AVL/operator/search.cpp, [272](#)
- src/AVL/rendering.cpp, [273](#)
- src/AVL/rotate.cpp, [275](#)
- src/AVL/step.cpp, [275](#)
- src/btree4th/constructor.cpp, [263](#)
- src/btree4th/destructor.cpp, [265](#)
- src/btree4th/node.cpp, [275](#)
- src/btree4th/operator/addRecord.cpp, [276](#)
- src/btree4th/operator/init.cpp, [269](#)
- src/btree4th/operator/insert.cpp, [271](#)
- src/btree4th/operator/search.cpp, [272](#)
- src/btree4th/operator/split.cpp, [276](#)
- src/button/action.cpp, [276](#)
- src/button/constructor.cpp, [263](#)
- src/button/destructor.cpp, [265](#)
- src/button/event.cpp, [267](#)
- src/button/mouse_action.cpp, [276](#)
- src/button/rendering.cpp, [273](#)
- src/data_structures/AVL.cpp, [276](#)
- src/data_structures/btree4th.cpp, [276](#)
- src/data_structures/constructor.cpp, [263](#)
- src/data_structures/destructor.cpp, [265](#)
- src/data_structures/event.cpp, [267](#)
- src/data_structures/graph.cpp, [276](#)
- src/data_structures/hash_table.cpp, [276](#)
- src/data_structures/heap.cpp, [276](#)
- src/data_structures/operator.cpp, [276](#)
- src/data_structures/operator/AVL/init.cpp, [269](#)
- src/data_structures/operator/AVL/insert.cpp, [271](#)
- src/data_structures/operator/AVL/remove.cpp, [277](#)
- src/data_structures/operator/AVL/search.cpp, [272](#)
- src/data_structures/operator/AVL/setting.cpp, [278](#)
- src/data_structures/operator/graph/dijkstra.cpp, [279](#)
- src/data_structures/operator/graph/init.cpp, [269](#)
- src/data_structures/operator/graph/mst.cpp, [279](#)
- src/data_structures/operator/graph/scc.cpp, [279](#)
- src/data_structures/operator/graph/setting.cpp, [278](#)
- src/data_structures/operator/hash_table/init.cpp, [269](#)
- src/data_structures/operator/hash_table/insert.cpp, [271](#)
- src/data_structures/operator/hash_table/remove.cpp, [277](#)
- src/data_structures/operator/hash_table/search.cpp, [272](#)
- src/data_structures/operator/hash_table/setting.cpp, [278](#)
- src/data_structures/operator/minheap/init.cpp, [269](#)
- src/data_structures/operator/minheap/insert.cpp, [271](#)
- src/data_structures/operator/minheap/pop.cpp, [280](#)
- src/data_structures/operator/minheap/setting.cpp, [278](#)
- src/data_structures/operator/minheap/size.cpp, [280](#)
- src/data_structures/operator/minheap/top.cpp, [280](#)
- src/data_structures/operator/trie/init.cpp, [269](#)
- src/data_structures/operator/trie/insert.cpp, [271](#)
- src/data_structures/operator/trie/remove.cpp, [277](#)
- src/data_structures/operator/trie/search.cpp, [272](#)
- src/data_structures/operator/trie/setting.cpp, [279](#)
- src/data_structures/rendering.cpp, [273](#)
- src/data_structures/step.cpp, [275](#)
- src/data_structures/trie.cpp, [280](#)
- src/display/constructor.cpp, [263](#)
- src/display/destructor.cpp, [266](#)
- src/display/event.cpp, [267](#)
- src/display/rendering.cpp, [273](#)
- src/graph/constructor.cpp, [263](#)
- src/graph/destructor.cpp, [266](#)
- src/graph/event.cpp, [268](#)
- src/graph/operator/dijkstra.cpp, [279](#)
- src/graph/operator/init.cpp, [270](#)
- src/graph/operator/mst.cpp, [279](#)
- src/graph/operator/repair.cpp, [280](#)
- src/graph/operator/scc.cpp, [280](#)
- src/graph/rendering.cpp, [273](#)
- src/hash_table/constructor.cpp, [264](#)
- src/hash_table/destructor.cpp, [266](#)
- src/hash_table/event.cpp, [268](#)
- src/hash_table/operator/init.cpp, [270](#)
- src/hash_table/operator/insert.cpp, [271](#)
- src/hash_table/operator/remove.cpp, [277](#)
- src/hash_table/operator/search.cpp, [272](#)
- src/hash_table/rendering.cpp, [273](#)
- src/hash_table/step.cpp, [275](#)
- src/inputbox/constructor.cpp, [264](#)
- src/inputbox/destructor.cpp, [266](#)
- src/inputbox/event.cpp, [268](#)

- src/inputbox/operator.cpp, 277
- src/inputbox/rendering.cpp, 274
- src/inputbox/typing.cpp, 280
- src/main.cpp, 281
- src/maxheap/constructor.cpp, 264
- src/maxheap/destructor.cpp, 266
- src/maxheap/operator/getmax.cpp, 281
- src/maxheap/operator/heapify.cpp, 281
- src/maxheap/operator/init.cpp, 270
- src/maxheap/operator/insert.cpp, 271
- src/maxheap/operator/remove.cpp, 277
- src/maxheap/operator/size.cpp, 280
- src/maxheap/rendering.cpp, 274
- src/minheap/constructor.cpp, 264
- src/minheap/destructor.cpp, 266
- src/minheap/event.cpp, 268
- src/minheap/operator/getmin.cpp, 281
- src/minheap/operator/heapify.cpp, 281
- src/minheap/operator/init.cpp, 270
- src/minheap/operator/insert.cpp, 272
- src/minheap/operator/remove.cpp, 277
- src/minheap/operator/size.cpp, 280
- src/minheap/rendering.cpp, 274
- src/minheap/step.cpp, 275
- src/object/coloring.cpp, 282
- src/object/constructor.cpp, 264
- src/object/cropping.cpp, 282
- src/object/destructor.cpp, 266
- src/object/external_storage.cpp, 282
- src/object/font.cpp, 282
- src/object/locating.cpp, 282
- src/object/rendering.cpp, 274
- src/object/shape.cpp, 283
- src/object/visible.cpp, 283
- src/script/constructor.cpp, 264
- src/script/destructor.cpp, 266
- src/script/event.cpp, 268
- src/script/highlight.cpp, 283
- src/script/rendering.cpp, 274
- src/services/file.cpp, 283
- src/services/json.cpp, 283
- src/services/number.cpp, 283
- src/services/random.cpp, 284
- src/services/string.cpp, 284
- src/sprite/aligning.cpp, 284
- src/sprite/coloring.cpp, 282
- src/sprite/constructor.cpp, 265
- src/sprite/destructor.cpp, 267
- src/sprite/locating.cpp, 282
- src/sprite/rendering.cpp, 274
- src/sprite/textbox.cpp, 284
- src/trie/constructor.cpp, 265
- src/trie/destructor.cpp, 267
- src/trie/event.cpp, 268
- src/trie/operator/init.cpp, 270
- src/trie/operator/insert.cpp, 272
- src/trie/operator/remove.cpp, 278
- src/trie/operator/search.cpp, 273
- src/trie/rendering.cpp, 274
- src/trie/step.cpp, 275
- src/window/constructor.cpp, 265
- src/window/destructor.cpp, 267
- src/window/event.cpp, 268
- src/window/rendering.cpp, 274
- src/window/running.cpp, 285
- src/window/status.cpp, 285
- src/window/updating.cpp, 285
- STRING
 - GLOBAL.hpp, 257
- STRINGS
 - GLOBAL.hpp, 257
- stringToArray
 - NUMBER, 18
- stringToInt
 - NUMBER, 18
- swap
 - distanceHeap, 129
 - maxHeap, 165
 - minHeap, 175
- swappable
 - distanceHeap, 129
 - maxHeap, 165
 - minHeap, 175
- Tarjan
 - Graph, 140
- textToTexture
 - Button, 71
 - DataStructures, 106
 - Display, 125
 - Object, 204
 - Sprite, 233
- textureFromFile
 - Button, 71
 - DataStructures, 106
 - Display, 126
 - Object, 204
 - Sprite, 233
- TOP
 - GLOBAL.hpp, 253, 256, 258
- top
 - DataStructures, 106
 - maxHeap, 166
 - minHeap, 175
- TRIANGLE
 - GLOBAL.hpp, 258
- TRIE
 - GLOBAL.hpp, 256
- Trie, 236
 - ~Trie, 237
 - closeScript, 237
 - drawEdges, 238
 - highlight, 238
 - init, 238
 - insert, 238, 240
 - isReceiveEvent, 240
 - locating, 240

- react, [241](#)
- remove, [242](#), [243](#)
- rendering, [243](#)
- search, [244](#)
- setEdgesColor, [245](#)
- setNodeColor, [245](#)
- setting, [245](#)
- Trie, [236](#)
- unhighlight, [245](#)
- waitForStep, [246](#)
- TRIE_
 - PATH::SAVING, [24](#)
- typing
 - Sprite, [234](#)
- unhighlight
 - AVL, [44](#)
 - HashTable, [154](#)
 - minHeap, [176](#)
 - Script, [212](#)
 - Sprite, [234](#)
 - Trie, [245](#)
- unionEdge
 - DSU, [132](#), [133](#)
- unionEdges
 - Graph, [141](#)
- unplugSmallest
 - AVL, [44](#)
- value
 - distanceHeap, [130](#)
- valuesReachedZero
 - remove.cpp, [278](#)
- VERTICAL_ALIGN
 - GLOBAL.hpp, [258](#)
- visible
 - Position, [207](#)
- waitForStep
 - AVL, [44](#)
 - Graph, [141](#)
 - HashTable, [154](#)
 - minHeap, [176](#)
 - Trie, [246](#)
- WINDOW_STATUS
 - GLOBAL.hpp, [258](#)
- WORKING_
 - DISPLAY, [13](#)
- x
 - Point, [206](#)
- y
 - Point, [206](#)
- zoom
 - Button, [71](#)
 - DataStructures, [107](#)
 - Display, [126](#)
 - Object, [204](#)
 - Sprite, [234](#)
- zoomH
 - Button, [72](#)
 - DataStructures, [107](#)
 - Display, [126](#)
 - Object, [204](#)
 - Sprite, [235](#)
- zoomInMiddle
 - Button, [72](#)
 - DataStructures, [107](#)
 - Display, [126](#)
 - Object, [205](#)
 - Sprite, [235](#)
- zoomW
 - Button, [72](#)
 - DataStructures, [107](#)
 - Display, [127](#)
 - Object, [205](#)
 - Sprite, [235](#)