

CircuitSim

Gerado por Doxygen 1.9.1

1 Descrição	1
2 Índice dos Módulos	3
2.1 Módulos	3
3 Namespaces	5
3.1 Lista de Namespaces	5
4 Índice Hierárquico	7
4.1 Hierarquia de Classes	7
5 Índice dos Componentes	9
5.1 Lista de Classes	9
6 Índice dos Arquivos	11
6.1 Lista de Arquivos	11
7 Módulos	13
7.1 Cores padrão	13
7.1.1 Descrição detalhada	13
7.1.2 Definições e macros	13
7.1.2.1 DEFAULT_BGC	13
7.1.2.2 DEFAULT_CC	14
7.1.2.3 DEFAULT_LC	14
7.1.2.4 DEFAULT_SC	14
7.2 Tamanho dos Componentes	14
7.2.1 Descrição detalhada	14
7.2.2 Definições e macros	14
7.2.2.1 HEIGHT	14
7.2.2.2 WIDTH	14
8 Namespace	15
8.1 Referência do Namespace CCT	15
8.2 Referência do Namespace CMP	15
8.2.1 Enumerações	15
8.2.1.1 type	15
8.3 Referência do Namespace GRF	16
8.4 Referência do Namespace NM	16
9 Classes	17
9.1 Referência da Classe GRF::adjacencyMatrix	17
9.1.1 Descrição detalhada	17
9.1.2 Construtores e Destrutores	18
9.1.2.1 adjacencyMatrix() [1/2]	18
9.1.2.2 adjacencyMatrix() [2/2]	18

9.1.3 Funções membros	18
9.1.3.1 getVertexNumber()	18
9.1.3.2 insertEdge()	19
9.1.3.3 insertVertex()	19
9.1.3.4 query()	19
9.1.3.5 removeVertex()	20
9.1.4 Atributos	20
9.1.4.1 adjMatrix	20
9.1.4.2 vertexNumber	20
9.2 Referência da Classe CCT::Circuit	21
9.2.1 Descrição detalhada	22
9.2.2 Construtores e Destrutores	22
9.2.2.1 Circuit()	22
9.2.2.2 ~Circuit()	22
9.2.3 Funções membros	22
9.2.3.1 addComponent()	22
9.2.3.2 editComponent() [1/2]	23
9.2.3.3 editComponent() [2/2]	23
9.2.3.4 getComponentLabel()	24
9.2.3.5 getCurrent()	24
9.2.3.6 getVoltage()	24
9.2.3.7 initialize()	25
9.2.3.8 removeComponent() [1/2]	25
9.2.3.9 removeComponent() [2/2]	26
9.2.3.10 reset()	26
9.2.3.11 Solve()	26
9.2.3.12 updateComponents()	26
9.2.4 Atributos	27
9.2.4.1 chords	27
9.2.4.2 circuitMatrix	27
9.2.4.3 components	27
9.3 Referência da Classe CMP::Component	27
9.3.1 Descrição detalhada	28
9.3.2 Construtores e Destrutores	28
9.3.2.1 Component()	28
9.3.2.2 ~Component()	29
9.3.3 Funções membros	29
9.3.3.1 getCurrent()	29
9.3.3.2 getLabel()	29
9.3.3.3 getNodes()	30
9.3.3.4 getType()	30
9.3.3.5 getVoltage()	30

9.3.3.6 setCurrent()	30
9.3.3.7 setLabel()	31
9.3.3.8 setVoltage()	31
9.3.4 Atributos	32
9.3.4.1 current	32
9.3.4.2 label	32
9.3.4.3 voltage	32
9.3.4.4 vtxs	32
9.4 Referência da Classe Diagram	33
9.4.1 Descrição detalhada	36
9.4.2 Construtores e Destrutores	36
9.4.2.1 Diagram()	36
9.4.3 Funções membros	36
9.4.3.1 clickedControl()	36
9.4.3.2 edit	37
9.4.3.3 editMode	37
9.4.3.4 getBGColor()	37
9.4.3.5 getComponentColor()	38
9.4.3.6 getFileName()	38
9.4.3.7 getGridColor()	38
9.4.3.8 getPixMap()	38
9.4.3.9 getSelectedColor()	39
9.4.3.10 getStatus()	39
9.4.3.11 initializeDiagram()	39
9.4.3.12 insert()	39
9.4.3.13 leftButtonClicked()	40
9.4.3.14 load()	40
9.4.3.15 loadError	41
9.4.3.16 modified	41
9.4.3.17 mouseMoveEvent()	41
9.4.3.18 mousePressEvent()	42
9.4.3.19 paintEvent()	42
9.4.3.20 query	42
9.4.3.21 queryMode	43
9.4.3.22 remove	43
9.4.3.23 rightButtonClicked()	43
9.4.3.24 save()	44
9.4.3.25 setBGColor()	44
9.4.3.26 setComponentColor()	44
9.4.3.27 setFileName()	45
9.4.3.28 setGridColor()	45
9.4.3.29 setSelectedButton()	46

9.4.3.30 setSelectedColor()	46
9.4.3.31 setStatus()	46
9.4.3.32 showEditDialog	47
9.4.3.33 statusBarText	47
9.4.4 Atributos	47
9.4.4.1 backgroundColor	48
9.4.4.2 circuit	48
9.4.4.3 clickedStack	48
9.4.4.4 componentColor	48
9.4.4.5 connections	48
9.4.4.6 cursorLocation	48
9.4.4.7 drawList	49
9.4.4.8 editButton	49
9.4.4.9 editMenu	49
9.4.4.10 fileName	49
9.4.4.11 gridColor	49
9.4.4.12 mode	49
9.4.4.13 playButton	50
9.4.4.14 queryMenu	50
9.4.4.15 selectedButton	50
9.4.4.16 selectedColor	50
9.4.4.17 selectedComponent	50
9.4.4.18 selectedPrev	50
9.4.4.19 status	51
9.4.4.20 vtxCounter	51
9.4.4.21 wireCounter	51
9.5 Referência da Classe NM::EquationSystem	51
9.5.1 Descrição detalhada	52
9.5.2 Construtores e Destrutores	52
9.5.2.1 EquationSystem()	52
9.5.3 Funções membros	52
9.5.3.1 findPivot()	52
9.5.3.2 gaussJordan()	53
9.5.3.3 gaussSeidel()	53
9.5.3.4 getSolution()	54
9.5.3.5 sassenfeldCriteria()	54
9.5.4 Atributos	54
9.5.4.1 A	54
9.5.4.2 B	55
9.5.4.3 x	55
9.6 Referência da Classe GraphicComponent	55
9.6.1 Descrição detalhada	57

9.6.2 Construtores e Destrutores	57
9.6.2.1 GraphicComponent()	57
9.6.3 Funções membros	57
9.6.3.1 clickedArea()	58
9.6.3.2 draw()	58
9.6.3.3 getBottom()	58
9.6.3.4 getBoundRect()	59
9.6.3.5 getHeight()	59
9.6.3.6 getLabel()	59
9.6.3.7 getLeft()	60
9.6.3.8 getOrientation()	60
9.6.3.9 getRight()	60
9.6.3.10 getTop()	60
9.6.3.11 getType()	61
9.6.3.12 getValue()	61
9.6.3.13 getVertex1()	61
9.6.3.14 getVertex1Point()	62
9.6.3.15 getVertex2()	62
9.6.3.16 getVertex2Point()	62
9.6.3.17 getWidth()	62
9.6.3.18 setValue()	62
9.6.3.19 setVertex1()	63
9.6.3.20 setVertex2()	63
9.6.3.21 updateName()	64
9.6.4 Atributos	64
9.6.4.1 boundRect	64
9.6.4.2 componentType	64
9.6.4.3 label	64
9.6.4.4 map	64
9.6.4.5 orientation	65
9.6.4.6 value	65
9.6.4.7 vertex1	65
9.6.4.8 vertex2	65
9.6.4.9 vertexArea1	65
9.6.4.10 vertexArea2	65
9.6.4.11 x	66
9.6.4.12 y	66
9.7 Referência da Classe GRF::incidenceMatrix	66
9.7.1 Descrição detalhada	67
9.7.2 Construtores e Destrutores	67
9.7.2.1 incidenceMatrix() [1/2]	68
9.7.2.2 incidenceMatrix() [2/2]	68

9.7.3 Funções membros	68
9.7.3.1 addEdge()	68
9.7.3.2 getConNum()	69
9.7.3.3 getEdgeNumber()	69
9.7.3.4 getEdges() [1/2]	69
9.7.3.5 getEdges() [2/2]	70
9.7.3.6 getLoop()	70
9.7.3.7 getSpanningTree()	71
9.7.3.8 getVertex()	71
9.7.3.9 getVertexCon()	71
9.7.3.10 getVertexNumber()	72
9.7.3.11 makeCon()	72
9.7.3.12 removeEdge()	73
9.7.3.13 removeVertex()	73
9.7.4 Atributos	73
9.7.4.1 edgeNumber	73
9.7.4.2 inMatrix	74
9.7.4.3 vertexNumber	74
9.8 Referência da Classe MainWindow	74
9.8.1 Descrição detalhada	76
9.8.2 Construtores e Destrutores	77
9.8.2.1 MainWindow() [1/2]	77
9.8.2.2 MainWindow() [2/2]	77
9.8.3 Funções membros	77
9.8.3.1 closeFile	77
9.8.3.2 drawRes180	78
9.8.3.3 drawRes90	78
9.8.3.4 drawVcc180	78
9.8.3.5 drawVcc90	79
9.8.3.6 getMainWindow()	79
9.8.3.7 initializeMenu()	79
9.8.3.8 initializeStatusBar()	79
9.8.3.9 initializeTabs()	80
9.8.3.10 initializeToolbar()	80
9.8.3.11 loadConfig()	80
9.8.3.12 newFile	80
9.8.3.13 openFile	81
9.8.3.14 operator=()	81
9.8.3.15 resetConfig	81
9.8.3.16 saveConfig()	81
9.8.3.17 saveFile	82
9.8.3.18 saveFileAs	82

9.8.3.19 setBGColor	82
9.8.3.20 setComponentColor	83
9.8.3.21 setGridColor	83
9.8.3.22 setSelectedColor	83
9.8.3.23 setTabStatus	83
9.8.3.24 tutorial	84
9.8.4 Atributos	84
9.8.4.1 diagrams	84
9.8.4.2 fileMenu	84
9.8.4.3 helpMenu	84
9.8.4.4 instance	85
9.8.4.5 mainBar	85
9.8.4.6 newFileAct	85
9.8.4.7 openFileAct	85
9.8.4.8 prefMenu	85
9.8.4.9 resetConfigAct	85
9.8.4.10 saveFileAct	86
9.8.4.11 saveFileAsAct	86
9.8.4.12 setBGColorAct	86
9.8.4.13 setComponentColorAct	86
9.8.4.14 setGridColorAct	86
9.8.4.15 setSelectedColorAct	86
9.8.4.16 statusBar	87
9.8.4.17 tabs	87
9.8.4.18 toolbar	87
9.8.4.19 tutorialAct	87
9.9 Referência da Classe NM::Matrix	87
9.9.1 Descrição detalhada	88
9.9.2 Construtores e Destrutores	89
9.9.2.1 Matrix() [1/2]	89
9.9.2.2 Matrix() [2/2]	89
9.9.3 Funções membros	89
9.9.3.1 Abs()	89
9.9.3.2 getCol()	90
9.9.3.3 getColNumber()	90
9.9.3.4 getRowNumber()	90
9.9.3.5 operator*()	91
9.9.3.6 operator*=()	91
9.9.3.7 operator+()	91
9.9.3.8 operator+=()	92
9.9.3.9 operator-() [1/2]	92
9.9.3.10 operator-() [2/2]	92

9.9.3.11 operator-=()	93
9.9.3.12 operator=()	93
9.9.3.13 operator[] ()	93
9.9.3.14 swapLines ()	94
9.9.3.15 transpose ()	94
9.9.4 Atributos	94
9.9.4.1 colNumber	95
9.9.4.2 realMatrix	95
9.9.4.3 rowNumber	95
9.10 Referência da Classe CMP::Resistor	95
9.10.1 Descrição detalhada	96
9.10.2 Construtores e Destrutores	96
9.10.2.1 Resistor ()	96
9.10.2.2 ~Resistor ()	97
9.10.3 Funções membros	97
9.10.3.1 getResistance ()	97
9.10.3.2 getType ()	97
9.10.3.3 setCurrent ()	97
9.10.3.4 setResistance ()	98
9.10.3.5 setVoltage ()	98
9.10.4 Atributos	98
9.10.4.1 resistance	99
9.11 Referência da Classe Resistor	99
9.11.1 Descrição detalhada	99
9.11.2 Construtores e Destrutores	99
9.11.2.1 Resistor ()	100
9.11.3 Funções membros	100
9.11.3.1 getType ()	100
9.12 Referência da Classe CMP::Vcc	101
9.12.1 Descrição detalhada	101
9.12.2 Construtores e Destrutores	101
9.12.2.1 Vcc ()	101
9.12.2.2 ~Vcc ()	102
9.12.3 Funções membros	102
9.12.3.1 getType ()	102
9.12.3.2 setCurrent ()	102
9.12.3.3 setVoltage ()	103
9.13 Referência da Classe Vcc	103
9.13.1 Descrição detalhada	104
9.13.2 Construtores e Destrutores	104
9.13.2.1 Vcc ()	104
9.13.3 Funções membros	105

9.13.3.1 <code>getType()</code>	105
9.13.4 Atributos	105
9.13.4.1 <code>vccCounter</code>	105
10 Arquivos	107
10.1 Referência do Arquivo <code>CircuitSim/Circuit.cpp</code>	107
10.1.1 Descrição detalhada	107
10.2 Referência do Arquivo <code>CircuitSim/Circuit.h</code>	108
10.2.1 Descrição detalhada	108
10.3 Referência do Arquivo <code>CircuitSim/Component.cpp</code>	108
10.3.1 Descrição detalhada	109
10.4 Referência do Arquivo <code>CircuitSim/Component.h</code>	109
10.4.1 Descrição detalhada	110
10.5 Referência do Arquivo <code>CircuitSim/Diagram.cpp</code>	110
10.5.1 Descrição detalhada	110
10.6 Referência do Arquivo <code>CircuitSim/Diagram.h</code>	111
10.6.1 Descrição detalhada	112
10.6.2 Enumerações	112
10.6.2.1 <code>cmpStyle</code>	112
10.6.2.2 <code>mode</code>	112
10.6.2.3 <code>stats</code>	113
10.7 Referência do Arquivo <code>CircuitSim/Graph.cpp</code>	113
10.7.1 Descrição detalhada	113
10.8 Referência do Arquivo <code>CircuitSim/Graph.h</code>	114
10.8.1 Descrição detalhada	114
10.9 Referência do Arquivo <code>CircuitSim/GraphicComponent.cpp</code>	114
10.9.1 Descrição detalhada	115
10.10 Referência do Arquivo <code>CircuitSim/GraphicComponent.h</code>	115
10.10.1 Descrição detalhada	116
10.10.2 Enumerações	116
10.10.2.1 <code>orien</code>	116
10.11 Referência do Arquivo <code>CircuitSim/main.cpp</code>	116
10.11.1 Funções	117
10.11.1.1 <code>main()</code>	117
10.12 Referência do Arquivo <code>CircuitSim/MainWindow.cpp</code>	117
10.12.1 Descrição detalhada	117
10.13 Referência do Arquivo <code>CircuitSim/MainWindow.h</code>	118
10.13.1 Descrição detalhada	118
10.14 Referência do Arquivo <code>CircuitSim/Numeric.cpp</code>	118
10.14.1 Descrição detalhada	119
10.15 Referência do Arquivo <code>CircuitSim/Numeric.h</code>	119
10.15.1 Descrição detalhada	119

Capítulo 1

Descrição

Este projeto consiste em um simulador de circuitos compostos por resistores e fontes de tensão contínua com interface gráfica.

Capítulo 2

Índice dos Módulos

2.1 Módulos

Esta é a lista de todos os módulos:

Cores padrão	13
Tamanho dos Componentes	14

Capítulo 3

Namespaces

3.1 Lista de Namespaces

Esta é a lista de todos os Namespaces com suas respectivas descrições:

CCT	15
CMP	15
GRF	16
NM	16

Capítulo 4

Índice Hierárquico

4.1 Hierarquia de Classes

Esta lista de hierarquias está parcialmente ordenada (ordem alfabética):

GRF::adjacencyMatrix	17
CMP::Component	27
CMP::Resistor	95
CMP::Vcc	101
NM::EquationSystem	51
GRF::incidenceMatrix	66
CCT::Circuit	21
NM::Matrix	87
QMainWindow	
MainWindow	74
QObject	
GraphicComponent	55
Resistor	99
Vcc	103
QWidget	
Diagram	33

Capítulo 5

Índice dos Componentes

5.1 Lista de Classes

Aqui estão as classes, estruturas, uniões e interfaces e suas respectivas descrições:

GRF::adjacencyMatrix	
Declaração da classe adjacencyMatrix	17
CCT::Circuit	
Declaração da classe Circuit	21
CMP::Component	
Declaração da classe abstrata Component	27
Diagram	
Declaração da classe Diagram	33
NM::EquationSystem	
Declaração da classe EquationSystem	51
GraphicComponent	
Declaração da classe GraphicComponent	55
GRF::incidenceMatrix	
Declaração da classe incidenceMatrix	66
MainWindow	
Declaração da classe MainWindow	74
NM::Matrix	
Declaração da classe Matrix	87
CMP::Resistor	
Declaração da classe Resistor	95
Resistor	
Declaração da classe Resistor	99
CMP::Vcc	
Declaração da classe Vcc	101
Vcc	
Declaração da classe Vcc	103

Capítulo 6

Índice dos Arquivos

6.1 Lista de Arquivos

Esta é a lista de todos os arquivos e suas respectivas descrições:

CircuitSim/ Circuit.cpp	
Implementação da classe Circuit	107
CircuitSim/ Circuit.h	
Implementação da classe Circuit	108
CircuitSim/ Component.cpp	
Implementação da classe Component	108
CircuitSim/ Component.h	
Declaração da classe Component	109
CircuitSim/ Diagram.cpp	
Implementação da classe Diagram	110
CircuitSim/ Diagram.h	
Declaração da classe Numeric	111
CircuitSim/ Graph.cpp	
Implementação das classes de grafo utilizadas no projeto	113
CircuitSim/ Graph.h	
Declaração das classes de grafo utilizadas no projeto	114
CircuitSim/ GraphicComponent.cpp	
Implementação da classe GraphicComponent para a inserção gráfica de componentes	114
CircuitSim/ GraphicComponent.h	
Declaração da classe GraphicComponent para a inserção gráfica de componentes	115
CircuitSim/ main.cpp	116
CircuitSim/ MainWindow.cpp	
Implementação da classe MainWindow	117
CircuitSim/ MainWindow.h	
Declaração da classe MainWindow	118
CircuitSim/ Numeric.cpp	
Implementação das classe Matrix e EquationSystem	118
CircuitSim/ Numeric.h	
Declaração das classe Matrix e EquationSystem	119

Capítulo 7

Módulos

7.1 Cores padrão

Definição as macros para os códigos hexadecimais de cores padrão do tema do programa.

Definições e Macros

- `#define DEFAULT_BGC "#272947"`
Código hexadecimal para a cor padrão do plano de fundo.
- `#define DEFAULT_LC "#141516"`
Código hexadecimal para a cor padrão da grade do plano de fundo.
- `#define DEFAULT_CC "#FFFFFF"`
Código hexadecimal para a cor padrão dos componentes.
- `#define DEFAULT_SC "#0AA206"`
Código hexadecimal para a cor padrão de seleção de componentes.

7.1.1 Descrição detalhada

Definição as macros para os códigos hexadecimais de cores padrão do tema do programa.

7.1.2 Definições e macros

7.1.2.1 DEFAULT_BGC

```
#define DEFAULT_BGC "#272947"
```

Código hexadecimal para a cor padrão do plano de fundo.

7.1.2.2 DEFAULT_CC

```
#define DEFAULT_CC "#FFFFFF"
```

Código hexadecimal para a cor padrão dos componentes.

7.1.2.3 DEFAULT_LC

```
#define DEFAULT_LC "#141516"
```

Código hexadecimal para a cor padrão da grade do plano de fundo.

7.1.2.4 DEFAULT_SC

```
#define DEFAULT_SC "#0AA206"
```

Código hexadecimal para a cor padrão de seleção de componentes.

7.2 Tamanho dos Componentes

Definição as macros para os tamanhos dos desenhos dos componentes na tela.

Definições e Macros

- #define HEIGHT 115
- #define WIDTH 50

7.2.1 Descrição detalhada

Definição as macros para os tamanhos dos desenhos dos componentes na tela.

7.2.2 Definições e macros

7.2.2.1 HEIGHT

```
#define HEIGHT 115
```

7.2.2.2 WIDTH

```
#define WIDTH 50
```

Capítulo 8

Namespace

8.1 Refência do Namespace CCT

Componentes

- class `Circuit`
Declaração da classe `Circuit`.

8.2 Refência do Namespace CMP

Componentes

- class `Component`
Declaração da classe abstrata `Component`.
- class `Resistor`
Declaração da classe `Resistor`.
- class `Vcc`
Declaração da classe `Vcc`.

Enumerações

- enum `type` { `RESISTOR` , `VCC` }
Fornece uma identificação para os tipos de componentes disponíveis no programa.

8.2.1 Enumerações

8.2.1.1 `type`

```
enum CMP::type
```

Fornece uma identificação para os tipos de componentes disponíveis no programa.

Estas identificações são utilizadas nos métodos da classe para se tomar ações com base no tipo especificado.

Enumeradores

RESISTOR	Componente do tipo Resistor .
VCC	Fonte de Tensão de corrente contínua.

8.3 Refência do Namespace GRF

Componentes

- class [incidenceMatrix](#)
Declaração da classe [incidenceMatrix](#).
- class [adjacencyMatrix](#)
Declaração da classe [adjacencyMatrix](#).

8.4 Refência do Namespace NM

Componentes

- class [Matrix](#)
Declaração da classe [Matrix](#).
- class [EquationSystem](#)
Declaração da classe [EquationSystem](#).

Capítulo 9

Classes

9.1 Referência da Classe GRF::adjacencyMatrix

Declaração da classe [adjacencyMatrix](#).

```
#include <Graph.h>
```

Membros Públicos

- [adjacencyMatrix](#) ()
Construtor para a classe [adjacencyMatrix](#).
- [adjacencyMatrix](#) (unsigned int vertexNum)
Construtor para a classe [adjacencyMatrix](#).
- void [insertEdge](#) (unsigned int vtx1, unsigned int vtx2)
Insera uma aresta entre dois vértices de um objeto da classe [adjacencyMatrix](#).
- void [insertVertex](#) (unsigned int vtx)
Insera um vértice em um objeto da classe [adjacencyMatrix](#).
- void [removeVertex](#) (unsigned int vtx)
Remove um vértice de um objeto da classe [adjacencyMatrix](#).
- int [query](#) (unsigned int vtx1, unsigned int vtx2)
Verifica se há conexão entre 2 vértices de um objeto da classe [adjacencyMatrix](#).
- unsigned int [getVertexNumber](#) ()
Getter para o número de vértices de um objeto da classe [adjacencyMatrix](#).

Atributos Privados

- unsigned int [vertexNumber](#)
Armazena o número de vértices do grafo.
- std::vector< std::vector< int > > [adjMatrix](#)
Vector bidimensional que armazena as conexões entre os vértices.

9.1.1 Descrição detalhada

Declaração da classe [adjacencyMatrix](#).

Representa um grafo direcionado através de uma matriz de adjacências.

9.1.2 Construtores e Destrutores

9.1.2.1 adjacencyMatrix() [1/2]

```
GRF::adjacencyMatrix::adjacencyMatrix ( )
```

Construtor para a classe [adjacencyMatrix](#).

Constroi um objeto da classe [adjacencyMatrix](#) cujo vetor de conexões está inicialmente vazio.

9.1.2.2 adjacencyMatrix() [2/2]

```
GRF::adjacencyMatrix::adjacencyMatrix (
    unsigned int vertexNum )
```

Construtor para a classe [adjacencyMatrix](#).

Constroi um objeto da classe [adjacencyMatrix](#) com número de vértices predefinido.

Parâmetros

<i>vertexNum</i>	Número de vértices do grafo.
------------------	------------------------------

Esse é um método provido por conveniência. Ele difere do método acima apenas na lista de argumentos que devem ser utilizados.

9.1.3 Funções membros

9.1.3.1 getVertexNumber()

```
unsigned int GRF::adjacencyMatrix::getVertexNumber ( )
```

Getter para o número de vértices de um objeto da classe [adjacencyMatrix](#).

Retorna o número de vértices presentes atualmente no grafo.

Retorna

Número de vértices do grafo.

9.1.3.2 insertEdge()

```
void GRF::adjacencyMatrix::insertEdge (
    unsigned int vtx1,
    unsigned int vtx2 )
```

Insere uma aresta entre dois vértices de um objeto da classe [adjacencyMatrix](#).

Insere dinamicamente uma aresta entre dois vértices. Se ao menos um dos vértices não existe, são criados novos vértices até que o grafo tenha o tamanho necessário para que se possa utilizar estes vértices.

Parâmetros

<i>vtx1</i>	Vértice de saída.
<i>vtx2</i>	Vértice de entrada.

Retorna

Void.

9.1.3.3 insertVertex()

```
void GRF::adjacencyMatrix::insertVertex (
    unsigned int vtx )
```

Insere um vértice em um objeto da classe [adjacencyMatrix](#).

Insere dinamicamente um vértice no grafo. São criados novos vértices até que o grafo tenha o tamanho necessário para que se possa utilizar o novo vértice. Caso o vértice já esteja no grafo, não executa nada.

Parâmetros

<i>vtx</i>	Vértice a ser inserido.
------------	-------------------------

Retorna

Void.

9.1.3.4 query()

```
int GRF::adjacencyMatrix::query (
    unsigned int vtx1,
    unsigned int vtx2 )
```

Verifica se há conexão entre 2 vértices de um objeto da classe [adjacencyMatrix](#).

Verifica se há uma conexão direcionada de *vtx1* para *vtx2* no grafo.

Parâmetros

<i>vtx1</i>	Vértice de saída.
<i>vtx2</i>	Vértice de entrada

Retorna

1 se há conexão e 0 se não há.

9.1.3.5 removeVertex()

```
void GRF::adjacencyMatrix::removeVertex (
    unsigned int vtx )
```

Remove um vértice de um objeto da classe [adjacencyMatrix](#).

Remove um vértice do grafo. A linha e a coluna correspondentes ao vértice são removidas e o tamanho do grafo é reduzido. Todos os vértices com número maior são deslocados e têm seu número reduzido.

Parâmetros

<i>vtx</i>	Vértice que se deseja remover.
------------	--------------------------------

Retorna

Void.

9.1.4 Atributos**9.1.4.1 adjMatrix**

```
std::vector<std::vector<int> > GRF::adjacencyMatrix::adjMatrix [private]
```

Vector bidimensional que armazena as conexões entre os vértices.

9.1.4.2 vertexNumber

```
unsigned int GRF::adjacencyMatrix::vertexNumber [private]
```

Armazena o número de vértices do grafo.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

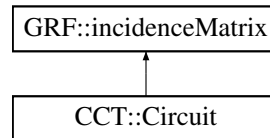
- [CircuitSim/Graph.h](#)
- [CircuitSim/Graph.cpp](#)

9.2 Referência da Classe CCT::Circuit

Declaração da classe `Circuit`.

```
#include <Circuit.h>
```

Diagrama de hierarquia para CCT::Circuit:



Membros Públicos

- `Circuit ()`
Construtor para a classe `Circuit`.
- `~Circuit ()`
Desconstrutor para a classe `Circuit`.
- `void initialize ()`
Inicializa um objeto da classe `Circuit`.
- `void reset ()`
Reseta a matriz de circuitos fundamentais.
- `void addComponent (CMP::type t, std::string l, double value, unsigned int vtx1, unsigned int vtx2)`
Adiciona um objeto da classe `Component` no circuito.
- `void editComponent (std::string label, double value)`
Edita o valor de um objeto da classe `Component` pertencente ao circuito.
- `void editComponent (std::string label, std::string newLabel)`
Edita a nome de identificação de um objeto da classe `Component` pertencente ao circuito.
- `void removeComponent (std::string l)`
Remove um objeto da classe `Component` pertencente ao circuito.
- `void removeComponent (unsigned int edge)`
Remove um objeto da classe `Component` pertencente ao circuito.
- `std::string getComponentLabel (unsigned int edge)`
Getter para o nome de identificação de um componente pertencente ao circuito.
- `double getVoltage (std::string l)`
Getter para a tensão através de um componente.
- `double getCurrent (std::string l)`
Getter para a corrente através de um componente.

Membros Privados

- `void updateComponents (std::vector< double > currents)`
Atualiza os valores dos componentes contidos em um objeto da classe `Circuit`.
- `void Solve ()`
Resolve o circuito.

Atributos Privados

- `std::vector< CMP::Component * > components`
Armazena os componentes inseridos.
- `std::vector< std::vector< int > > circuitMatrix`
Armazena os circuitos fundamentais do grafo.
- `std::vector< unsigned int > chords`
Armazena as arestas que não estão na árvore geradora do grafo.

Outros membros herdados

9.2.1 Descrição detalhada

Declaração da classe `Circuit`.

Representa um circuito de componentes eletrônicos através de uma matriz de incidência. Esta classe herda da classe `incidenceMatrix`.

9.2.2 Construtores e Destrutores

9.2.2.1 `Circuit()`

```
CCT::Circuit::Circuit ( )
```

Construtor para a classe `Circuit`.

Constroi um objeto da Classe `Circuit`.

9.2.2.2 `~Circuit()`

```
CCT::Circuit::~~Circuit ( )
```

Desconstrutor para a classe `Circuit`.

Destrói um objeto da Classe `Circuit`.

9.2.3 Funções membros

9.2.3.1 `addComponent()`

```
void CCT::Circuit::addComponent (
    CMP::type t,
    std::string l,
    double value,
    unsigned int vtx1,
    unsigned int vtx2 )
```

Adiciona um objeto da classe `Component` no circuito.

Os vértices do componente representam nós com que outros componetes podem ser aadicionados, isto é, se dois componentes estão conectados, então, estes componentes possuem um vértice em comum.

Parâmetros

<i>t</i>	Tipo do componente a ser adicionado no circuito.
<i>l</i>	Nome de identificação do componente a ser adicionado.
<i>value</i>	Valor associado ao componente a ser adicionado (resistência/tensão).
<i>vtx1</i>	Número do primeiro vértice do componente.
<i>vtx2</i>	Número do segundo vértice do componente.

Retorna

Void.

9.2.3.2 editComponent() [1/2]

```
void CCT::Circuit::editComponent (
    std::string label,
    double value )
```

Edita o valor de um objeto da classe Component pertencente ao circuito.

Caso o componente esteja no circuito, altera o valor relacionado ao componente (resistência/tensão) cujo nome de identificação é *l*.

Parâmetros

<i>value</i>	Novo valor do componente a ser editado.
<i>label</i>	Nome de identificação do componente a ser editado.

Retorna

Void.

9.2.3.3 editComponent() [2/2]

```
void CCT::Circuit::editComponent (
    std::string label,
    std::string newLabel )
```

Edita a nome de identificação de um objeto da classe Component pertencente ao circuito.

Caso o componente esteja no circuito, altera o nome de identificação do componente a ser editado.

Parâmetros

<i>label</i>	Nome de identificação do componente a ser editado.
<i>newLabel</i>	Novo nome de identificação do componente a ser editado.

Retorna

Void.

Esse é um método provido por conveniência. Ele difere do método acima apenas na lista de argumentos que devem ser utilizados.

9.2.3.4 GetComponentLabel()

```
std::string CCT::Circuit::GetComponentLabel (
    unsigned int edge )
```

Getter para o nome de identificação de um componente pertencente ao circuito.

Caso o componente exista no circuito, retorna o nome de identificação do componente identificado pela aresta edge.

Parâmetros

<i>edge</i>	Aresta do componente consultado.
-------------	----------------------------------

Retorna

Nome de identificação do componente consultado.

9.2.3.5 getCurrent()

```
double CCT::Circuit::getCurrent (
    std::string l )
```

Getter para a corrente através de um componente.

Caso o componente exista no circuito, retorna o valor da corrente através do componente com a Nome de identificação fornecida.

Parâmetros

<i>l</i>	Nome de identificação do componente consultado.
----------	---

Retorna

Valor da corrente através do componente consultado.

9.2.3.6 getVoltage()

```
double CCT::Circuit::getVoltage (
    std::string l )
```

Getter para a tensão através de um componente.

Caso o componente exista no circuito, retorna o valor da tensão do componente com a Nome de identificação fornecida.

Parâmetros

/	Nome de identificação do componente consultado.
---	---

Retorna

Valor da tensão através do componente consultado.

9.2.3.7 initialize()

```
void CCT::Circuit::initialize ( )
```

Inicializa um objeto da classe [Circuit](#).

Obtém os circuitos fundamentais do grafo de circuito e armazena em circuitMatrix, bem como, atualiza o vector chords contendo as arestas que não estão contidas na árvore geradora do grafo e após isso, resolve o circuito e atualiza os valores de cada componente.

Retorna

void

9.2.3.8 removeComponent() [1/2]

```
void CCT::Circuit::removeComponent (
    std::string l )
```

Remove um objeto da classe Component pertencente ao circuito.

Caso o componente com o nome de identificação fornecido exista, remove este do circuito.

Parâmetros

/	Nome de identificação do componente a ser removido.
---	---

Retorna

Void.

9.2.3.9 removeComponent() [2/2]

```
void CCT::Circuit::removeComponent (
    unsigned int edge )
```

Remove um objeto da classe Component pertencente ao circuito.

Caso o componente exista no circuito, remove este remove.

Parâmetros

<i>edge</i>	Aresta do componente a ser removido.
-------------	--------------------------------------

Retorna

Void.

9.2.3.10 reset()

```
void CCT::Circuit::reset ( )
```

Reseta a matriz de circuitos fundamentais.

Remove todos os elementos contidos do vector circuitMatrix e do vector chords.

Retorna

Void.

9.2.3.11 Solve()

```
void CCT::Circuit::Solve ( ) [private]
```

Resolve o circuito.

Obtém os valores das correntes em cada componente e atualiza cada componente.

Retorna

void.

9.2.3.12 updateComponents()

```
void CCT::Circuit::updateComponents (
    std::vector< double > currents ) [private]
```

Atualiza os valores dos componentes contidos em um objeto da classe [Circuit](#).

Atualiza as correntes e tensões dos componentes contidos no circuito com base nos dados passados por parâmetro.

Parâmetros

<i>currents</i>	um vector contendo os novos valores para as correntes dos componentes.
-----------------	--

Retorna

void.

9.2.4 Atributos

9.2.4.1 chords

```
std::vector<unsigned int> CCT::Circuit::chords [private]
```

Armazena as arestas que não estão na árvore geradora do grafo.

9.2.4.2 circuitMatrix

```
std::vector<std::vector<int> > CCT::Circuit::circuitMatrix [private]
```

Armazena os circuitos fundamentais do grafo.

9.2.4.3 components

```
std::vector<CMP::Component*> CCT::Circuit::components [private]
```

Armazena os componentes inseridos.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

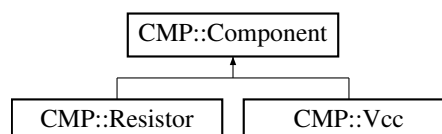
- [CircuitSim/Circuit.h](#)
- [CircuitSim/Circuit.cpp](#)

9.3 Referência da Classe CMP::Component

Declaração da classe abstrata [Component](#).

```
#include <Component.h>
```

Diagrama de hierarquia para CMP::Component:



Membros Públicos

- [Component](#) (std::string l, unsigned int vtx1, unsigned int vtx2)
Construtor para a classe [Component](#).
- virtual [~Component](#) ()
Desconstrutor virtual da classe [Component](#).
- void [setLabel](#) (std::string newLabel)
Setter para o nome de identificação de um objeto da classe [Component](#).
- std::string [getLabel](#) ()
Getter para o nome de identificação de um objeto da classe [Component](#).
- double [getVoltage](#) ()
Getter para a tensão de um objeto da classe [Component](#).
- double [getCurrent](#) ()
Getter para a corrente de um objeto da classe [Component](#).
- std::pair< unsigned int, unsigned int > [getNodes](#) ()
Getter para os vértices (terminais) de um objeto da classe [Component](#).
- virtual enum [type](#) [getType](#) ()=0
Getter para o tipo de um objeto da classe [Component](#).
- virtual void [setVoltage](#) (double value)=0
Setter para a tensão de um objeto da classe [Component](#).
- virtual void [setCurrent](#) (double value)=0
Setter para a corrente de um objeto da classe [Component](#).

Atributos Protegidos

- std::string [label](#)
Armazena o nome de identificação do componente.
- std::pair< unsigned int, unsigned int > [vtxs](#)
Armazena um par de inteiros utilizados para identificar os vértices do componente.
- double [voltage](#)
Armazena a tensão através dos terminais do componente.
- double [current](#)
Armazena a corrente através dos terminais do componente.

9.3.1 Descrição detalhada

Declaração da classe abstrata [Component](#).

Representa um componente genérico, especificando os atributos em comum de qualquer componente presente no circuito. Esta classe é desenvolvida para a utilização conjunta com a classe [Circuit](#).

9.3.2 Construtores e Destrutores

9.3.2.1 Component()

```
CMP::Component::Component (
    std::string l,
    unsigned int vtx1,
    unsigned int vtx2 )
```

Construtor para a classe [Component](#).

Constroi um objeto da Classe [Component](#) com os parâmetros especificados.

Parâmetros

<i>l</i>	Nome para a identificação do componente.
<i>vtx1</i>	Identificação do terminal 1 do componente.
<i>vtx2</i>	Identificação do terminal 2 do componente.

9.3.2.2 ~Component()

```
CMP::Component::~~Component ( ) [virtual]
```

Desconstrutor virtual da classe [Component](#).

Destrói um objeto da classe [Component](#).

9.3.3 Funções membros**9.3.3.1 getCurrent()**

```
double CMP::Component::getCurrent ( )
```

Getter para a corrente de um objeto da classe [Component](#).

Retorna a corrente através dos terminais do componente representado pelo objeto.

Retorna

Corrente do componente.

9.3.3.2 getLabel()

```
std::string CMP::Component::getLabel ( )
```

Getter para o nome de identificação de um objeto da classe [Component](#).

Retorna o nome de identificação do objeto em formato no formato std::string.

Retorna

Nome de identificação do componente.

9.3.3.3 `getNodes()`

```
std::pair< unsigned int, unsigned int > CMP::Component::getNodes ( )
```

Getter para os vértices (terminais) de um objeto da classe [Component](#).

Retorna um `std::pair` constendo os vértices que representam os terminais do componente. O primeiro elemento do `pair` representa o vértice 1 e o segundo elemento representa o vértice 2.

Retorna

Vértices do componente.

9.3.3.4 `getType()`

```
virtual enum type CMP::Component::getType ( ) [pure virtual]
```

Getter para o tipo de um objeto da classe [Component](#).

Método virtual puro que retorna o tipo do objeto. Este método deve ser sobrescrito pelas classes derivadas, pois cada componente tem seu tipo.

Retorna

Tipo do objeto.

Implementado por [CMP::Vcc](#) e [CMP::Resistor](#).

9.3.3.5 `getVoltage()`

```
double CMP::Component::getVoltage ( )
```

Getter para a tensão de um objeto da classe [Component](#).

Retorna a tensão através dos terminais do componente representado pelo objeto.

Retorna

Tensão do componente.

9.3.3.6 `setCurrent()`

```
virtual void CMP::Component::setCurrent (
    double value ) [pure virtual]
```

Setter para a corrente de um objeto da classe [Component](#).

Método virtual puro que altera a corrente do componente.

Parâmetros

<i>value</i>	Valor para o parâmetro utilizado para alterar a corrente de um objeto da classe Component . Este método deve ser sobrescrito pelas classes derivadas, uma vez que, cada componente possui uma forma específica de se atribuir a corrente. Apesar de neste projeto ambos os componentes possuírem a mesma forma de se atribuir a corrente, o método é virtual puro pois, em futuras atualizações, outros tipos de componentes podem ser adicionados.
--------------	---

Retorna

Void.

Implementado por [CMP::Resistor](#) e [CMP::Vcc](#).

9.3.3.7 setLabel()

```
void CMP::Component::setLabel (
    std::string newLabel )
```

Setter para o nome de identificação de um objeto da classe [Component](#).

Aletera o nome do objeto para newLabel.

Parâmetros

<i>newLabel</i>	Novo nome de identificação do componente.
-----------------	---

return Void.

9.3.3.8 setVoltage()

```
virtual void CMP::Component::setVoltage (
    double value ) [pure virtual]
```

Setter para a tensão de um objeto da classe [Component](#).

Método virtual puro que altera a tensão do componente.

Parâmetros

<i>value</i>	Valor para o parâmetro utilizado para alterar a corrente de um objeto da classe Component . Este método deve ser sobrescrito pelas classes derivadas, uma vez que, cada componente possui uma forma específica de se atribuir a tensão.
--------------	---

Retorna

Void.

Implementado por [CMP::Vcc](#) e [CMP::Resistor](#).

9.3.4 Atributos

9.3.4.1 current

```
double CMP::Component::current [protected]
```

Armazena a corrente através dos terminais do componente.

9.3.4.2 label

```
std::string CMP::Component::label [protected]
```

Armazena o nome de identificação do componente.

9.3.4.3 voltage

```
double CMP::Component::voltage [protected]
```

Armazena a tensão através dos terminais do componente.

9.3.4.4 vtxs

```
std::pair<unsigned int, unsigned int> CMP::Component::vtxs [protected]
```

Armazena um par de inteiros utilizados para identificar os vértices do componente.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

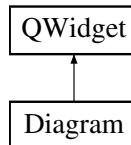
- [CircuitSim/Component.h](#)
- [CircuitSim/Component.cpp](#)

9.4 Referência da Classe Diagram

Declaração da classe [Diagram](#).

```
#include <Diagram.h>
```

Diagrama de hierarquia para Diagram:



Slots Públicos

- void [queryMode](#) ()
Ativa o modo de Consulta para o circuito representado no diagrama.
- void [editMode](#) ()
Ativa o modo de Edição para o circuito representado no diagrama.
- void [showEditDialog](#) ()
Ativa a dialog de edição.
- void [edit](#) (double newValue)
Altera o valor do componente selecionado.
- void [remove](#) ()
Remove o componente selecionado do circuito.
- void [query](#) ()
Consulta os dados obtidos após a solução do circuito.

Sinais

- void [modified](#) (bool checked=false)
Sinal emitido quando o diagrama é modificado.
- void [loadError](#) (bool checked=false)
Sinal emitido quando um erro de abertura de arquivo ocorre.
- void [statusBarText](#) (QString str)
Sinal emitido quando a classe [Diagram](#) necessita exibir uma mensagem na barra de status da janela principal.

Membros Públicos

- [Diagram](#) (QWidget *parent=nullptr)
Construtor para a classe [Diagram](#).
- QString [getFileName](#) ()
Getter para o nome do arquivo que contém o objeto.
- void [setFileName](#) (QString file)
Setter para o nome do arquivo que contém o objeto.
- void [save](#) ()
Salva os dados cruciais do objeto da classe [Diagram](#).
- void [load](#) ()

- enum `stats getStatus` ()
Carrega os dados previamente salvos para um objeto da classe `Diagram`.
- void `setSelectedButton` (enum `cmpStyle` button)
Getter para o status do arquivo de um objeto da classe `Diagram`.
- void `insert` (int x, int y)
Setter para o botão selecionado na barra de ferramentas.
- void `insert` (int x, int y)
Insere um componente em um objeto da classe `Diagram`.

Membros Públicos Estáticos

- static void `setBGColor` (QColor color)
Setter da cor do plano de fundo dos objetos da classe `Diagram`.
- static void `setGridColor` (QColor color)
Setter da cor da grade do plano de fundo dos objetos da classe `Diagram`.
- static void `setComponentColor` (QColor color)
Setter da cor dos componentes dos objetos da classe `Diagram`.
- static void `setSelectedColor` (QColor color)
Setter da cor de seleção dos components dos objetos da classe `Diagram`.
- static QColor `getBGColor` ()
Getter para a cor do plano de fundo dos objetos da classe `Diagram`.
- static QColor `getGridColor` ()
getter da cor da grade do plano de fundo dos objetos da classe `Diagram`.
- static QColor `getComponentColor` ()
Getter da cor dos componentes dos objetos da classe `Diagram`.
- static QColor `getSelectedColor` (void)
Setter da cor de seleção dos components dos objetos da classe `Diagram`.

Membros Protegidos

- void `paintEvent` (QPaintEvent *event) override
Sobrescrita do método `paintEvent`.
- void `mousePressEvent` (QMouseEvent *event) override
Sobrescrita do método `mousePressEvent`.
- void `mouseMoveEvent` (QMouseEvent *event) override
Sobrescrita do método `mouseMovePressEvent`.

Membros Privados

- void `initializeDiagram` ()
Método que inicializa os elementos gráficos da classe `Diagram`.
- void `setStatus` (enum `stats` newStatus)
Setter para o status do arquivo do objeto da classe `Diagram`.
- void `clickedControl` (int x, int y, int cArea)
Controla as ações referentes ao clique sobre componentes.
- std::pair< QRect, QPixmap > `getPixMap` (enum `cmpStyle` type)
Getter para o `pixmap` de um dos tipos de representações gráficas dos componentes.
- void `rightButtonClicked` (int x, int y, int cArea)
Método executado quando o o botão direito do mouse é pressionado.
- void `leftButtonClicked` (int x, int y, int cArea)
Método executado quando o o botão esquerdo do mouse é pressionado.

Atributos Privados

- [CCT::Circuit circuit](#)
Representa o circuito do diagrama.
- `std::string` [fileName](#)
Representa o nome do arquivo do diagrama.
- `enum` [stats status](#)
Representa o estado atual do arquivo do diagrama.
- `enum` [mode mode](#)
Representa o modo atual do diagrama (QUERY/EDIT).
- `enum` [cmpStyle selectedButton](#)
Representa o botão selecionado.
- `QPushButton *` [playButton](#)
Representa o botão play (entra em modo de consulta).
- `QPushButton *` [editButton](#)
Representa o botão edit (entra em modo de edição).
- `std::vector<` [GraphicComponent *](#) `>` [drawList](#)
Lista de componentes que devem ser desenhados na tela.
- [GraphicComponent *](#) [selectedComponent](#)
Representa o componente atualmente selecionado.
- [GRF::adjacencyMatrix connections](#)
Grafo que armazena as conexões entre os vértices dos componentes.
- `unsigned int` [wireCounter](#)
Contador para a quantidade de cabos inseridos.
- `unsigned int` [vtxCounter](#)
Contador para a quantidade de vértices inseridos.
- `QMenu *` [editMenu](#)
Menu de edição.
- `QMenu *` [queryMenu](#)
Menu de consulta.
- `QPoint` [cursorLocation](#)
Armazena a posição do cursor do mouse.
- `QPoint` [selectedPrev](#)
Armazena o último ponto selecionado.
- `std::stack<` `std::pair<` `int,` [GraphicComponent *](#) `>` `>` [clickedStack](#)
Pilha que armazena o último componente selecionado.

Atributos Privados Estáticos

- `static QColor` [backgroundColor](#) = `QColor(`[DEFAULT_BGC](#)`)`
Membro estático que armazena a cor do plano de fundo dos objetos da classe [Diagram](#).
- `static QColor` [gridColor](#) = `QColor(`[DEFAULT_LC](#)`)`
Membro estático que armazena a cor da grade do plano de fundo dos objetos da classe [Diagram](#).
- `static QColor` [componentColor](#) = `QColor(`[DEFAULT_CC](#)`)`
Membro estático que armazena a cor dos componentes dos objetos da classe [Diagram](#).
- `static QColor` [selectedColor](#) = `QColor(`[DEFAULT_SC](#)`)`
Membro estático que armazena a cor de seleção dos componentes dos objetos da classe [Diagram](#).

9.4.1 Descrição detalhada

Declaração da classe [Diagram](#).

[Diagram](#) herda da classe `QWidget`. Atua como uma interface gráfica para a utilização dos métodos da classe `Circuit`.

9.4.2 Construtores e Destrutores

9.4.2.1 `Diagram()`

```
Diagram::Diagram (
    QWidget * parent = nullptr ) [explicit]
```

Construtor para a classe [Diagram](#).

Parâmetros

<i>parent</i>	Pai do objeto.
---------------	----------------

Constrói um objeto da classe [Diagram](#).

9.4.3 Funções membros

9.4.3.1 `clickedControl()`

```
void Diagram::clickedControl (
    int x,
    int y,
    int cArea ) [private]
```

Controla as ações referentes ao clique sobre componentes.

Este método controla as ações ao se clicar em um componente com o botão esquerdo no modo de edição.

Parâmetros

<i>x</i>	Coordenada x do clique do mouse.
<i>y</i>	Coordenada y do clique do mouse.
<i>cArea</i>	Representa a área do componente que foi clicada (Área correspondente ao vértice 1 ou vértice 2).

Retorna

Void.

9.4.3.2 edit

```
void Diagram::edit (
    double newValue ) [slot]
```

Altera o valor do componente selecionado.

O SLOT é conectado com o sinal emitido pelo menu de edição. Quando o menu de edição é selecionado, uma janela para a edição do valor associado ao componente é exibida.

Parâmetros

<i>newValue</i>	Novo valor associado ao componente.
-----------------	-------------------------------------

Retorna

Void.

9.4.3.3 editMode

```
void Diagram::editMode ( ) [slot]
```

Ativa o modo de Edição para o circuito representado no diagrama.

O SLOT é conectado com o sinal emitido pelo botão edit. Assim que o botão é pressionado, o programa entra em modo de Edição.

Retorna

Void.

9.4.3.4 getBGColor()

```
QColor Diagram::getBGColor ( ) [static]
```

Getter para a cor do plano de fundo dos objetos da classe [Diagram](#).

Método estático que retorna a cor do plano de fundo de todos os objetos ativos e que serão criados da classe [Diagram](#).

Retorna

Cor do plano de fundo.

9.4.3.5 `getComponentColor()`

```
QColor Diagram::getComponentColor ( ) [static]
```

Getter da cor dos componentes dos objetos da classe [Diagram](#).

Método estático que retorna a cor dos componentes de todos os objetos ativos e que serão criados da classe [Diagram](#).

Retorna

Cor dos componentes.

9.4.3.6 `getFileName()`

```
QString Diagram::getFileName ( )
```

Getter para o nome do arquivo que contém o objeto.

Retorna o nome do arquivo em que o diagrama está salvo.

Retorna

Nome do arquivo que contém o diagrama.

9.4.3.7 `getGridColor()`

```
QColor Diagram::getGridColor ( ) [static]
```

getter da cor da grade do plano de fundo dos objetos da classe [Diagram](#).

Método estático que retorna a cor da grade do plano de fundo de todos os objetos ativos e que serão criados da classe [Diagram](#).

Retorna

Cor da grade do plano de fundo.

9.4.3.8 `getPixmap()`

```
std::pair< QRect, QPixmap > Diagram::getPixmap (
    enum cmpStyle type ) [private]
```

Getter para o pixmap de um dos tipos de representações gráficas dos componentes.

Este método retorna um `std::pair` contendo o retângulo que limita a área do componente e o pixmap de uma das representações gráficas dos componentes disponíveis.

Parâmetros

<i>type</i>	Estilo do componente.
-------------	-----------------------

Retorna

std::pair contendo o retângulo que limita a área do componente e o pixmap da representação gráfica do componente.

9.4.3.9 getSelectedColor()

```
QColor Diagram::getSelectedColor (
    void ) [static]
```

Setter da cor de seleção dos componentes dos objetos da classe [Diagram](#).

Método estático que retorna a cor de seleção de componentes de todos os objetos ativos e que serão criados da classe [Diagram](#).

Retorna

Cor de seleção dos componentes.

9.4.3.10 getStatus()

```
enum stats Diagram::getStatus ( )
```

Getter para o status do arquivo de um objeto da classe [Diagram](#).

Retorna o status atual do arquivo.

Retorna

Status atual do arquivo.

9.4.3.11 initializeDiagram()

```
void Diagram::initializeDiagram ( ) [private]
```

Método que inicializa os elementos gráficos da classe [Diagram](#).

Este método inicializa os elementos gráficos da classe [Diagram](#), tais como menus e botões.

Retorna

Void.

9.4.3.12 insert()

```
void Diagram::insert (
    int x,
    int y )
```

Insere um componente em um objeto da classe [Diagram](#).

Insere um objeto graficamente na janela e no membro da classe Circuit do diagrama.

Parâmetros

<i>x</i>	Coordenada x para inserção gráfica do componente.
<i>y</i>	Coordenada y para inserção gráfica do componente.

Retorna

Void.

9.4.3.13 leftButtonClicked()

```
void Diagram::leftButtonClicked (
    int x,
    int y,
    int cArea ) [private]
```

Método executado quando o o botão esquerdo do mouse é pressionado.

Este método controla as possíveis interações com os componentes gráficos quando o botão esquerdo do mouse é pressionado sobre um destes.

Parâmetros

<i>x</i>	Coordenada x do clique do mouse.
<i>y</i>	Coordenada y do clique do mouse.
<i>cArea</i>	Representa a área do componente que foi clicada (Área correspondente ao vértice 1 ou vértice 2).

Retorna

Void.

9.4.3.14 load()

```
void Diagram::load ( )
```

Carrega os dados previamente salvos para um objeto da classe [Diagram](#).

Lê os dados armazenados previamente um arquivo e transcreve estes dados para um objeto da classe [Diagram](#) e dependências.

Retorna

Void.

9.4.3.15 loadError

```
void Diagram::loadError (
    bool checked = false ) [signal]
```

Sinal emitido quando um erro de abertura de arquivo ocorre.

O sinal emite true assim que ocorre erro na abertura de algum arquivo, seja no processo de carregamento ou de salvamento.

Parâmetros

<i>checked</i>	Valor booleano indicando se houve erro na abertura de um arquivo.
----------------	---

Retorna

Void.

9.4.3.16 modified

```
void Diagram::modified (
    bool checked = false ) [signal]
```

Sinal emitido quando o diagrama é modificado.

O sinal emite true assim que um componente é inserido na tela ou uma conexão é feita.

Parâmetros

<i>checked</i>	Valor booleano indicando se o diagrama foi modificado.
----------------	--

Retorna

Void.

9.4.3.17 mouseMoveEvent()

```
void Diagram::mouseMoveEvent (
    QMouseEvent * event ) [override], [protected]
```

Sobrescrita do método mouseMovePressEvent.

Este método identifica o movimento do mouse.

Retorna

Void.

9.4.3.18 mousePressEvent()

```
void Diagram::mousePressEvent (
    QMouseEvent * event ) [override], [protected]
```

Sobrescrita do método mousePressEvent.

Este método identifica o clique do mouse.

Retorna

Void.

9.4.3.19 paintEvent()

```
void Diagram::paintEvent (
    QPaintEvent * event ) [override], [protected]
```

Sobrescrita do método paintEvent.

Este método desenha a interface gráfica referente à área de desenho do circuito.

Retorna

Void.

9.4.3.20 query

```
void Diagram::query ( ) [slot]
```

Consulta os dados obtidos após a solução do circuito.

O SLOT é conectado com o sinal emitido pelo menu de consulta. Quando o menu de consulta é selecionado, os dados acerca do componente são exibidos na tela.

Retorna

Void.

9.4.3.21 queryMode

```
void Diagram::queryMode ( ) [slot]
```

Ativa o modo de Consulta para o circuito representado no diagrama.

O SLOT é conectado com o sinal emitido pelo botão play. Assim que o botão é pressionado, o programa entra em modo de consulta.

Retorna

Void.

9.4.3.22 remove

```
void Diagram::remove ( ) [slot]
```

Remove o componente selecionado do circuito.

O SLOT é conectado com o sinal emitido pelo menu de remoção. Quando o menu de remoção é selecionado, o componente é removido do circuito e sua representação gráfica é removida da tela.

Retorna

Void.

9.4.3.23 rightButtonClicked()

```
void Diagram::rightButtonClicked (
    int x,
    int y,
    int cArea ) [private]
```

Método executado quando o o botão direito do mouse é pressionado.

Este método controla as possíveis interações com os componentes gráficos quando o botão direito do mouse é pressionado sobre um destes.

Parâmetros

<i>x</i>	Coordenada x do clique do mouse.
<i>y</i>	Coordenada y do clique do mouse.
<i>cArea</i>	Representa a área do componente que foi clicada (Área correspondente ao vértice 1 ou vértice 2).

Retorna

Void.

9.4.3.24 save()

```
void Diagram::save ( )
```

Salva os dados cruciais do objeto da classe [Diagram](#).

Salva os dados necessários do objeto e dependências no arquivo que contém o objeto.

Retorna

Void.

9.4.3.25 setBGColor()

```
void Diagram::setBGColor (
    QColor color ) [static]
```

Setter da cor do plano de fundo dos objetos da classe [Diagram](#).

Método estático para alterar a cor do plano de fundo de todos os objetos ativos e que serão criados da classe [Diagram](#).

Parâmetros

<i>color</i>	Cor para o plano de fundo.
--------------	----------------------------

Retorna

Void.

9.4.3.26 setComponentColor()

```
void Diagram::setComponentColor (
    QColor color ) [static]
```

Setter da cor dos componentes dos objetos da classe [Diagram](#).

Método estático para alterar a cor dos componentes de todos os objetos ativos e que serão criados da classe [Diagram](#).

Parâmetros

<i>color</i>	Cor para a grade do plano de fundo.
--------------	-------------------------------------

Retorna

Void.

9.4.3.27 setFileName()

```
void Diagram::setFileName (
    QString file )
```

Setter para o nome do arquivo que contém o objeto.

Altera o nome do arquivo que contém o diagrama.

Parâmetros

<i>file</i>	Nome do arquivo.
-------------	------------------

Retorna

Void.

9.4.3.28 setGridColor()

```
void Diagram::setGridColor (
    QColor color ) [static]
```

Setter da cor da grade do plano de fundo dos objetos da classe [Diagram](#).

Método estático para alterar a cor da grade do plano de fundo de todos os objetos ativos e que serão criados da classe [Diagram](#).

Parâmetros

<i>color</i>	Cor para a grade do plano de fundo.
--------------	-------------------------------------

Retorna

Void.

9.4.3.29 setSelectedButton()

```
void Diagram::setSelectedButton (
    enum cmpStyle button )
```

Setter para o botão selecionado na barra de ferramentas.

Retorna o status atual do arquivo.

Parâmetros

<i>button</i>	Botão pressionado na barra de ferramentas.
---------------	--

Retorna

Void.

9.4.3.30 setSelectedColor()

```
void Diagram::setSelectedColor (
    QColor color ) [static]
```

Setter da cor de seleção dos components dos objetos da classe [Diagram](#).

Método estático para alterar a cor de seleção de componentes de todos os objetos ativos e que serão criados da classe [Diagram](#).

Parâmetros

<i>color</i>	Cor para a seleção de componentes.
--------------	------------------------------------

Retorna

Void.

9.4.3.31 setStatus()

```
void Diagram::setStatus (
    enum stats newStatus ) [private]
```

Setter para o status do arquivo do objeto da classe [Diagram](#).

Este método altera o estado do arquivo que contém o diagrama.

Parâmetros

<i>newStatus</i>	Novo estado do arquivo.
------------------	-------------------------

Retorna

Void.

9.4.3.32 showEditDialog

```
void Diagram::showEditDialog ( ) [slot]
```

Ativa a dialog de edição.

O SLOT é conectado com o sinal emitido pelo menu de edição. Quando o menu de edição é selecionado, uma janela para a edição do valor associado ao componente é exibida.

Retorna

Void.

9.4.3.33 statusBarText

```
void Diagram::statusBarText (
    QString str ) [signal]
```

Sinal emitido quando a classe [Diagram](#) necessita exibir uma mensagem na barra de status da janela principal.

O sinal emite a string que deve ser exibida na barra de status é emitido assim que a classe entra no modo de edição ou modo de consulta.

Parâmetros

<i>str</i>	String a ser exibida na barra de status.
------------	--

Retorna

Void.

9.4.4 Atributos

9.4.4.1 backgroundColor

```
QColor Diagram::backgroundColor = QColor(DEFAULT_BGC) [static], [private]
```

Membro estático que armazena a cor do plano de fundo dos objetos da classe [Diagram](#).

9.4.4.2 circuit

```
CCT::Circuit Diagram::circuit [private]
```

Representa o circuito do diagrama.

9.4.4.3 clickedStack

```
std::stack<std::pair<int, GraphicComponent*> > Diagram::clickedStack [private]
```

Pilha que armazena o último componente selecionado.

9.4.4.4 componentColor

```
QColor Diagram::componentColor = QColor(DEFAULT_CC) [static], [private]
```

Membro estático que armazena a cor dos componentes dos objetos da classe [Diagram](#).

9.4.4.5 connections

```
GRF::adjacencyMatrix Diagram::connections [private]
```

Grafo que armazena as conexões entre os vértices dos componentes.

9.4.4.6 cursorLocation

```
QPoint Diagram::cursorLocation [private]
```

Armazena a posição do cursor do mouse.

9.4.4.7 drawList

```
std::vector<GraphicComponent*> Diagram::drawList [private]
```

Lista de componentes que devem ser desenhados na tela.

9.4.4.8 editButton

```
QPushButton* Diagram::editButton [private]
```

Representa o botão edit (entra em modo de edição).

9.4.4.9 editMenu

```
QMenu* Diagram::editMenu [private]
```

Menu de edição.

9.4.4.10 fileName

```
std::string Diagram::fileName [private]
```

Representa o nome do arquivo do diagrama.

9.4.4.11 gridColor

```
QColor Diagram::gridColor = QColor(DEFAULT_LC) [static], [private]
```

Membro estático que armazena a cor da grade do plano de fundo dos objetos da classe [Diagram](#).

9.4.4.12 mode

```
enum mode Diagram::mode [private]
```

Representa o modo atual do diagrama (QEURY/EDIT).

9.4.4.13 playButton

```
QPushButton* Diagram::playButton [private]
```

Representa o botão play (entra em modo de consulta).

9.4.4.14 queryMenu

```
QMenu* Diagram::queryMenu [private]
```

Menu de consulta.

9.4.4.15 selectedButton

```
enum cmpStyle Diagram::selectedButton [private]
```

Representa o botão selecionado.

9.4.4.16 selectedColor

```
QColor Diagram::selectedColor = QColor(DEFAULT_SC) [static], [private]
```

Membro estático que armazena a cor de seleção dos componentes dos objetos da classe [Diagram](#).

9.4.4.17 selectedComponent

```
GraphicComponent* Diagram::selectedComponent [private]
```

Representa o componente atualmente selecionado.

9.4.4.18 selectedPrev

```
QPoint Diagram::selectedPrev [private]
```

Armazena o último ponto selecionado.

9.4.4.19 status

```
enum stats Diagram::status [private]
```

Representa o estado atual do arquivo do diagrama.

9.4.4.20 vtxCounter

```
unsigned int Diagram::vtxCounter [private]
```

Contador para a quantidade de vértices inseridos.

9.4.4.21 wireCounter

```
unsigned int Diagram::wireCounter [private]
```

Contador para a quantidade de cabos inseridos.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- [CircuitSim/Diagram.h](#)
- [CircuitSim/Diagram.cpp](#)

9.5 Referência da Classe NM::EquationSystem

Declaração da classe [EquationSystem](#).

```
#include <Numeric.h>
```

Membros Públicos

- [EquationSystem](#) ([Matrix](#) a, [Matrix](#) b)
Construtor da classe [EquationSystem](#).
- [Matrix](#) [getSolution](#) (double tol, unsigned int maxIter)
Getter para a solução da classe [EquationSystem](#).

Membros Privados

- void [gaussSeidel](#) (double tol, unsigned int maxIter)
Resolve o sistema pelo método de Gauss-Seidel.
- bool [sassenfeldCriteria](#) ()
Checa se o sistema obedece ao critério de Sassenfeld.
- void [findPivot](#) (unsigned int startI)
Realiza o pivoteamento parcial do sistema.
- void [gaussJordan](#) ()
Resolve o sistema pelo método de Gauss-Jordan.

Atributos Privados

- [Matrix A](#)
Matriz dos coeficientes do sistema.
- [Matrix B](#)
Matriz (vetor) das constantes do sistema.
- [Matrix x](#)
Matriz (vetor) solução do sistema.

9.5.1 Descrição detalhada

Declaração da classe [EquationSystem](#).

Representa um sistema de equações e formas de solucioná-lo.

9.5.2 Construtores e Destrutores

9.5.2.1 EquationSystem()

```
NM::EquationSystem::EquationSystem (
    Matrix a,
    Matrix b )
```

Construtor da classe [EquationSystem](#).

Constrói um sistema de equações baseado numa matriz de coeficientes e outra de constantes.

Parâmetros

<i>a</i>	Matriz dos coeficientes.
<i>b</i>	Matriz das constantes.

9.5.3 Funções membros

9.5.3.1 findPivot()

```
void NM::EquationSystem::findPivot (
    unsigned int startI ) [private]
```

Realiza o pivoteamento parcial do sistema.

Procura o maior dentre os elementos da coluna, começando por um índice inicial. Se existir, são trocadas as linhas do índice inicial e do índice cujo elemento é o maior.

Parâmetros

<i>startI</i>	Índice inicial.
---------------	-----------------

Retorna

Void. Esse é um método provido por conveniência. Ele difere do método acima apenas na lista de argumentos que devem ser utilizados.

9.5.3.2 gaussJordan()

```
void NM::EquationSystem::gaussJordan ( ) [private]
```

Resolve o sistema pelo método de Gauss-Jordan.

Encontra a solução exata do sistema através do método de Gauss-Jordan.

Retorna

Void.

Esse é um método provido por conveniência. Ele difere do método acima apenas na lista de argumentos que devem ser utilizados.

9.5.3.3 gaussSeidel()

```
void NM::EquationSystem::gaussSeidel (
    double tol,
    unsigned int maxIter ) [private]
```

Resolve o sistema pelo método de Gauss-Seidel.

Encontra a solução aproximada iterativamente pelo método de Gauss-Seidel assumindo uma tolerância e um número máximo de iterações.

Parâmetros

<i>tol</i>	Tolerância.
<i>maxIter</i>	Número máximo de iterações.

Retorna

Void. Esse é um método provido por conveniência. Ele difere do método acima apenas na lista de argumentos que devem ser utilizados.

9.5.3.4 `getSolution()`

```
Matrix NM::EquationSystem::getSolution (
    double tol,
    unsigned int maxIter )
```

Getter para a solução da classe [EquationSystem](#).

Resolve o sistema representado pela classe [EquationSystem](#) e retorna uma matriz contendo a solução.

Parâmetros

<i>tol</i>	Tolerância para o erro aproximado.
<i>maxIter</i>	Número máximo de iterações.

9.5.3.5 `sassenfeldCriteria()`

```
bool NM::EquationSystem::sassenfeldCriteria ( ) [private]
```

Checa se o sistema obedece ao critério de Sassenfeld.

Aplica o critério de Sassenfeld ao sistema. Retorna verdadeiro se o método de Gauss-Seidel converge para uma solução e falso caso contrário.

Retorna

Se o sistema satisfaz o critério de Sassenfeld retorna 1, caso contrário, retorna 0.

Esse é um método provido por conveniência. Ele difere do método acima apenas na lista de argumentos que devem ser utilizados.

9.5.4 Atributos

9.5.4.1 `A`

```
Matrix NM::EquationSystem::A [private]
```

Matriz dos coeficientes do sistema.

9.5.4.2 B

```
Matrix NM::EquationSystem::B [private]
```

Matriz (vetor) das constantes do sistema.

9.5.4.3 x

```
Matrix NM::EquationSystem::x [private]
```

Matriz (vetor) solução do sistema.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

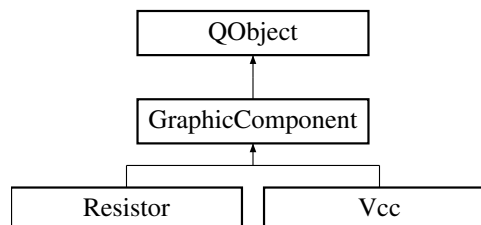
- [CircuitSim/Numeric.h](#)
- [CircuitSim/Numeric.cpp](#)

9.6 Referência da Classe GraphicComponent

Declaração da classe [GraphicComponent](#).

```
#include <GraphicComponent.h>
```

Diagrama de hierarquia para GraphicComponent:



Membros Públicos

- [GraphicComponent](#) (int *x*, int *y*, unsigned int vtx1, unsigned int vtx2, enum *orien* s, QObject *parent=nullptr)
Construtor para a classe [GraphicComponent](#).
- int [clickedArea](#) (int *x*, int *y*)
Identifica a área do componente que foi clicada.
- void [draw](#) (QPainter *painter, QColor color=QColor())
Desenha o componente na área do diagrama.
- int [getHeight](#) ()
Getter para a altura do componente.
- int [getWidth](#) ()
Getter para a largura do componente.
- QPoint [getBottom](#) ()
Getter para o ponto inferior do componente.

- QPoint [getTop](#) ()
Getter para o ponto superior do componente.
- QPoint [getLeft](#) ()
Getter para o ponto da esquerda do componente.
- QPoint [getRight](#) ()
Getter para o ponto da direita do componente.
- QRect [getBoundRect](#) ()
Getter para o retângulo limitante do componente.
- enum [orien](#) [getOrientation](#) ()
Getter para a orientação do componente.
- void [setVertex1](#) (unsigned int vtx)
Setter para o vértice 1 do componente.
- void [setVertex2](#) (unsigned int vtx)
Setter para o vértice 2 do componente.
- unsigned int [getVertex1](#) ()
Getter para o vértice 1 do componente.
- unsigned int [getVertex2](#) ()
Getter para o vértice 2 do componente.
- std::string [getLabel](#) ()
Getter para o nome de identificação do componente.
- virtual [CMP::type](#) [getType](#) ()=0
Getter para o tipo do componente.
- double [getValue](#) ()
Getter para o valor associado ao componente.
- void [setValue](#) (double newValue)
Setter para o valor associado ao componente.
- QPoint [getVertex1Point](#) ()
Getter para o ponto do vértice 1 do componente.
- QPoint [getVertex2Point](#) ()
Getter para o ponto do vértice 2 do componente.
- void [updateName](#) ()
Atualiza o nome de identificação do componente.

Atributos Protegidos

- int [x](#)
Coordenada x do desenho.
- int [y](#)
Coordenada y do desenho.
- QRect [vertexArea1](#)
Retângulo representando a área clicável referente ao vértice 1 do componente.
- QRect [vertexArea2](#)
Retângulo representando a área clicável referente ao vértice 2 do componente.
- QRect [boundRect](#)
Retângulo representando a área para desenho do componente.
- QPixmap [map](#)
Pixmap para desenho do componente.
- unsigned int [vertex1](#)
Aramazena o vértice 1 do componente.
- unsigned int [vertex2](#)

- *Aramazena o vértice 2 do componente.*
- [CMP::type componentType](#)
Aramazena o tipo do componente.
- [QString label](#)
Aramazena o nome de identificação do componente.
- [double value](#)
Aramazena o valor associado ao componente.
- [enum orien orientation](#)
Aramazena a orientação do componente.

9.6.1 Descrição detalhada

Declaração da classe [GraphicComponent](#).

Herda da classe [QObject](#). Implementa a representação gráfica de um componente e permite a interação do usuário com estes objetos gráficos.

9.6.2 Construtores e Destrutores

9.6.2.1 GraphicComponent()

```
GraphicComponent::GraphicComponent (
    int x,
    int y,
    unsigned int vtx1,
    unsigned int vtx2,
    enum orien s,
    QObject * parent = nullptr ) [explicit]
```

Construtor para a classe [GraphicComponent](#).

Constrói um objeto da classe [GraphicComponent](#). O sentido assumido para a corrente é saindo de vtx1 em direção a vtx2.

Parâmetros

<i>x</i>	Coordenada x para a inserção do componente.
<i>y</i>	Coordenada y para a inserção do componente.
<i>vtx1</i>	Vértice de saída .
<i>vtx2</i>	Vértice de entrada.
<i>s</i>	Orientação do desenho.
<i>parent</i>	Pai do objeto.

9.6.3 Funções membros

9.6.3.1 clickedArea()

```
int GraphicComponent::clickedArea (
    int x,
    int y )
```

Identifica a área do componente que foi clicada.

Identifica qual o vértice foi clicado com base na área clicável do componente.

Parâmetros

<i>x</i>	Coordenada x clicada pelo mouse.
<i>y</i>	Coordenada y clicada pelo mouse.

Retorna

Número do vértice clicado.

9.6.3.2 draw()

```
void GraphicComponent::draw (
    QPainter * painter,
    QColor color = QColor() )
```

Desenha o componente na área do diagrama.

Cada objeto tem um modo único de se desenhar. Dessa forma, é conveniente "pedir" ao objeto para se desenhar na tela.

Parâmetros

<i>painter</i>	Ponteiro par ao painter do diagrama.
<i>color</i>	Cor do desenho.

Retorna

Void.

9.6.3.3 getBottom()

```
QPoint GraphicComponent::getBottom ( )
```

Getter para o ponto inferior do componente.

Retorna o ponto Inferior localizado no eixo de simetria vertical do componente.

Retorna

Ponto inferior do componente.

9.6.3.4 `getBoundRect()`

```
QRect GraphicComponent::getBoundRect ( )
```

Getter para o retângulo limitante do componente.

Retorna o retângulo que define a área de desenho do componente.

Retorna

Retângulo limitante.

9.6.3.5 `getHeight()`

```
int GraphicComponent::getHeight ( )
```

Getter para a altura do componente.

Retorna a altura do componente em pixels.

Retorna

Altura do componente.

9.6.3.6 `getLabel()`

```
std::string GraphicComponent::getLabel ( )
```

Getter para o nome de identificação do componente.

Retorna o nome de identificação do componente.

Retorna

Nome de identificação do componente.

9.6.3.7 getLeft()

```
QPoint GraphicComponent::getLeft ( )
```

Getter para o ponto da esquerda do componente.

Retorna o ponto lateral esquerdo localizado no eixo de simetria horizontal do componente.

Retorna

Ponto esquerdo do componente.

9.6.3.8 getOrientation()

```
enum orien GraphicComponent::getOrientation ( )
```

Getter para a orientação do componente.

Retorna a orientação do componente em questão.

Retorna

Orientação do componente.

9.6.3.9 getRight()

```
QPoint GraphicComponent::getRight ( )
```

Getter para o ponto da direita do componente.

Retorna o ponto lateral direito localizado no eixo de simetria horizontal do componente.

Retorna

Ponto direito do componente.

9.6.3.10 getTop()

```
QPoint GraphicComponent::getTop ( )
```

Getter para o ponto superior do componente.

Retorna o ponto superior localizado no eixo de simetria vertical do componente.

Retorna

Ponto superior do componente.

9.6.3.11 getType()

```
virtual CMP::type GraphicComponent::getType ( ) [pure virtual]
```

Getter para o tipo do componente.

Método virtual puro que retorna o tipo do componente.

Retorna

tipo do componente.

Implementado por [Vcc](#) e [Resistor](#).

9.6.3.12 getValue()

```
double GraphicComponent::getValue ( )
```

Getter para o valor associado ao componente.

Retorna o valor associado ao componente consultado.

Retorna

Valor do componente.

9.6.3.13 getVertex1()

```
unsigned int GraphicComponent::getVertex1 ( )
```

Getter para o vértice 1 do componente.

Retorna o número do vértice 1 do componente.

Retorna

Vértice 1 do componente.

9.6.3.14 `getVertex1Point()`

```
QPoint GraphicComponent::getVertex1Point ( )
```

Getter para o ponto do vértice 1 do componente.

Retorna o ponto no eixo de simetria (horizontal ou vertical) referente ao vértice 1 do componente.

Retorna

Ponto do vértice 1 do componente.

9.6.3.15 `getVertex2()`

```
unsigned int GraphicComponent::getVertex2 ( )
```

Getter para o vértice 2 do componente.

Retorna o número do vértice 2 do componente.

Retorna

Vértice 2 do componente.

9.6.3.16 `getVertex2Point()`

```
QPoint GraphicComponent::getVertex2Point ( )
```

Getter para o ponto do vértice 2 do componente.

Retorna o ponto no eixo de simetria (horizontal ou vertical) referente ao vértice 2 do componente.

Retorna

Ponto do vértice 2 do componente.

9.6.3.17 `getWidth()`

```
int GraphicComponent::getWidth ( )
```

Getter para a largura do componente.

Retorna a largura do componente em pixels.

Retorna

Largura do componente.

9.6.3.18 `setValue()`

```
void GraphicComponent::setValue (
    double newValue )
```

Setter para o valor associado ao componente.

Altera o valor associado ao componente.

Parâmetros

<i>newValue</i>	Novo valor do componente.
-----------------	---------------------------

Retorna

Void.

9.6.3.19 setVertex1()

```
void GraphicComponent::setVertex1 (
    unsigned int vtx )
```

Setter para o vértice 1 do componente.

Altera o número do vértice 1 do componente.

Parâmetros

<i>vtx</i>	Novo número de identificação para o vértice.
------------	--

Retorna

Void.

9.6.3.20 setVertex2()

```
void GraphicComponent::setVertex2 (
    unsigned int vtx )
```

Setter para o vértice 2 do componente.

Altera o número do vértice 2 do componente.

Parâmetros

<i>vtx</i>	Novo número de identificação para o vértice.
------------	--

Retorna

Void.

9.6.3.21 updateName()

```
void GraphicComponent::updateName ( )
```

Atualiza o nome de identificação do componente.

Este método é utilizado para alterar o nome e evitar colisões ao se remover um componente da classe [Diagram](#).

Retorna

Void.

9.6.4 Atributos

9.6.4.1 boundRect

```
QRect GraphicComponent::boundRect [protected]
```

Retângulo representando a área para desenho do componente.

9.6.4.2 componentType

```
CMP::type GraphicComponent::componentType [protected]
```

Aramazena o tipo do componente.

9.6.4.3 label

```
QString GraphicComponent::label [protected]
```

Aramazena o nome de identificação do componente.

9.6.4.4 map

```
QPixmap GraphicComponent::map [protected]
```

Pixmap para desenho do componente.

9.6.4.5 orientation

```
enum orien GraphicComponent::orientation [protected]
```

Aramazena a orientação do componente.

9.6.4.6 value

```
double GraphicComponent::value [protected]
```

Aramazena o valor associado ao componente.

9.6.4.7 vertex1

```
unsigned int GraphicComponent::vertex1 [protected]
```

Aramazena o vértice 1 do componente.

9.6.4.8 vertex2

```
unsigned int GraphicComponent::vertex2 [protected]
```

Aramazena o vértice 2 do componente.

9.6.4.9 vertexArea1

```
QRect GraphicComponent::vertexArea1 [protected]
```

Retângulo representando a área clicável referente ao vértice 1 do componente.

9.6.4.10 vertexArea2

```
QRect GraphicComponent::vertexArea2 [protected]
```

Retângulo representando a área clicável referente ao vértice 2 do componente.

9.6.4.11 x

```
int GraphicComponent::x [protected]
```

Coordenada x do desenho.

9.6.4.12 y

```
int GraphicComponent::y [protected]
```

Coordenada y do desenho.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

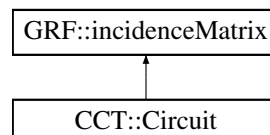
- [CircuitSim/GraphicComponent.h](#)
- [CircuitSim/GraphicComponent.cpp](#)

9.7 Referência da Classe GRF::incidenceMatrix

Declaração da classe [incidenceMatrix](#).

```
#include <Graph.h>
```

Diagrama de hierarquia para GRF::incidenceMatrix:



Membros Públicos

- [incidenceMatrix](#) ()
Construtor para a classe [incidenceMatrix](#).
- [incidenceMatrix](#) (unsigned int v, unsigned int e)
Construtor para a classe [incidenceMatrix](#).
- void [addEdge](#) (unsigned int vtx1, unsigned int vtx2)
Adiciona uma conexão entre dois vértices.
- void [makeCon](#) (unsigned int vtx1, unsigned int vtx2, unsigned int edge)
Cria uma conexão entre dois vértices em um objeto da classe [incidenceMatrix](#).
- void [removeEdge](#) (unsigned int edge)
Remove uma conexão de um objeto da classe [incidenceMatrix](#).
- void [removeVertex](#) (unsigned int vtx)
Remove uma vértice de um objeto da classe [incidenceMatrix](#).
- unsigned int [getVertexNumber](#) ()
Getter para o número de vértices de um objeto da classe [incidenceMatrix](#).

- unsigned int [getEdgeNumber](#) ()
Getter para o número de arestas de um objeto da classe [incidenceMatrix](#).
- std::vector< int > [getLoop](#) (unsigned int vtx)
Getter auxiliar para obter um ciclo de um objeto da classe [incidenceMatrix](#).
- [incidenceMatrix](#) [getSpanningTree](#) (unsigned int key)
Getter para a árvore geradora de um objeto da classe [incidenceMatrix](#).
- unsigned int [getVertexCon](#) (unsigned int vtx, unsigned int edge)
Getter para um vértice de um objeto da classe [incidenceMatrix](#).
- std::vector< unsigned int > [getEdges](#) (unsigned int vtx)
Getter para as arestas de um vértice de um objeto da classe [incidenceMatrix](#).
- std::vector< unsigned int > [getEdges](#) (unsigned int vtx1, unsigned int vtx2)
Getter para as arestas que conectam dois vértices de um objeto da classe [incidenceMatrix](#).
- std::pair< unsigned int, unsigned int > [getVertex](#) (unsigned int edge)
Getter para os vértices que são conectados por uma aresta de um objeto da classe [incidenceMatrix](#).
- unsigned int [getConNum](#) (unsigned int vtx)
Getter para o número de conexões de um vértice de um objeto da classe [incidenceMatrix](#).

Atributos Protegidos

- std::vector< std::vector< int > > [inMatrix](#)
Vector bidimensional que armazena as conexões entre os vértices.

Atributos Privados

- unsigned int [vertexNumber](#)
Armazena o número de vértices do grafo.
- unsigned int [edgeNumber](#)
Armazena o número de arestas do grafo.

9.7.1 Descrição detalhada

Declaração da classe [incidenceMatrix](#).

Representa um grafo direcionado através de uma matriz de incidência. Cada vértice do grafo é representado por um número, conforme a ordem de inserção, começando do 0. Cada aresta do grafo é representada por um número, conforme a ordem de inserção, começando do 0.

Esta classe é feita especificamente para grafos que não possuem uma aresta com mesmo vértice nos pontos final e inicial (self-loop).

9.7.2 Construtores e Destrutores

9.7.2.1 incidenceMatrix() [1/2]

```
GRF::incidenceMatrix::incidenceMatrix ( )
```

Construtor para a classe [incidenceMatrix](#).

Constrói um objeto da classe [incidenceMatrix](#) cujo vetor de conexões está inicialmente vazio.

9.7.2.2 incidenceMatrix() [2/2]

```
GRF::incidenceMatrix::incidenceMatrix (
    unsigned int v,
    unsigned int e )
```

Construtor para a classe [incidenceMatrix](#).

Constrói um objeto da classe [incidenceMatrix](#) com número de vértices e arestas predefinido.

Parâmetros

<i>v</i>	Número de vértices do grafo.
<i>e</i>	Número de arestas do grafo.

Esse é um método provido por conveniência. Ele difere do método acima apenas na lista de argumentos que devem ser utilizados.

9.7.3 Funções membros

9.7.3.1 addEdge()

```
void GRF::incidenceMatrix::addEdge (
    unsigned int vtx1,
    unsigned int vtx2 )
```

Adiciona uma conexão entre dois vértices.

Cria uma conexão entre dois vértices do grafo. A conexão é direcionada de *vtx1* para *vtx2*. Uma nova coluna é alocada dinamicamente. Se ao menos um dos vértices não existe, estes são criados até que a matriz tenha tamanho suficiente para armazená-lo.

Parâmetros

<i>vtx1</i>	Vértice de saída.
<i>vtx2</i>	Vértice de entrada.

Retorna

Void.

9.7.3.2 getConNum()

```
unsigned int GRF::incidenceMatrix::getConNum (
    unsigned int vtx )
```

Getter para o número de conexões de um vértice de um objeto da classe [incidenceMatrix](#).

Retorna o número de conexões que um vértice pertencente ao grafo possui.

Parâmetros

vtx	Vértice pertencente ao grafo.
-----	-------------------------------

Retorna

Número de conexões de um vértice.

9.7.3.3 getEdgeNumber()

```
unsigned int GRF::incidenceMatrix::getEdgeNumber ( )
```

Getter para o número de arestas de um objeto da classe [incidenceMatrix](#).

Retorna o número de arestas do grafo.

Retorna

Número de arestas do grafo.

9.7.3.4 getEdges() [1/2]

```
std::vector< unsigned int > GRF::incidenceMatrix::getEdges (
    unsigned int vtx )
```

Getter para as arestas de um vértice de um objeto da classe [incidenceMatrix](#).

Retorna todas as arestas que conectam vtx a algum outro vértice.

Parâmetros

<i>vtx</i>	Vértice pertencente ao grafo.
------------	-------------------------------

Retorna

std::vector contendo as arestas conectadas ao vértice *vtx*.

9.7.3.5 getEdges() [2/2]

```
std::vector< unsigned int > GRF::incidenceMatrix::getEdges (
    unsigned int vtx1,
    unsigned int vtx2 )
```

Getter para as arestas que conectam dois vértices de um objeto da classe [incidenceMatrix](#).

Retorna todas as arestas que conectam *vtx1* e *vtx2*.

Parâmetros

<i>vtx1</i>	Vértice pertencente ao grafo.
<i>vtx2</i>	Vértice pertencente ao grafo.

Retorna

std::vector contendo as arestas que conectam ambos os vértices.

Esse é um método provido por conveniência. Ele difere do método acima apenas na lista de argumentos que devem ser utilizados.

9.7.3.6 getLoop()

```
std::vector< int > GRF::incidenceMatrix::getLoop (
    unsigned int vtx )
```

Getter auxiliar para obter um ciclo de um objeto da classe [incidenceMatrix](#).

Este método auxiliar foi desenvolvido para identificar o ciclo de um grafo que possui apenas um loop.

O vértice inicial deve fazer parte do ciclo.

Parâmetros

<i>vtx</i>	Vértice de início da busca
------------	----------------------------

Retorna

Vector de inteiros contendo as arestas do grafo. As arestas que fazem parte de um ciclo dgrafa são marcadas com 1 se a aresta foi percorrida na direção atribuída ou -1 quando esta foi percorrida na direção oposta da aresta. As arestas marcadas com 0 não fazem parte do ciclo.

9.7.3.7 getSpanningTree()

```
incidenceMatrix GRF::incidenceMatrix::getSpanningTree (
    unsigned int key )
```

Getter para a árvore geradora de um objeto da classe [incidenceMatrix](#).

Retorna a árvore geradora do grafo.

Retorna

Objeto da classe [incidenceMatrix](#) representando a árvore geradora do grafo.

9.7.3.8 getVertex()

```
std::pair< unsigned int, unsigned int > GRF::incidenceMatrix::getVertex (
    unsigned int edge )
```

Getter para os vértices que são conectados por uma aresta de um objeto da classe [incidenceMatrix](#).

std::pair contendo os 2 vértices que estão conectados por edge. O primeiro elemento do pair é o vértice de saída enquanto que o segundo elemento do pair é o vértice de entrada da conexão. Caso não haja conexão, retorna o par (0,0).

Parâmetros

<i>edge</i>	Aresta pertencente ao grafo.
-------------	------------------------------

Retorna

std::pair contendo os 2 vértices que estão conectados por edge.

9.7.3.9 getVertexCon()

```
unsigned int GRF::incidenceMatrix::getVertexCon (
    unsigned int vtx,
    unsigned int edge )
```

Getter para um vértice de um objeto da classe [incidenceMatrix](#).

Retorna o vértice conectado a vtx pela aresta edge.

Parâmetros

<i>vtx</i>	Vértice pertencente ao grafo.
<i>edge</i>	Aresta pertencente ao grafo.

Retorna

Vértice conectado a *vtx* pela aresta *edge*.

9.7.3.10 getVertexNumber()

```
unsigned int GRF::incidenceMatrix::getVertexNumber ( )
```

Getter para o número de vértices de um objeto da classe [incidenceMatrix](#).

Retorna o número de vértices do grafo.

Retorna

Número de vértices.

9.7.3.11 makeCon()

```
void GRF::incidenceMatrix::makeCon (
    unsigned int vtx1,
    unsigned int vtx2,
    unsigned int edge )
```

Cria uma conexão entre dois vértices em um objeto da classe [incidenceMatrix](#).

Cria uma conexão entre dois vértices do grafo, direcionado de *vtx1* para *vtx2*, posicionando estas conexões na aresta especificada, este método deve ser chamado apenas para arestas que existam, mas que não possuam conexões.

Parâmetros

<i>vtx1</i>	Vértice de saída.
<i>vtx2</i>	Vértice de entrada.
<i>edge</i>	Aresta que conecta ambos os vértices.

Retorna

Void.

9.7.3.12 removeEdge()

```
void GRF::incidenceMatrix::removeEdge (
    unsigned int edge )
```

Remove uma conexão de um objeto da classe [incidenceMatrix](#).

Remove a aresta especificada existente entre dois vértices de um objeto, reduzindo o tamanho da matriz através da remoção de uma coluna.

Parâmetros

<i>edge</i>	Aresta a ser removida do grafo.
-------------	---------------------------------

Retorna

Void.

9.7.3.13 removeVertex()

```
void GRF::incidenceMatrix::removeVertex (
    unsigned int vtx )
```

Remove uma vértice de um objeto da classe [incidenceMatrix](#).

Remove o vértice especificado existente de um objeto, reduzindo o tamanho da matriz através da remoção de uma linha. Todas as conexões entre este e outros vértices também é excluído.

Parâmetros

<i>vtx</i>	Vértice a ser removido do grafo.
------------	----------------------------------

Retorna

Void.

9.7.4 Atributos

9.7.4.1 edgeNumber

```
unsigned int GRF::incidenceMatrix::edgeNumber [private]
```

Armazena o número de arestas do grafo.

9.7.4.2 inMatrix

```
std::vector<std::vector<int> > GRF::incidenceMatrix::inMatrix [protected]
```

Vector bidimensional que armazena as conexões entre os vértices.

9.7.4.3 vertexNumber

```
unsigned int GRF::incidenceMatrix::vertexNumber [private]
```

Armazena o número de vértices do grafo.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

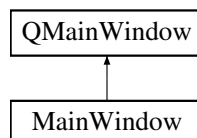
- [CircuitSim/Graph.h](#)
- [CircuitSim/Graph.cpp](#)

9.8 Referência da Classe MainWindow

Declaração da classe [MainWindow](#).

```
#include <MainWindow.h>
```

Diagrama de hierarquia para MainWindow:



Slots Públicos

- void [newFile](#) ()
Cria um novo objeto da classe [Diagram](#).
- void [openFile](#) ()
Abre um arquivo de projeto salvo anteriormente.
- void [saveFile](#) ()
Salva as alterações feitas no arquivo.
- void [saveFileAs](#) ()
Salva o arquivo conforme as especificações do usuário.
- void [setTabStatus](#) (bool modified)
Altera o status da aba.
- void [closeFile](#) (int index)
Deleta uma aba e o objeto da classe [Diagram](#) associado à ela.
- void [tutorial](#) ()
Abre o tutorial.

- void `drawRes90` ()
Seleciona o objeto a ser inserido em um objeto da classe `Diagram`.
- void `drawRes180` ()
Seleciona o objeto a ser inserido em um objeto da classe `Diagram`.
- void `drawVcc90` ()
Seleciona o objeto a ser inserido em um objeto da classe `Diagram`.
- void `drawVcc180` ()
Seleciona o objeto a ser inserido na classe `Diagram`.
- void `setBGColor` ()
Seleciona a cor do plano de fundo dos objetos da classe `Diagram`.
- void `setGridColor` ()
Seleciona a cor da grade do plano de fundo dos objetos da classe `Diagram`.
- void `setComponentColor` ()
Seleciona a cor dos componentes dos objetos da classe `Diagram`.
- void `setSelectedColor` ()
Seleciona a cor de seleção dos componentes dos objetos da classe `Diagram`.
- void `resetConfig` ()
Reseta as configurações do programa para os valores padrão.

Membros Públicos Estáticos

- static `MainWindow * getMainWindow` ()
Getter para a instância da classe `MainWindow`.

Membros Privados

- `MainWindow` (`QWidget *parent=nullptr`)
Construtor para a classe `MainWindow`.
- `MainWindow` (`const MainWindow &`)
Construtor de cópia para a classe `MainWindow`.
- `MainWindow operator=` (`MainWindow &`)
Sobrescrita do operador = pra a classe `MainWindow`.
- void `initializeMenu` ()
Método auxiliar para inicializar a barra de menus.
- void `initializeToolBar` ()
Método auxiliar para inicializar a barra de ferramentas.
- void `initializeTabs` ()
Método auxiliar para inicializar o sistema de abas.
- void `initializeStatusBar` ()
Método auxiliar para inicializar a barra de status.
- void `loadConfig` ()
Carrega as configurações do programa.
- void `saveConfig` ()
Salva as configurações do programa.

Atributos Privados

- `std::list< Diagram * > diagrams`
Código hexadecimal para a cor padrão de seleção de componentes.
- `QMenuBar * mainBar`
Barra de menus.
- `QMenu * fileMenu`
Menu para manipulação de arquivos.
- `QAction * openFileAct`
Ação para a abertura de arquivos.
- `QAction * saveFileAct`
Ação para salvar um arquivo já criado.
- `QAction * saveFileAsAct`
Ação para salvar um arquivo ainda não criado.
- `QAction * newFileAct`
Ação para criar um novo arquivo.
- `QMenu * prefMenu`
Menu para as preferências.
- `QAction * setBGColorAct`
Ação para alterar a cor do plano de fundo.
- `QAction * setGridColorAct`
Ação para alterar a cor da grade do plano de fundo.
- `QAction * setComponentColorAct`
Ação para alterar a cor dos componentes.
- `QAction * setSelectedColorAct`
Ação para alterar a cor de seleção dos componentes.
- `QAction * resetConfigAct`
Ação para resetar as configurações padrão.
- `QMenu * helpMenu`
Menu de ajuda.
- `QAction * tutorialAct`
Ação para abrir o tutorial.
- `QToolBar * toolbar`
Barra de ferramentas utilizada para inserir elementos na Classe [Diagram](#).
- `QStatusBar * statusBar`
Barra de status para informar atualizações.
- `QTabWidget * tabs`
QTabWidget para controlar as abas do programa.

Atributos Privados Estáticos

- `static MainWindow * instance = nullptr`
Ponteiro estático para instância.

9.8.1 Descrição detalhada

Declaração da classe [MainWindow](#).

A classe [MainWindow](#) é implementada com o padrão singleton e controla o fluxo de execução do programa.

9.8.2 Construtores e Destrutores

9.8.2.1 MainWindow() [1/2]

```
MainWindow::MainWindow (
    QWidget * parent = nullptr ) [explicit], [private]
```

Construtor para a classe [MainWindow](#).

Constrói um objeto da classe [MainWindow](#). O construtor é privatizado para evitar que mais de um objeto da classe seja instanciado.

Parâmetros

<i>parent</i>	Objeto pai.
---------------	-------------

9.8.2.2 MainWindow() [2/2]

```
MainWindow::MainWindow (
    const MainWindow & ) [private]
```

Construtor de cópia para a classe [MainWindow](#).

Privatiza o construtor de cópia para evitar que mais de um objeto da classe [MainWindow](#) seja instanciado.

9.8.3 Funções membros

9.8.3.1 closeFile

```
void MainWindow::closeFile (
    int index ) [slot]
```

Deleta uma aba e o objeto da classe [Diagram](#) associado à ela.

Este SLOT está conectado com o sinal de fechamento emitido pela aba, deletando o objeto caso este já tenha sido salvo, caso contrário, pede ao usuário se este deseja sair ou salvar as modificações não salvas.

Parâmetros

<i>modified</i>	Índice da aba a ser fechada.
-----------------	------------------------------

Retorna

Void.

9.8.3.2 drawRes180

```
void MainWindow::drawRes180 ( ) [slot]
```

Seleciona o objeto a ser inserido em um objeto da classe [Diagram](#).

Este SLOT esta conectado com o sinal do botão de resistor horizontal da barra de ferramentas. Ao clicar no botão, seleciona o componente desejado na aba atual.

Retorna

Void.

9.8.3.3 drawRes90

```
void MainWindow::drawRes90 ( ) [slot]
```

Seleciona o objeto a ser inserido em um objeto da classe [Diagram](#).

Este SLOT esta conectado com o sinal do botão de resistor vertical da barra de ferramentas. Ao clicar no botão, seleciona o componente desejado na aba atual.

Retorna

Void.

9.8.3.4 drawVcc180

```
void MainWindow::drawVcc180 ( ) [slot]
```

Seleciona o objeto a ser inserido na classe [Diagram](#).

Este SLOT esta conectado com o sinal do botão de Fonte horizontal da barra de ferramentas. Ao clicar no botão, seleciona o componente desejado na aba atual.

Retorna

Void.

9.8.3.5 drawVcc90

```
void MainWindow::drawVcc90 ( ) [slot]
```

Seleciona o objeto a ser inserido em um objeto da classe [Diagram](#).

Este SLOT esta conectado com o sinal do botão de Fonte vertical da barra de ferramentas. Ao clicar no botão, seleciona o componente desejado na aba atual.

Retorna

Void.

9.8.3.6 getMainWindow()

```
MainWindow * MainWindow::getMainWindow ( ) [static]
```

Getter para a instância da classe [MainWindow](#).

Este método retorna um ponteiro para a instância para o singleton.

Retorna

Ponteiro para a instância do singleton.

9.8.3.7 initializeMenu()

```
void MainWindow::initializeMenu ( ) [private]
```

Método auxiliar para inicializar a barra de menus.

Este método inicializa os componentes gráficos referentes à barra de menus.

Retorna

Void.

9.8.3.8 initializeStatusBar()

```
void MainWindow::initializeStatusBar ( ) [private]
```

Método auxiliar para inicializar a barra de status.

Este método inicializa os componentes gráficos referentes à barra de status.

Retorna

Void.

9.8.3.9 initializeTabs()

```
void MainWindow::initializeTabs ( ) [private]
```

Método auxiliar para inicializar o sistema de abas.

Este método inicializa os componentes gráficos referentes ao sistema de abas.

Retorna

Void.

9.8.3.10 initializeToolbar()

```
void MainWindow::initializeToolbar ( ) [private]
```

Método auxiliar para inicializar a barra de ferramentas.

Este método inicializa os componentes gráficos referentes à barra de ferramentas.

Retorna

Void.

9.8.3.11 loadConfig()

```
void MainWindow::loadConfig ( ) [private]
```

Carrega as configurações do programa.

Este método faz a leitura do arquivo de configurações do programa e atualiza as informações do programa conforme o arquivo carregado.

Retorna

Void.

9.8.3.12 newFile

```
void MainWindow::newFile ( ) [slot]
```

Cria um novo objeto da classe [Diagram](#).

Este SLOT está conectado com a ação referente à criação de um novo objeto da classe [Diagram](#) e adiciona uma nova aba contendo este objeto.

Retorna

Void.

9.8.3.13 openFile

```
void MainWindow::openFile ( ) [slot]
```

Abre um arquivo de projeto salvo anteriormente.

Este SLOT esta conectado com a ação `openFileAct`. Carrega um projeto salvo anteriormente e adiciona à uma nova aba.

Retorna

Void.

9.8.3.14 operator=()

```
MainWindow MainWindow::operator= (
    MainWindow & ) [private]
```

Sobrescrita do operador = pra a classe `MainWindow`.

Privatiza o operador = para evitar que mais de um objeto da classe `MainWindow` seja instanciado.

9.8.3.15 resetConfig

```
void MainWindow::resetConfig ( ) [slot]
```

Reseta as configurações do programa para os valores padrão.

Este SLOT esta conectado com o sinal do `resetConfigAct` do menu de preferências. Reseta o arquivo de configurações para os valores padrão.

Retorna

Void.

9.8.3.16 saveConfig()

```
void MainWindow::saveConfig ( ) [private]
```

Salva as configurações do programa.

Este método salva as configurações atuais em um arquivo de configuração.

Retorna

Void.

9.8.3.17 saveFile

```
void MainWindow::saveFile ( ) [slot]
```

Salva as alterações feitas no arquivo.

Este SLOT esta conectado com a ação rsaveFileAct. Caso o projeto ja tenha sido salvo anteriormente, apenas assimila as alterações feitas ao arquivo. Caso arquivo ainda não tenha sido salvo, abre uma janela para a escolha do nome e diretório do arquivo.

Retorna

Void.

9.8.3.18 saveFileAs

```
void MainWindow::saveFileAs ( ) [slot]
```

Salva o arquivo conforme as especificações do usuário.

Este SLOT esta conectado com a ação saveFileAct. Abre uma janela para a escolha do diretório e nome do arquivo em que se deseja salvar o projeto.

Retorna

Void.

9.8.3.19 setBGColor

```
void MainWindow::setBGColor ( ) [slot]
```

Seleciona a cor do plano de fundo dos objetos da classe [Diagram](#).

Este SLOT esta conectado com o sinal do setBGColorAct do menu de preferências. Abre uma janela para seleção da cor desejada e salva no arquivo de configurações.

Retorna

Void.

9.8.3.20 setComponentColor

```
void MainWindow::setComponentColor ( ) [slot]
```

Seleciona a cor dos componentes dos objetos da classe [Diagram](#).

Este SLOT esta conectado com o sinal do setComponentColorAct do menu de preferências. Abre uma janela para seleção da cor desejada e salva no arquivo de configurações.

Retorna

Void.

9.8.3.21 setGridColor

```
void MainWindow::setGridColor ( ) [slot]
```

Seleciona a cor da grade do plano de fundo dos objetos da classe [Diagram](#).

Este SLOT esta conectado com o sinal do setGridColorAct do menu de preferências. Abre uma janela para seleção da cor desejada e salva no arquivo de configurações.

Retorna

Void.

9.8.3.22 setSelectedColor

```
void MainWindow::setSelectedColor ( ) [slot]
```

Seleciona a cor de seleção dos componentes dos objetos da classe [Diagram](#).

Este SLOT esta conectado com o sinal do setSelectedColorAct do menu de preferências. Abre uma janela para seleção da cor desejada e salva no arquivo de configurações.

Retorna

Void.

9.8.3.23 setTabStatus

```
void MainWindow::setTabStatus (
    bool modified ) [slot]
```

Altera o status da aba.

Este SLOT esta conectado com a os sinais emitidos pela classe [Diagram](#). Conforme o arquivo que contenha o diagrama for alterado, altera o nome da aba para indicar se o arquivo tem alterações não salvas.

Parâmetros

<i>modified</i>	Valor booleano que indica se o objeto Diagram armazenado na aba foi modificado.
-----------------	---

Retorna

Void.

9.8.3.24 tutorial

```
void MainWindow::tutorial ( ) [slot]
```

Abre o tutorial.

Este SLOT esta conectado com o sinal de abertura de tutorial emitido pelo menu de ajuda. Abre uma janela explicando como utilizar o programa.

Retorna

Void.

9.8.4 Atributos**9.8.4.1 diagrams**

```
std::list<Diagram*> MainWindow::diagrams [private]
```

Código hexadecimal para a cor padrão de seleção de componentes.

9.8.4.2 fileMenu

```
QMenu* MainWindow::fileMenu [private]
```

Menu para manipulação de arquivos.

9.8.4.3 helpMenu

```
QMenu* MainWindow::helpMenu [private]
```

Menu de ajuda.

9.8.4.4 instance

```
MainWindow * MainWindow::instance = nullptr [static], [private]
```

Ponteiro estático para instância.

9.8.4.5 mainBar

```
QMenuBar* MainWindow::mainBar [private]
```

Barra de menus.

9.8.4.6 newFileAct

```
QAction* MainWindow::newFileAct [private]
```

Ação para criar um novo arquivo.

9.8.4.7 openFileAct

```
QAction* MainWindow::openFileAct [private]
```

Ação para a abertura de arquivos.

9.8.4.8 prefMenu

```
QMenu* MainWindow::prefMenu [private]
```

Menu para as preferências.

9.8.4.9 resetConfigAct

```
QAction* MainWindow::resetConfigAct [private]
```

Ação para resetar as configurações padrão.

9.8.4.10 saveFileAct

```
QAction* MainWindow::saveFileAct [private]
```

Ação para salvar um arquivo já criado.

9.8.4.11 saveFileAsAct

```
QAction* MainWindow::saveFileAsAct [private]
```

Ação para salvar um arquivo ainda não criado.

9.8.4.12 setBGColorAct

```
QAction* MainWindow::setBGColorAct [private]
```

Ação para alterar a cor do plano de fundo.

9.8.4.13 setComponentColorAct

```
QAction* MainWindow::setComponentColorAct [private]
```

Ação para alterar a cor dos componentes.

9.8.4.14 setGridColorAct

```
QAction* MainWindow::setGridColorAct [private]
```

Ação para alterar a cor da grade do plano de fundo.

9.8.4.15 setSelectedColorAct

```
QAction* MainWindow::setSelectedColorAct [private]
```

Ação para alterar a cor de seleção dos componentes.

9.8.4.16 statusBar

```
QStatusBar* MainWindow::statusBar [private]
```

Barra de status para informar atualizações.

9.8.4.17 tabs

```
QTabWidget* MainWindow::tabs [private]
```

QTabWidget para controlar as abas do programa.

9.8.4.18 toolbar

```
QToolBar* MainWindow::toolbar [private]
```

Barra de ferramentas utilizada para inserir elementos na Classe [Diagram](#).

9.8.4.19 tutorialAct

```
QAction* MainWindow::tutorialAct [private]
```

Ação para abrir o tutorial.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- [CircuitSim/MainWindow.h](#)
- [CircuitSim/MainWindow.cpp](#)

9.9 Referência da Classe NM::Matrix

Declaração da classe [Matrix](#).

```
#include <Numeric.h>
```

Membros Públicos

- [Matrix](#) (unsigned int r)
Construtor da classe [Matrix](#).
- [Matrix](#) (unsigned int r, unsigned int c)
Construtor da classe [Matrix](#).
- [Matrix operator*](#) ([Matrix](#) &m1)
*Sobrecarga do operador * da classe [Matrix](#).*
- void [operator*=](#) ([Matrix](#) &m1)
*Sobrecarga do operador *= da classe [Matrix](#).*
- [Matrix operator+](#) ([Matrix](#) &m1)
Sobrecarga do operador + da classe [Matrix](#).
- void [operator+=](#) ([Matrix](#) &m1)
Sobrecarga do operador += da classe [Matrix](#).
- [Matrix operator-](#) ([Matrix](#) &m1)
Sobrecarga do operador - da classe [Matrix](#).
- void [operator-=](#) ([Matrix](#) &m1)
Sobrecarga do operador -= da classe [Matrix](#).
- void [operator=](#) (const [Matrix](#) &m1)
Sobrecarga do operador = da classe [Matrix](#).
- std::vector< double > & [operator\[\]](#) (unsigned int index)
Sobrecarga do operador [] da classe [Matrix](#).
- [Matrix operator-](#) ()
Sobrecarga do operador - da classe [Matrix](#).
- [Matrix transpose](#) ()
Transpõe uma matriz.
- [Matrix Abs](#) ()
Pega o "valor absoluto" de uma matriz.
- std::vector< double > [getCol](#) (unsigned int col)
Getter para uma coluna.
- unsigned int [getColNumber](#) () const
Getter para a quantidade de colunas da matriz.
- unsigned int [getRowNumber](#) () const
Getter para a quantidade de linhas da matriz.
- void [swapLines](#) (double l1, double l2)
Troca duas linhas.

Atributos Privados

- std::vector< std::vector< double > > [realMatrix](#)
Implementação concreta da matriz.
- unsigned int [rowNumber](#)
Número de linhas da matriz.
- unsigned int [colNumber](#)
Número de colunas da matriz.

9.9.1 Descrição detalhada

Declaração da classe [Matrix](#).

Representa uma matriz matemática e implementa suas operações.

9.9.2 Construtores e Destrutores

9.9.2.1 Matrix() [1/2]

```
NM::Matrix::Matrix (
    unsigned int r )
```

Construtor da classe [Matrix](#).

Constrói uma matriz com um número escolhido linhas, preenchida com zeros.

Parâmetros

<i>r</i>	Número de linhas.
----------	-------------------

9.9.2.2 Matrix() [2/2]

```
NM::Matrix::Matrix (
    unsigned int r,
    unsigned int c )
```

Construtor da classe [Matrix](#).

Constrói uma matriz com um número escolhido de linhas e colunas, preenchida com zeros.

Parâmetros

<i>r</i>	Número de linhas.
<i>c</i>	Número de colunas.

Esse é um método provido por conveniência. Ele difere do método acima apenas na lista de argumentos que devem ser utilizados.

9.9.3 Funções membros

9.9.3.1 Abs()

```
Matrix NM::Matrix::Abs ( )
```

Pega o "valor absoluto" de uma matriz.

Faz com que todas as entradas da matriz se tornem positivas e retorna a matriz resultante.

Retorna

Matriz em termos absolutos. Esse é um método provido por conveniência. Ele difere do método acima apenas na lista de argumentos que devem ser utilizados.

9.9.3.2 getCol()

```
std::vector< double > NM::Matrix::getCol (
    unsigned int col )
```

Getter para uma coluna.

Retorna um vector contendo as entradas de uma coluna da matriz.

Parâmetros

<i>col</i>	índice da coluna.
------------	-------------------

Retorna

Vector representando a coluna. Esse é um método provido por conveniência. Ele difere do método acima apenas na lista de argumentos que devem ser utilizados.

9.9.3.3 getColNumber()

```
unsigned int NM::Matrix::getColNumber ( ) const
```

Getter para a quantidade de colunas da matriz.

Retorna a quantidade de colunas da matriz.

Retorna

Número de colunas. Esse é um método provido por conveniência. Ele difere do método acima apenas na lista de argumentos que devem ser utilizados.

9.9.3.4 getRowNumber()

```
unsigned int NM::Matrix::getRowNumber ( ) const
```

Getter para a quantidade de linhas da matriz.

Retorna a quantidade de linhas da matriz.

Retorna

Número de cinhas. Esse é um método provido por conveniência. Ele difere do método acima apenas na lista de argumentos que devem ser utilizados.

9.9.3.5 operator*()

```
Matrix NM::Matrix::operator* (
    Matrix & m1 )
```

Sobrecarga do operador * da classe [Matrix](#).

Multiplica duas matrizes e retorna o resultado.

Parâmetros

<code>&m1</code>	referência à matriz que à direita.
----------------------	------------------------------------

Retorna

Matriz à esquerda. Esse é um método provido por conveniência. Ele difere do método acima apenas na lista de argumentos que devem ser utilizados.

9.9.3.6 operator*=()

```
void NM::Matrix::operator*= (
    Matrix & m1 )
```

Sobrecarga do operador *= da classe [Matrix](#).

Multiplica duas matrizes e insere o resultado na matriz à esquerda.

Parâmetros

<code>&m1</code>	referência à matriz à direita.
----------------------	--------------------------------

Esse é um método provido por conveniência. Ele difere do método acima apenas na lista de argumentos que devem ser utilizados.

9.9.3.7 operator+()

```
Matrix NM::Matrix::operator+ (
    Matrix & m1 )
```

Sobrecarga do operador + da classe [Matrix](#).

Soma duas matrizes e retorna o resultado.

Parâmetros

<code>&m1</code>	referência à matriz à direita.
----------------------	--------------------------------

Retorna

Matriz à esquerda.

Esse é um método provido por conveniência. Ele difere do método acima apenas na lista de argumentos que devem ser utilizados.

9.9.3.8 operator+=()

```
void NM::Matrix::operator+= (
    Matrix & m1 )
```

Sobrecarga do operador += da classe [Matrix](#).

Soma as entradas de duas matrizes e insere o resultado na primeira.

Parâmetros

<i>&m1</i>	referência à matriz que à direita. Esse é um método provido por conveniência. Ele difere do método acima apenas na lista de argumentos que devem ser utilizados.
----------------	--

9.9.3.9 operator-() [1/2]

```
Matrix NM::Matrix::operator- ( )
```

Sobrecarga do operador - da classe [Matrix](#).

Multiplica todas as entradas da matriz à esquerda por -1 e retorna o resultado.

Retorna

Matriz com sinais invertidos. Esse é um método provido por conveniência. Ele difere do método acima apenas na lista de argumentos que devem ser utilizados.

9.9.3.10 operator-() [2/2]

```
Matrix NM::Matrix::operator- (
    Matrix & m1 )
```

Sobrecarga do operador - da classe [Matrix](#).

Subtrai as entradas da matriz à direita da matriz à esquerda e retorna o resultado.

Parâmetros

<i>&m1</i>	referência à matriz à direita.
----------------	--------------------------------

Retorna

Matriz à esquerda. Esse é um método provido por conveniência. Ele difere do método acima apenas na lista de argumentos que devem ser utilizados.

9.9.3.11 operator-=()

```
void NM::Matrix::operator-= (
    Matrix & m1 )
```

Sobrecarga do operador -= da classe [Matrix](#).

Subtrai as entradas matriz à direita da matriz à esquerda e insere o resultado na matriz à esquerda.

Parâmetros

<i>&m1</i>	referência à matriz à direita. Esse é um método provido por conveniência. Ele difere do método acima apenas na lista de argumentos que devem ser utilizados.
----------------	--

9.9.3.12 operator=()

```
void NM::Matrix::operator= (
    const Matrix & m1 )
```

Sobrecarga do operador = da classe [Matrix](#).

Substitui as entradas da matriz à esquerda pelas entradas da matriz à direita.

Parâmetros

<i>&m1</i>	referência à matriz que à direita. Esse é um método provido por conveniência. Ele difere do método acima apenas na lista de argumentos que devem ser utilizados.
----------------	--

9.9.3.13 operator[]()

```
std::vector< double > & NM::Matrix::operator[] (
    unsigned int index )
```

Sobrecarga do operador [] da classe [Matrix](#).

Permite acesso direto aos valores presentes na matriz.

Parâmetros

<i>index</i>	número da linha que será considerada.
--------------	---------------------------------------

Retorna

Linha do índice *index*. Esse é um método provido por conveniência. Ele difere do método acima apenas na lista de argumentos que devem ser utilizados.

9.9.3.14 swapLines()

```
void NM::Matrix::swapLines (
    double l1,
    double l2 )
```

Troca duas linhas.

Dados dois índices representando duas linhas, troca as entradas dessas duas linhas.

Parâmetros

<i>l1</i>	linha 1.
<i>l2</i>	linha 2.

Retorna

Void. Esse é um método provido por conveniência. Ele difere do método acima apenas na lista de argumentos que devem ser utilizados.

9.9.3.15 transpose()

```
Matrix NM::Matrix::transpose ( )
```

Transpõe uma matriz.

Encontra a matriz transposta e retorna o resultado.

Retorna

Matriz transposta. Esse é um método provido por conveniência. Ele difere do método acima apenas na lista de argumentos que devem ser utilizados.

9.9.4 Atributos

9.9.4.1 colNumber

```
unsigned int NM::Matrix::colNumber [private]
```

Número de colunas da matriz.

9.9.4.2 realMatrix

```
std::vector<std::vector<double> > NM::Matrix::realMatrix [private]
```

Implementação concreta da matriz.

9.9.4.3 rowNumber

```
unsigned int NM::Matrix::rowNumber [private]
```

Número de linhas da matriz.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

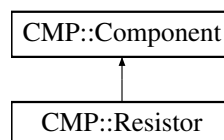
- [CircuitSim/Numeric.h](#)
- [CircuitSim/Numeric.cpp](#)

9.10 Referência da Classe CMP::Resistor

Declaração da classe Resistor.

```
#include <Component.h>
```

Diagrama de hierarquia para CMP::Resistor:



Membros Públicos

- [Resistor](#) (std::string l, double res, unsigned int vtx1, unsigned int vtx2)
Construtor para a classe [Resistor](#).
- [~Resistor](#) ()
Desconstrutor para a classe [Resistor](#).
- double [getResistance](#) ()
Getter para a resistência de um objeto da classe [Resistor](#).
- void [setResistance](#) (double res)
Setter para a resistência de um objeto da classe [Resistor](#).
- enum [type](#) [getType](#) ()
Getter para o tipo do componente da classe [Resistor](#).
- void [setCurrent](#) (double value)
Setter para a corrente do resistor.
- void [setVoltage](#) (double value)
Setter para a tensão do resistor.

Atributos Privados

- double [resistance](#)
Armazena a resistência do resistor.

Outros membros herdados

9.10.1 Descrição detalhada

Declaração da classe [Resistor](#).

Classe derivada de [Component](#). Esta classe fornece os métodos necessários para a manipulação de um objeto que representa um resistor.

9.10.2 Construtores e Destrutores

9.10.2.1 [Resistor](#)()

```
Resistor::Resistor (
    std::string l,
    double res,
    unsigned int vtx1,
    unsigned int vtx2 )
```

Construtor para a classe [Resistor](#).

Constroi um objeto da Classe [Resistor](#) com os parâmetros especificados.

Parâmetros

<i>l</i>	Nome para a identificação do resistor.
<i>res</i>	Valor da resistência do resistor.
<i>vtx1</i>	Identificação do terminal 1 do resistor.
<i>vtx2</i>	Identificação do terminal 2 do resistor.

9.10.2.2 ~Resistor()

```
Resistor::~~Resistor ( )
```

Desconstrutor para a classe [Resistor](#).

Destrói um objeto da classe [Resistor](#).

9.10.3 Funções membros**9.10.3.1 getResistance()**

```
double Resistor::getResistance ( )
```

Getter para a resistência de um objeto da classe [Resistor](#).

Retorna o valor da resistência do objeto.

Retorna

Resistência do [Resistor](#).

9.10.3.2 getType()

```
enum type Resistor::getType ( ) [virtual]
```

Getter para o tipo do componente da classe [Resistor](#).

Retorna enum type RESISTOR.

Retorna

RESISTOR.

Implementa [CMP::Component](#).

9.10.3.3 setCurrent()

```
void Resistor::setCurrent (
    double value ) [virtual]
```

Setter para a corrente do resistor.

Altera a corrente do resistor com base no parâmetro passado.

Parâmetros

<i>value</i>	Valor da corrente através do resistor.
--------------	--

Retorna

Void.

Implementa [CMP::Component](#).

9.10.3.4 setResistance()

```
void Resistor::setResistance (
    double res )
```

Setter para a resistência de um objeto da classe [Resistor](#).

Altera o valor da resistência do objeto.

Retorna

Void.

9.10.3.5 setVoltage()

```
void Resistor::setVoltage (
    double value ) [virtual]
```

Setter para a tensão do resistor.

Altera a tensão do resistor com base no parâmetro passado.

Parâmetros

<i>value</i>	Valor da corrente através do resistor.
--------------	--

Retorna

Void.

Implementa [CMP::Component](#).

9.10.4 Atributos

9.10.4.1 resistance

```
double CMP::Resistor::resistance [private]
```

Armazena a resistência do resistor.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

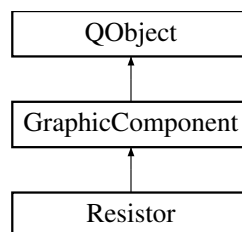
- CircuitSim/[Component.h](#)
- CircuitSim/[Component.cpp](#)

9.11 Referência da Classe Resistor

Declaração da classe [Resistor](#).

```
#include <GraphicComponent.h>
```

Diagrama de hierarquia para Resistor:



Membros Públicos

- [Resistor](#) (int `x`, int `y`, unsigned int `vtx1`, unsigned int `vtx2`, enum `orientation` `s=VERTICAL`, `QObject *parent=nullptr`)
Construtor para a classe [Resistor](#).
- `CMP::type getType ()`
Getter para o tipo do componente.

Outros membros herdados

9.11.1 Descrição detalhada

Declaração da classe [Resistor](#).

Herda da classe [Resistor](#). Implementa a representação gráfica de um resistor e permite a interação do usuário com este objeto gráfico.

9.11.2 Construtores e Destrutores

9.11.2.1 Resistor()

```
Resistor::Resistor (
    int x,
    int y,
    unsigned int vtx1,
    unsigned int vtx2,
    enum orien s = VERTICAL,
    QObject * parent = nullptr ) [explicit]
```

Construtor para a classe [Resistor](#).

Constrói um objeto da classe [Resistor](#). O sentido assumido para a corrente é saindo de vtx1 em direção a vtx2.

Parâmetros

<i>x</i>	Coordenada x para a inserção do componente.
<i>y</i>	Coordenada y para a inserção do componente.
<i>vtx1</i>	Vértice de saída .
<i>vtx2</i>	Vértice de entrada.
<i>s</i>	Orientação do desenho.
<i>parent</i>	Pai do objeto.

9.11.3 Funções membros

9.11.3.1 getType()

```
CMP::type Resistor::getType ( ) [virtual]
```

Getter para o tipo do componente.

Sobrescrita do método virtual puro da classe base que retorna o tipo do componente.

Retorna

[CMP::RESISTOR](#).

Implementa [GraphicComponent](#).

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

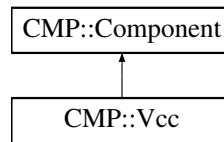
- [CircuitSim/GraphicComponent.h](#)
- [CircuitSim/GraphicComponent.cpp](#)

9.12 Referência da Classe CMP::Vcc

Declaração da classe [Vcc](#).

```
#include <Component.h>
```

Diagrama de hierarquia para CMP::Vcc:



Membros Públicos

- [Vcc](#) (std::string l, double vol, unsigned int negative, unsigned int positive)
Construtor para a classe [Vcc](#).
- [~Vcc](#) ()
Desconstrutor para a classe [Vcc](#).
- enum [type](#) [getType](#) ()
Getter para o tipo do componente da classe [Vcc](#).
- void [setCurrent](#) (double cur)
Setter para a corrente da fonte.
- void [setVoltage](#) (double vol)
Setter para a tensão do fonte.

Outros membros herdados

9.12.1 Descrição detalhada

Declaração da classe [Vcc](#).

Classe derivada de [Component](#). Esta classe fornece os métodos para a manipulação de um objeto que representa uma fonte de tensão de corrente contínua.

9.12.2 Construtores e Destrutores

9.12.2.1 Vcc()

```
Vcc::Vcc (  
    std::string l,  
    double vol,  
    unsigned int negative,  
    unsigned int positive )
```

Construtor para a classe [Vcc](#).

Constroi um objeto da Classe [Vcc](#) com os parâmetros especificados.

Parâmetros

<i>I</i>	Nome para a identificação da fonte.
<i>vol</i>	valor da tensão da fonte.
<i>vtx1</i>	Identificação do terminal negativo da fonte.
<i>vtx2</i>	Identificação do terminal positivo da fonte.

9.12.2.2 ~Vcc()

```
Vcc::~~Vcc ( )
```

Desconstrutor para a classe [Vcc](#).

Destrói um objeto da classe [Vcc](#).

9.12.3 Funções membros**9.12.3.1 getType()**

```
enum type Vcc::getType ( ) [virtual]
```

Getter para o tipo do componente da classe [Vcc](#).

Retorna enum type VCC.

Retorna

VCC.

Implementa [CMP::Component](#).

9.12.3.2 setCurrent()

```
void Vcc::setCurrent (
    double cur ) [virtual]
```

Setter para a corrente da fonte.

Altera a corrente da fonte com base no parâmetro passado.

Parâmetros

<i>value</i>	Valor da corrente através da fonte.
--------------	-------------------------------------

Retorna

Void.

Implementa [CMP::Component](#).

9.12.3.3 setVoltage()

```
void Vcc::setVoltage (
    double vol ) [virtual]
```

Setter para a tensão do fonte.

Altera a tensão da fonte com base no parâmetro passado.

Parâmetros

<i>value</i>	Valor da tensão da fonte.
--------------	---------------------------

Retorna

Void.

Implementa [CMP::Component](#).

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

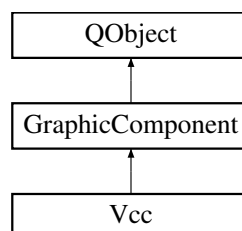
- [CircuitSim/Component.h](#)
- [CircuitSim/Component.cpp](#)

9.13 Referência da Classe Vcc

Declaração da classe [Vcc](#).

```
#include <GraphicComponent.h>
```

Diagrama de hierarquia para Vcc:



Membros Públicos

- `Vcc` (int `x`, int `y`, unsigned int `vtx1`, unsigned int `vtx2`, enum `orien` `s`=`VERTICAL`, `QObject *parent`=`nullptr`)
Construtor para a classe `Vcc`.
- `CMP::type getType` ()
Getter para o tipo do componente.

Atributos Privados Estáticos

- static unsigned int `vccCounter`

Outros membros herdados

9.13.1 Descrição detalhada

Declaração da classe `Vcc`.

Herda da classe `Vcc`. Implementa a representação gráfica de uma fonte e permite a interação do usuário com este objeto gráfico.

9.13.2 Construtores e Destrutores

9.13.2.1 `Vcc()`

```
Vcc::Vcc (
    int x,
    int y,
    unsigned int vtx1,
    unsigned int vtx2,
    enum orien s = VERTICAL,
    QObject * parent = nullptr ) [explicit]
```

Construtor para a classe `Vcc`.

Constrói um objeto da classe `Vcc`. O sentido assumido para a corrente é saindo de `vtx1` em direção a `vtx2`.

Parâmetros

<code>x</code>	Coordenada x para a inserção do componente.
<code>y</code>	Coordenada y para a inserção do componente.
<code>vtx1</code>	Vértice de saída .
<code>vtx2</code>	Vértice de entrada.
<code>s</code>	Orientação do desenho.
<code>parent</code>	Pai do objeto.

9.13.3 Funções membros

9.13.3.1 getType()

```
CMP::type Vcc::getType ( ) [virtual]
```

Getter para o tipo do componente.

Sobrescrita do método virtual puro da classe base que retorna o tipo do componente.

Retorna

[CMP::VCC](#).

Implementa [GraphicComponent](#).

9.13.4 Atributos

9.13.4.1 vccCounter

```
unsigned int Vcc::vccCounter [static], [private]
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- [CircuitSim/GraphicComponent.h](#)
- [CircuitSim/GraphicComponent.cpp](#)

Capítulo 10

Arquivos

10.1 Referência do Arquivo CircuitSim/Circuit.cpp

Implementação da classe Circuit.

```
#include "Circuit.h"  
#include "Graph.h"  
#include "Numeric.h"  
#include <vector>  
#include <string>  
#include <stack>  
#include <fstream>
```

Namespaces

- [CCT](#)

10.1.1 Descrição detalhada

Implementação da classe Circuit.

Autores

: Lucas Carvalho; Rafael Marasca Martins

Data

: 20 04 2021

Este arquivo contém as implementações dos métodos e membros da classe Circuit.

A classe Circuit fornece os métodos para se interligar objetos da classe Component e obter os valores relacionados a cada um destes.

10.2 Referência do Arquivo CircuitSim/Circuit.h

Implementação da classe Circuit.

```
#include "Graph.h"
#include "Component.h"
#include <vector>
#include <string>
```

Componentes

- class [CCT::Circuit](#)
Declaração da classe [Circuit](#).

Namespaces

- [CCT](#)

10.2.1 Descrição detalhada

Implementação da classe Circuit.

Autores

: Lucas Carvalho; Rafael Marasca Martins

Data

: 20 04 2021

Este arquivo contém as declarações dos métodos e membros da classe Circuit.

A classe Circuit fornece os métodos para se interligar objetos da classe Component e obter os valores relacionados a cada um destes.

10.3 Referência do Arquivo CircuitSim/Component.cpp

Implementação da classe Component.

```
#include "Component.h"
#include <string>
#include <iostream>
```

Namespaces

- [CMP](#)

10.3.1 Descrição detalhada

Implementação da classe Component.

Autores

: Lucas Carvalho; Rafael Marasca Martins

Data

: 30 04 2021

Este arquivo contém as implementações dos métodos e membros da classe Component.

A classe Component fornece os métodos e definições para se criar e alterar objetos da classe Component. Estes objetos são necessários para se resolver o circuito.

10.4 Referência do Arquivo CircuitSim/Component.h

Declaração da classe Component.

```
#include <string>
#include <iostream>
```

Componentes

- class [CMP::Component](#)
Declaração da classe abstrata [Component](#).
- class [CMP::Resistor](#)
Declaração da classe [Resistor](#).
- class [CMP::Vcc](#)
Declaração da classe [Vcc](#).

Namespaces

- [CMP](#)

Enumerações

- enum [CMP::type](#) { [CMP::RESISTOR](#) , [CMP::VCC](#) }
Fornece uma identificação para os tipos de componentes disponíveis no programa.

10.4.1 Descrição detalhada

Declaração da classe Component.

Autores

: Lucas Carvalho; Rafael Marasca Martins

Data

: 30 04 2021

Este arquivo contém as declarações dos métodos e membros da classe Component.

A classe Component fornece os métodos e definições para se criar e alterar objetos da classe Component. Estes objetos são necessários para se resolver o circuito.

10.5 Referência do Arquivo CircuitSim/Diagram.cpp

Implementação da classe [Diagram](#).

```
#include "Diagram.h"
#include <fstream>
#include <QPushButton>
#include <QVBoxLayout>
#include <QPainter>
#include <QScrollBar>
#include <QDebug>
#include <QMenu>
#include <QAction>
#include <QMessageBox>
#include <QInputDialog>
```

10.5.1 Descrição detalhada

Implementação da classe [Diagram](#).

Autores

: Lucas Carvalho; Rafael Marasca Martins

Data

: 30 04 2021

Este arquivo contém as implementações dos métodos e membros da classe [Diagram](#).

A classe [Diagram](#) fornece as facilidades para inserir objetos da classe GraphicsComponent graficamente, identificar as interações do usuário com estes objetos, bem como, interligar a implementação gráfica com a parte numérica.

10.6 Referência do Arquivo CircuitSim/Diagram.h

Declaração da classe Numeric.

```
#include "Circuit.h"
#include "Graph.h"
#include "GraphicComponent.h"
#include <QWidget>
#include <QObject>
#include <QScrollBar>
#include <QPushButton>
#include <QScrollArea>
#include <QMouseEvent>
#include <stack>
#include <QMessageBox>
#include <QPropertyAnimation>
```

Componentes

- class [Diagram](#)

Declaração da classe [Diagram](#).

Definições e Macros

- #define [DEFAULT_BGC](#) "#272947"
Código hexadecimal para a cor padrão do plano de fundo.
- #define [DEFAULT_LC](#) "#141516"
Código hexadecimal para a cor padrão da grade do plano de fundo.
- #define [DEFAULT_CC](#) "#FFFFFF"
Código hexadecimal para a cor padrão dos componentes.
- #define [DEFAULT_SC](#) "#0AA206"
Código hexadecimal para a cor padrão de seleção de componentes.

Enumerações

- enum [cmpStyle](#) {
 [VCC90](#) , [VCC180](#) , [RES90](#) , [RES180](#) ,
 [NONE](#) }
Fornece uma identificação para os estilos de componentes que podem ser inseridos.
- enum [stats](#) { [UNSAVED](#) , [MODIFIED](#) , [OK](#) , [ERROR](#) }
Fornece uma identificação para o estado atual do arquivo que contém um objeto da classe [Diagram](#).
- enum [mode](#) { [EDIT](#) , [QUERY](#) }
Fornece uma identificação para o modo de atual do Diagrama.

10.6.1 Descrição detalhada

Declaração da classe Numeric.

Autores

: Lucas Carvalho; Rafael Marasca Martins

Data

: 30 04 2021

Este arquivo contém as declarações dos métodos e membros da classe [Diagram](#).

A classe [Diagram](#) fornece as facilidades para inserir objetos da classe GraphicsComponent graficamente, identificar as interações do usuário com estes objetos, bem como, interligar a implementação gráfica com a parte numérica.

10.6.2 Enumerações

10.6.2.1 cmpStyle

enum [cmpStyle](#)

Fornece uma identificação para os estilos de componentes que podem ser inseridos.

Esta enumeração é utilizada em vários métodos ao longo do código, com o intuito de identificar o estilo a ser desenhado na tela.

Enumeradores

VCC90	Fonte de tensão vertical.
VCC180	Fonte de tensão horizontal.
RES90	Resistor vertical.
RES180	Resistor horizontal.
NONE	Nenhum estilo selecionado.

10.6.2.2 mode

enum [mode](#)

Fornece uma identificação para o modo de atual do Diagrama.

Esta enumeração é utilizada em vários métodos ao longo do código, com o intuito de identificar o modo de atuação do diagrama.

Enumeradores

EDIT	Modo de edição.
QUERY	Modo de consulta.

10.6.2.3 stats

```
enum stats
```

Fornece uma identificação para o estado atual do arquivo que contém um objeto da classe [Diagram](#).

Esta enumeração é utilizada em vários métodos ao longo do código, com o intuito de identificar o estado atual do arquivo.

Enumeradores

UNSAVED	Arquivo não salvo.
MODIFIED	Arquivo modificado.
OK	Arquivo salvo.
ERROR	Erro de abertura de arquivo.

10.7 Referência do Arquivo CircuitSim/Graph.cpp

Implementação das classes de grafo utilizadas no projeto.

```
#include "Graph.h"
#include <vector>
#include <stack>
```

Namespaces

- [GRF](#)

10.7.1 Descrição detalhada

Implementação das classes de grafo utilizadas no projeto.

Autores

: Lucas Carvalho; Rafael Marasca Martins

Data

: 19 04 2021

Este arquivo contém as implementações dos métodos e membros utilizados por duas classes de grafo distintas.

A classe incidenceMatrix implementa um grafo que utiliza matriz de incidência como estrutura base.

A classe adjacencyMatrix implemeta um grafo utilizando matriz de adjacência como estrutura base.

10.8 Referência do Arquivo CircuitSim/Graph.h

Declaração das classes de grafo utilizadas no projeto.

```
#include <iostream>
#include <vector>
```

Componentes

- class [GRF::incidenceMatrix](#)
Declaração da classe [incidenceMatrix](#).
- class [GRF::adjacencyMatrix](#)
Declaração da classe [adjacencyMatrix](#).

Namespaces

- [GRF](#)

10.8.1 Descrição detalhada

Declaração das classes de grafo utilizadas no projeto.

Autores

: Lucas Carvalho; Rafael Marasca Martins

Data

: 19 04 2021

Este arquivo contém as declarações dos métodos e membros utilizados por duas classes de grafo distintas.

A classe [incidenceMatrix](#) implementa um grafo que utiliza matriz de incidência como estrutura base.

A classe [adjacencyMatrix](#) implemeta um grafo utilizando matriz de adjacência como estrutura base.

10.9 Referência do Arquivo CircuitSim/GraphicComponent.cpp

Implementação da classe [GraphicComponent](#) para a inserção gráfica de componentes.

```
#include "GraphicComponent.h"
#include <QtDebug>
#include <QPen>
#include <QPoint>
#include <QPixmap>
#include <QBitmap>
```

10.9.1 Descrição detalhada

Implementação da classe [GraphicComponent](#) para a inserção gráfica de componentes.

Autores

: Lucas Carvalho; Rafael Marasca Martins

Data

: 30 04 2021

Este arquivo contém as implementações dos métodos e membros que são utilizados para inserir componentes graficamente em objeto da classe Diagrama através dos métodos da classe [Diagram](#).

10.10 Referência do Arquivo CircuitSim/GraphicComponent.h

Declaração da classe [GraphicComponent](#) para a inserção gráfica de componentes.

```
#include "Graph.h"
#include "Component.h"
#include <QObject>
#include <QRect>
#include <QPoint>
#include <QPainter>
```

Componentes

- class [GraphicComponent](#)
Declaração da classe [GraphicComponent](#).
- class [Resistor](#)
Declaração da classe [Resistor](#).
- class [Vcc](#)
Declaração da classe [Vcc](#).

Definições e Macros

- #define [HEIGHT](#) 115
- #define [WIDTH](#) 50

Enumerações

- enum [orien](#) { [VERTICAL](#) , [HORIZONTAL](#) }
- Fornece uma identificação para a orientação do desenho do componente.*

10.10.1 Descrição detalhada

Declaração da classe [GraphicComponent](#) para a inserção gráfica de componentes.

Autores

: Lucas Carvalho; Rafael Marasca Martins

Data

: 30 04 2021

Este arquivo contém as declarações necessárias para se inserir componentes graficamente em um objeto da classe [Diagram](#).

10.10.2 Enumerações

10.10.2.1 `orien`

enum `orien`

Fornece uma identificação para a orientação do desenho do componente.

Esta enumeração é utilizada em vários métodos ao longo do código, com o intuito de identificar o estilo a ser desenhado na tela.

Enumeradores

VERTICAL	Orientação vertical.
HORIZONTAL	Orientação horizontal.

10.11 Referência do Arquivo `CircuitSim/main.cpp`

```
#include "MainWindow.h"  
#include <QApplication>
```

Funções

- int `main` (int argc, char *argv[])
Implementação da função main.

10.11.1 Funções

10.11.1.1 main()

```
int main (
    int argc,
    char * argv[] )
```

Implementação da função main.

Controla o fluxo inicial e final do programa.

10.12 Referência do Arquivo CircuitSim/MainWindow.cpp

Implementação da classe [MainWindow](#).

```
#include "MainWindow.h"
#include "Diagram.h"
#include <iostream>
#include <fstream>
#include <QMenuBar>
#include <QMenu>
#include <QAction>
#include <QFileDialog>
#include <QMessageBox>
#include <QString>
#include <QDebug>
#include <QPushButton>
#include <QTabWidget>
#include <QTabBar>
#include <QFileInfo>
#include <QPixmap>
#include <QIcon>
#include <QStatusBar>
#include <QVBoxLayout>
#include <QLabel>
#include <QColorDialog>
```

10.12.1 Descrição detalhada

Implementação da classe [MainWindow](#).

Autores

: Lucas Carvalho; Rafael Marasca Martins

Data

: 28 04 2021

Este arquivo contém as implementações dos métodos e membros da classe [MainWindow](#).

A classe [MainWindow](#) controla o fluxo de execução com base nas interações do usuário.

10.13 Referência do Arquivo CircuitSim/MainWindow.h

Declaração da classe [MainWindow](#).

```
#include "Diagram.h"
#include <QMainWindow>
#include <QMenuBar>
#include <QMenu>
#include <QAction>
#include <QFileDialog>
#include <QString>
#include <QToolBar>
#include <QPushButton>
```

Componentes

- class [MainWindow](#)

Declaração da classe [MainWindow](#).

10.13.1 Descrição detalhada

Declaração da classe [MainWindow](#).

Autores

: Lucas Carvalho; Rafael Marasca Martins

Data

: 28 04 2021

Este arquivo contém as declarações dos métodos e membros da classe [MainWindow](#), que representa a janela principal do programa.

A classe [MainWindow](#) controla o fluxo de execução com base nas interações do usuário.

10.14 Referência do Arquivo CircuitSim/Numeric.cpp

Implementação das classe Matrix e EquationSystem.

```
#include "Numeric.h"
#include <algorithm>
#include <cmath>
#include <string>
```

Namespaces

- [NM](#)

10.14.1 Descrição detalhada

Implementação das classe Matrix e EquationSystem.

Autores

: Lucas Carvalho; Rafael Marasca Martins

Data

: 19 04 2021

Este arquivo contém as delcarações dos métodos e membros da classe Numeric.

A classe Numeric fornece as clsses e métodos para se criar e resolver sistemas de equações.

10.15 Referência do Arquivo CircuitSim/Numeric.h

IDeclaração das classe Matrix e EquationSystem..

```
#include "Circuit.h"  
#include <vector>  
#include <iostream>
```

Componentes

- class [NM::Matrix](#)
Declaração da classe [Matrix](#).
- class [NM::EquationSystem](#)
Declaração da classe [EquationSystem](#).

Namespaces

- [NM](#)

10.15.1 Descrição detalhada

IDeclaração das classe Matrix e EquationSystem..

Autores

: Lucas Carvalho; Rafael Marasca Martins

Data

: 19 04 2021

Este arquivo contém a declaração dos métodos e membros da classe Numeric.

A classe Numeric fornece as clsses e métodos para se criar e resolver sistemas de equações.

Índice Remissivo

- ~Circuit
 - CCT::Circuit, 22
- ~Component
 - CMP::Component, 29
- ~Resistor
 - CMP::Resistor, 97
- ~Vcc
 - CMP::Vcc, 102
- A
 - NM::EquationSystem, 54
- Abs
 - NM::Matrix, 89
- addComponent
 - CCT::Circuit, 22
- addEdge
 - GRF::incidenceMatrix, 68
- adjacencyMatrix
 - GRF::adjacencyMatrix, 18
- adjMatrix
 - GRF::adjacencyMatrix, 20
- B
 - NM::EquationSystem, 54
- backgroundColor
 - Diagram, 47
- boundRect
 - GraphicComponent, 64
- CCT, 15
- CCT::Circuit, 21
 - ~Circuit, 22
 - addComponent, 22
 - chords, 27
 - Circuit, 22
 - circuitMatrix, 27
 - components, 27
 - editComponent, 23
 - getComponentLabel, 24
 - getCurrent, 24
 - getVoltage, 24
 - initialize, 25
 - removeComponent, 25
 - reset, 26
 - Solve, 26
 - updateComponents, 26
- chords
 - CCT::Circuit, 27
- Circuit
 - CCT::Circuit, 22
- circuit
 - Diagram, 48
- circuitMatrix
 - CCT::Circuit, 27
- CircuitSim/Circuit.cpp, 107
- CircuitSim/Circuit.h, 108
- CircuitSim/Component.cpp, 108
- CircuitSim/Component.h, 109
- CircuitSim/Diagram.cpp, 110
- CircuitSim/Diagram.h, 111
- CircuitSim/Graph.cpp, 113
- CircuitSim/Graph.h, 114
- CircuitSim/GraphicComponent.cpp, 114
- CircuitSim/GraphicComponent.h, 115
- CircuitSim/main.cpp, 116
- CircuitSim/MainWindow.cpp, 117
- CircuitSim/MainWindow.h, 118
- CircuitSim/Numeric.cpp, 118
- CircuitSim/Numeric.h, 119
- clickedArea
 - GraphicComponent, 57
- clickedControl
 - Diagram, 36
- clickedStack
 - Diagram, 48
- closeFile
 - MainWindow, 77
- CMP, 15
 - RESISTOR, 16
 - type, 15
 - VCC, 16
- CMP::Component, 27
 - ~Component, 29
 - Component, 28
 - current, 32
 - getCurrent, 29
 - getLabel, 29
 - getNodes, 29
 - getType, 30
 - getVoltage, 30
 - label, 32
 - setCurrent, 30
 - setLabel, 31
 - setVoltage, 31
 - voltage, 32
 - vtxs, 32
- CMP::Resistor, 95
 - ~Resistor, 97
 - getResistance, 97

- getType, 97
 - resistance, 98
 - Resistor, 96
 - setCurrent, 97
 - setResistance, 98
 - setVoltage, 98
- CMP::Vcc, 101
 - ~Vcc, 102
 - getType, 102
 - setCurrent, 102
 - setVoltage, 103
 - Vcc, 101
- cmpStyle
 - Diagram.h, 112
- colNumber
 - NM::Matrix, 94
- Component
 - CMP::Component, 28
- componentColor
 - Diagram, 48
- components
 - CCT::Circuit, 27
- componentType
 - GraphicComponent, 64
- connections
 - Diagram, 48
- Cores padrão, 13
 - DEFAULT_BGC, 13
 - DEFAULT_CC, 13
 - DEFAULT_LC, 14
 - DEFAULT_SC, 14
- current
 - CMP::Component, 32
- cursorLocation
 - Diagram, 48
- DEFAULT_BGC
 - Cores padrão, 13
- DEFAULT_CC
 - Cores padrão, 13
- DEFAULT_LC
 - Cores padrão, 14
- DEFAULT_SC
 - Cores padrão, 14
- Diagram, 33
 - backgroundColor, 47
 - circuit, 48
 - clickedControl, 36
 - clickedStack, 48
 - componentColor, 48
 - connections, 48
 - cursorLocation, 48
 - Diagram, 36
 - drawList, 48
 - edit, 37
 - editButton, 49
 - editMenu, 49
 - editMode, 37
 - fileName, 49
 - getBGColor, 37
 - getComponentColor, 37
 - getFileName, 38
 - getGridColor, 38
 - getPixmap, 38
 - getSelectedColor, 39
 - getStatus, 39
 - gridColor, 49
 - initializeDiagram, 39
 - insert, 39
 - leftButtonClicked, 40
 - load, 40
 - loadError, 40
 - mode, 49
 - modified, 41
 - mouseMoveEvent, 41
 - mousePressEvent, 41
 - paintEvent, 42
 - playButton, 49
 - query, 42
 - queryMenu, 50
 - queryMode, 42
 - remove, 43
 - rightButtonClicked, 43
 - save, 44
 - selectedButton, 50
 - selectedColor, 50
 - selectedComponent, 50
 - selectedPrev, 50
 - setBGColor, 44
 - setComponentColor, 44
 - setFileName, 45
 - setGridColor, 45
 - setSelectedButton, 45
 - setSelectedColor, 46
 - setStatus, 46
 - showEditDialog, 47
 - status, 50
 - statusBarText, 47
 - vtxCounter, 51
 - wireCounter, 51
- Diagram.h
 - cmpStyle, 112
 - EDIT, 113
 - ERROR, 113
 - mode, 112
 - MODIFIED, 113
 - NONE, 112
 - OK, 113
 - QUERY, 113
 - RES180, 112
 - RES90, 112
 - stats, 113
 - UNSAVED, 113
 - VCC180, 112
 - VCC90, 112
- diagrams
 - MainWindow, 84

- draw
 - GraphicComponent, 58
- drawList
 - Diagram, 48
- drawRes180
 - MainWindow, 78
- drawRes90
 - MainWindow, 78
- drawVcc180
 - MainWindow, 78
- drawVcc90
 - MainWindow, 78
- edgeNumber
 - GRF::incidenceMatrix, 73
- EDIT
 - Diagram.h, 113
- edit
 - Diagram, 37
- editButton
 - Diagram, 49
- editComponent
 - CCT::Circuit, 23
- editMenu
 - Diagram, 49
- editMode
 - Diagram, 37
- EquationSystem
 - NM::EquationSystem, 52
- ERROR
 - Diagram.h, 113
- fileMenu
 - MainWindow, 84
- fileName
 - Diagram, 49
- findPivot
 - NM::EquationSystem, 52
- gaussJordan
 - NM::EquationSystem, 53
- gaussSeidel
 - NM::EquationSystem, 53
- getBGColor
 - Diagram, 37
- getBottom
 - GraphicComponent, 58
- getBoundRect
 - GraphicComponent, 59
- getCol
 - NM::Matrix, 90
- getColNumber
 - NM::Matrix, 90
- getComponentColor
 - Diagram, 37
- getComponentLabel
 - CCT::Circuit, 24
- getConNum
 - GRF::incidenceMatrix, 69
- getCurrent
 - CCT::Circuit, 24
 - CMP::Component, 29
- getEdgeNumber
 - GRF::incidenceMatrix, 69
- getEdges
 - GRF::incidenceMatrix, 69, 70
- getFileName
 - Diagram, 38
- getGridColor
 - Diagram, 38
- getHeight
 - GraphicComponent, 59
- getLabel
 - CMP::Component, 29
 - GraphicComponent, 59
- getLeft
 - GraphicComponent, 59
- getLoop
 - GRF::incidenceMatrix, 70
- getMainWindow
 - MainWindow, 79
- getNodes
 - CMP::Component, 29
- getOrientation
 - GraphicComponent, 60
- getPixMap
 - Diagram, 38
- getResistance
 - CMP::Resistor, 97
- getRight
 - GraphicComponent, 60
- getRowIndex
 - NM::Matrix, 90
- getSelectedColor
 - Diagram, 39
- getSolution
 - NM::EquationSystem, 53
- getSpanningTree
 - GRF::incidenceMatrix, 71
- getStatus
 - Diagram, 39
- getTop
 - GraphicComponent, 60
- getType
 - CMP::Component, 30
 - CMP::Resistor, 97
 - CMP::Vcc, 102
 - GraphicComponent, 60
 - Resistor, 100
 - Vcc, 105
- getValue
 - GraphicComponent, 61
- getVertex
 - GRF::incidenceMatrix, 71
- getVertex1
 - GraphicComponent, 61
- getVertex1Point

- GraphicComponent, 61
- getVertex2
 - GraphicComponent, 62
- getVertex2Point
 - GraphicComponent, 62
- getVertexCon
 - GRF::incidenceMatrix, 71
- getVertexNumber
 - GRF::adjacencyMatrix, 18
 - GRF::incidenceMatrix, 72
- getVoltage
 - CCT::Circuit, 24
 - CMP::Component, 30
- getWidth
 - GraphicComponent, 62
- GraphicComponent, 55
 - boundRect, 64
 - clickedArea, 57
 - componentType, 64
 - draw, 58
 - getBottom, 58
 - getBoundRect, 59
 - getHeight, 59
 - getLabel, 59
 - getLeft, 59
 - getOrientation, 60
 - getRight, 60
 - getTop, 60
 - getType, 60
 - getValue, 61
 - getVertex1, 61
 - getVertex1Point, 61
 - getVertex2, 62
 - getVertex2Point, 62
 - getWidth, 62
- GraphicComponent, 57
 - label, 64
 - map, 64
 - orientation, 64
 - setValue, 62
 - setVertex1, 63
 - setVertex2, 63
 - updateName, 63
 - value, 65
 - vertex1, 65
 - vertex2, 65
 - vertexArea1, 65
 - vertexArea2, 65
 - x, 65
 - y, 66
- GraphicComponent.h
 - HORIZONTAL, 116
 - orien, 116
 - VERTICAL, 116
- GRF, 16
- GRF::adjacencyMatrix, 17
 - adjacencyMatrix, 18
 - adjMatrix, 20
 - getVertexNumber, 18
 - insertEdge, 18
 - insertVertex, 19
 - query, 19
 - removeVertex, 20
 - vertexNumber, 20
- GRF::incidenceMatrix, 66
 - addEdge, 68
 - edgeNumber, 73
 - getConNum, 69
 - getEdgeNumber, 69
 - getEdges, 69, 70
 - getLoop, 70
 - getSpanningTree, 71
 - getVertex, 71
 - getVertexCon, 71
 - getVertexNumber, 72
 - incidenceMatrix, 67, 68
 - inMatrix, 73
 - makeCon, 72
 - removeEdge, 72
 - removeVertex, 73
 - vertexNumber, 74
- gridColor
 - Diagram, 49
- HEIGHT
 - Tamanho dos Componentes, 14
- helpMenu
 - MainWindow, 84
- HORIZONTAL
 - GraphicComponent.h, 116
- incidenceMatrix
 - GRF::incidenceMatrix, 67, 68
- initialize
 - CCT::Circuit, 25
- initializeDiagram
 - Diagram, 39
- initializeMenu
 - MainWindow, 79
- initializeStatusBar
 - MainWindow, 79
- initializeTabs
 - MainWindow, 79
- initializeToolbar
 - MainWindow, 80
- inMatrix
 - GRF::incidenceMatrix, 73
- insert
 - Diagram, 39
- insertEdge
 - GRF::adjacencyMatrix, 18
- insertVertex
 - GRF::adjacencyMatrix, 19
- instance
 - MainWindow, 84
- label

- CMP::Component, 32
 - GraphicComponent, 64
- leftButtonClicked
 - Diagram, 40
- load
 - Diagram, 40
- loadConfig
 - MainWindow, 80
- loadError
 - Diagram, 40
- main
 - main.cpp, 117
- main.cpp
 - main, 117
- mainBar
 - MainWindow, 85
- MainWindow, 74
 - closeFile, 77
 - diagrams, 84
 - drawRes180, 78
 - drawRes90, 78
 - drawVcc180, 78
 - drawVcc90, 78
 - fileMenu, 84
 - getMainWindow, 79
 - helpMenu, 84
 - initializeMenu, 79
 - initializeStatusBar, 79
 - initializeTabs, 79
 - initializeToolbar, 80
 - instance, 84
 - loadConfig, 80
 - mainBar, 85
 - MainWindow, 77
 - newFile, 80
 - newFileAct, 85
 - openFile, 80
 - openFileAct, 85
 - operator=, 81
 - prefMenu, 85
 - resetConfig, 81
 - resetConfigAct, 85
 - saveConfig, 81
 - saveFile, 81
 - saveFileAct, 85
 - saveFileAs, 82
 - saveFileAsAct, 86
 - setBGColor, 82
 - setBGColorAct, 86
 - setComponentColor, 82
 - setComponentColorAct, 86
 - setGridColor, 83
 - setGridColorAct, 86
 - setSelectedColor, 83
 - setSelectedColorAct, 86
 - setTabStatus, 83
 - statusBar, 86
 - tabs, 87
 - toolbar, 87
 - tutorial, 84
 - tutorialAct, 87
- makeCon
 - GRF::incidenceMatrix, 72
- map
 - GraphicComponent, 64
- Matrix
 - NM::Matrix, 89
- mode
 - Diagram, 49
 - Diagram.h, 112
- MODIFIED
 - Diagram.h, 113
- modified
 - Diagram, 41
- mouseMoveEvent
 - Diagram, 41
- mousePressEvent
 - Diagram, 41
- newFile
 - MainWindow, 80
- newFileAct
 - MainWindow, 85
- NM, 16
- NM::EquationSystem, 51
 - A, 54
 - B, 54
 - EquationSystem, 52
 - findPivot, 52
 - gaussJordan, 53
 - gaussSeidel, 53
 - getSolution, 53
 - sassenfeldCriteria, 54
 - x, 55
- NM::Matrix, 87
 - Abs, 89
 - colNumber, 94
 - getCol, 90
 - getColNumber, 90
 - getRowNumber, 90
 - Matrix, 89
 - operator*, 90
 - operator*=, 91
 - operator+, 91
 - operator+=, 92
 - operator-, 92
 - operator-=, 93
 - operator=, 93
 - operator[], 93
 - realMatrix, 95
 - rowNumber, 95
 - swapLines, 94
 - transpose, 94
- NONE
 - Diagram.h, 112
- OK

- Diagram.h, 113
- openFile
 - MainWindow, 80
- openFileAct
 - MainWindow, 85
- operator*
 - NM::Matrix, 90
- operator*=
 - NM::Matrix, 91
- operator+
 - NM::Matrix, 91
- operator+=
 - NM::Matrix, 92
- operator-
 - NM::Matrix, 92
- operator-=
 - NM::Matrix, 93
- operator=
 - MainWindow, 81
 - NM::Matrix, 93
- operator[]
 - NM::Matrix, 93
- orien
 - GraphicComponent.h, 116
- orientation
 - GraphicComponent, 64
- paintEvent
 - Diagram, 42
- playButton
 - Diagram, 49
- prefMenu
 - MainWindow, 85
- QUERY
 - Diagram.h, 113
- query
 - Diagram, 42
 - GRF::adjacencyMatrix, 19
- queryMenu
 - Diagram, 50
- queryMode
 - Diagram, 42
- realMatrix
 - NM::Matrix, 95
- remove
 - Diagram, 43
- removeComponent
 - CCT::Circuit, 25
- removeEdge
 - GRF::incidenceMatrix, 72
- removeVertex
 - GRF::adjacencyMatrix, 20
 - GRF::incidenceMatrix, 73
- RES180
 - Diagram.h, 112
- RES90
 - Diagram.h, 112
- reset
 - CCT::Circuit, 26
- resetConfig
 - MainWindow, 81
- resetConfigAct
 - MainWindow, 85
- resistance
 - CMP::Resistor, 98
- RESISTOR
 - CMP, 16
- Resistor, 99
 - CMP::Resistor, 96
 - getType, 100
 - Resistor, 99
- rightButtonClicked
 - Diagram, 43
- rowNumber
 - NM::Matrix, 95
- sassenfeldCriteria
 - NM::EquationSystem, 54
- save
 - Diagram, 44
- saveConfig
 - MainWindow, 81
- saveFile
 - MainWindow, 81
- saveFileAct
 - MainWindow, 85
- saveFileAs
 - MainWindow, 82
- saveFileAsAct
 - MainWindow, 86
- selectedButton
 - Diagram, 50
- selectedColor
 - Diagram, 50
- selectedComponent
 - Diagram, 50
- selectedPrev
 - Diagram, 50
- setBGColor
 - Diagram, 44
 - MainWindow, 82
- setBGColorAct
 - MainWindow, 86
- setComponentColor
 - Diagram, 44
 - MainWindow, 82
- setComponentColorAct
 - MainWindow, 86
- setCurrent
 - CMP::Component, 30
 - CMP::Resistor, 97
 - CMP::Vcc, 102
- setFileName
 - Diagram, 45
- setGridColor
 - Diagram, 45

- MainWindow, 83
- setGridColorAct
 - MainWindow, 86
- setLabel
 - CMP::Component, 31
- setResistance
 - CMP::Resistor, 98
- setSelectedButton
 - Diagram, 45
- setSelectedColor
 - Diagram, 46
 - MainWindow, 83
- setSelectedColorAct
 - MainWindow, 86
- setStatus
 - Diagram, 46
- setTabStatus
 - MainWindow, 83
- setValue
 - GraphicComponent, 62
- setVertex1
 - GraphicComponent, 63
- setVertex2
 - GraphicComponent, 63
- setVoltage
 - CMP::Component, 31
 - CMP::Resistor, 98
 - CMP::Vcc, 103
- showEditDialog
 - Diagram, 47
- Solve
 - CCT::Circuit, 26
- stats
 - Diagram.h, 113
- status
 - Diagram, 50
- statusBar
 - MainWindow, 86
- statusBarText
 - Diagram, 47
- swapLines
 - NM::Matrix, 94
- tabs
 - MainWindow, 87
- Tamanho dos Componentes, 14
 - HEIGHT, 14
 - WIDTH, 14
- toolbar
 - MainWindow, 87
- transpose
 - NM::Matrix, 94
- tutorial
 - MainWindow, 84
- tutorialAct
 - MainWindow, 87
- type
 - CMP, 15

- UNSAVED
 - Diagram.h, 113
- updateComponents
 - CCT::Circuit, 26
- updateName
 - GraphicComponent, 63
- value
 - GraphicComponent, 65
- VCC
 - CMP, 16
- Vcc, 103
 - CMP::Vcc, 101
 - getType, 105
 - Vcc, 104
 - vccCounter, 105
- VCC180
 - Diagram.h, 112
- VCC90
 - Diagram.h, 112
- vccCounter
 - Vcc, 105
- vertex1
 - GraphicComponent, 65
- vertex2
 - GraphicComponent, 65
- vertexArea1
 - GraphicComponent, 65
- vertexArea2
 - GraphicComponent, 65
- vertexNumber
 - GRF::adjacencyMatrix, 20
 - GRF::incidenceMatrix, 74
- VERTICAL
 - GraphicComponent.h, 116
- voltage
 - CMP::Component, 32
- vtxCounter
 - Diagram, 51
- vtxs
 - CMP::Component, 32
- WIDTH
 - Tamanho dos Componentes, 14
- wireCounter
 - Diagram, 51
- x
 - GraphicComponent, 65
 - NM::EquationSystem, 55
- y
 - GraphicComponent, 66