Quantum Circuit Simulator

Generated by Doxygen 1.12.0

1 Source content 1

2 Hierarchical Index 3

2.1 Class Hierarchy 3

3 Class Index 5

3.1 Class List 5

4 File Index 7

4.1 File List 7

5 Class Documentation 9

5.1 Button Class Reference 9

5.1.1 Detailed Description 10

5.1.2 Constructor & Destructor Documentation 10

5.1.2.1 Button() 10

5.1.2.2 ∼Button() 10

5.1.3 Member Function Documentation 11

5.1.3.1 draw() 11

5.1.3.2 getPosition() 11

5.1.3.3 isPressed() 11

5.1.3.4 isVisible() 12

5.1.3.5 moveTo() 12

5.1.3.6 setVisible() 13

5.2 CNOT Class Reference 14

5.2.1 Constructor & Destructor Documentation 16

5.2.1.1 CNOT() 16

5.2.2 Member Function Documentation 16

5.2.2.1 to_string() 16

5.3 Gate Class Reference 17

5.3.1 Detailed Description 19

5.3.2 Constructor & Destructor Documentation 19

5.3.2.1 ∼Gate() 19

5.3.2.2 Gate() 19

5.3.3 Member Function Documentation 19

5.3.3.1 get_controls() 19

5.3.3.2 get_matrix() 19

5.3.3.3 get_qubits() 20

5.3.3.4 num_qubits() 20

5.3.3.5 to_string() 20

5.3.4 Member Data Documentation 21

5.3.4.1 controls 21

5.3.4.2 qubits 21

5.4 H Class Reference 21

5.4.1 Constructor & Destructor Documentation 24
5.4.1.1 H() 24
5.4.2 Member Function Documentation 24
5.4.2.1 to_string() 24
5.5 PauliX Class Reference 25
5.5.1 Detailed Description 27
5.5.2 Constructor & Destructor Documentation 27
5.5.2.1 PauliX() 27
5.5.2.2 ∼PauliX() 28
5.5.3 Member Function Documentation 28
5.5.3.1 to_string() 28
5.6 PauliY Class Reference 28
5.6.1 Detailed Description 31
5.6.2 Constructor & Destructor Documentation 31
5.6.2.1 PauliY() 31
5.6.2.2 ∼PauliY() 32
5.6.3 Member Function Documentation 32
5.6.3.1 to_string() 32
5.7 PauliZ Class Reference 32
5.7.1 Detailed Description 35
5.7.2 Constructor & Destructor Documentation 35
5.7.2.1 PauliZ() 35
5.7.2.2 ∼PauliZ() 36
5.7.3 Member Function Documentation 36
5.7.3.1 to_string() 36
5.8 PlaceholderGate Class Reference 36
5.8.1 Detailed Description 39
5.8.2 Constructor & Destructor Documentation 39
5.8.2.1 PlaceholderGate() [1/2] 39
5.8.2.2 PlaceholderGate() [2/2] 39
5.8.2.3 ∼PlaceholderGate() 40
5.8.3 Member Function Documentation 40
5.8.3.1 draw() 40
5.8.3.2 getSize() 40
5.8.3.3 isVisible() 41
5.8.3.4 moveTo() 41
5.8.3.5 setVisible() 41
5.9 QuantumCircuit Class Reference 42
5.9.1 Detailed Description 43
5.9.2 Constructor & Destructor Documentation 43
5.9.2.1 QuantumCircuit() [1/2] 43
5.9.2.2 QuantumCircuit() [2/2] 43

5.9.3 Member Function Documentation 43
5.9.3.1 addGate() 43
5.9.3.2 addQubit() 44
5.9.3.3 getGates() 44
5.9.3.4 getQubits() 45
5.10 Result Class Reference 46
5.10.1 Detailed Description 46
5.10.2 Constructor & Destructor Documentation 47
5.10.2.1 Result() [1/2] 47
5.10.2.2 Result() [2/2] 47
5.10.2.3 ∼Result() 47
5.10.3 Member Function Documentation 47
5.10.3.1 draw() 47
5.10.3.2 moveTo() 48
5.10.3.3 operator=() 48
5.11 RotationGate Class Reference 48
5.11.1 Detailed Description 51
5.11.2 Constructor & Destructor Documentation 51
5.11.2.1 ∼RotationGate() 51
5.11.2.2 RotationGate() 51
5.11.3 Member Function Documentation 52
5.11.3.1 to_string() 52
5.12 Simulator Class Reference 52
5.12.1 Detailed Description 53
5.12.2 Constructor & Destructor Documentation 53
5.12.2.1 Simulator() 53
5.12.3 Member Function Documentation 53
5.12.3.1 run() 53
5.13 VisualCNOT Class Reference 54
5.13.1 Detailed Description 55
5.13.2 Constructor & Destructor Documentation 55
5.13.2.1 VisualCNOT() 55
5.13.2.2 ∼VisualCNOT() 56
5.13.3 Member Function Documentation 56
5.13.3.1 draw() 56
5.13.3.2 getControlPosition() 56
5.13.3.3 getTargetPosition() 57
5.14 VisualGate Class Reference 57
5.14.1 Detailed Description 60
5.14.2 Constructor & Destructor Documentation 60
5.14.2.1 VisualGate() 60
5.14.2.2 ∼VisualGate() 61

5.14.3 Member Function Documentation 61
5.14.3.1 draw() 61
5.14.3.2 getSelected() 61
5.14.3.3 moveTo() 62
5.14.3.4 setSelected() 62
5.15 VisualGateAbstract Class Reference 63
5.15.1 Detailed Description 65
5.15.2 Constructor & Destructor Documentation 65
5.15.2.1 VisualGateAbstract() 65
5.15.2.2 ∼VisualGateAbstract() 65
5.15.3 Member Function Documentation 65
5.15.3.1 draw() 65
5.15.3.2 getPosition() 65
5.15.3.3 isPressed() 66
5.15.3.4 moveTo() 67
5.15.4 Member Data Documentation 67
5.15.4.1 gate_ 67
5.15.4.2 size_ 67
5.16 VisualQubit Class Reference 67
5.16.1 Detailed Description 69
5.16.2 Constructor & Destructor Documentation 69
5.16.2.1 VisualQubit() 69
5.16.2.2 ∼VisualQubit() 70
5.16.3 Member Function Documentation 70
5.16.3.1 addCNOTGate() 70
5.16.3.2 addGate() 71
5.16.3.3 draw() 71
5.16.3.4 getGates() 72
5.16.3.5 getID() 72
5.16.3.6 getInitialState() 72
5.16.3.7 getPlaceholderPosition() 72
5.16.3.8 isInitialStageClicked() 72
5.16.3.9 isPlaceholderClicked() 73
5.16.3.10 movePlaceholder() 73
5.16.3.11 operator=() 74
5.16.3.12 resetQubit() 74
5.16.3.13 switchInitialState() 74

6 File Documentation **75**
6.1 src/controller/QuantumCircuit.hpp File Reference 75
6.2 QuantumCircuit.hpp 76
6.3 src/controller/Simulator.hpp File Reference 77

6.4 Simulator.hpp 77

6.5 src/gates/CNOT.hpp File Reference 78

6.6 CNOT.hpp 79

6.7 src/gates/Gate.hpp File Reference 79

6.8 Gate.hpp 80

6.9 src/gates/Hadamard.hpp File Reference 81

6.10 Hadamard.hpp 82

6.11 src/gates/PauliGates.hpp File Reference 83

6.11.1 Macro Definition Documentation 84

6.11.1.1 _USE_MATH_DEFINES 84

6.12 PauliGates.hpp 85

6.13 src/gates/RotationGate.hpp File Reference 85

6.13.1 Enumeration Type Documentation 86

6.13.1.1 Axis 86

6.13.2 Function Documentation 87

6.13.2.1 roundWithPrecision() 87

6.14 RotationGate.hpp 87

6.15 src/io/File-io.hpp File Reference 88

6.15.1 Typedef Documentation 89

6.15.1.1 json 89

6.15.2 Function Documentation 89

6.15.2.1 readCircuitFromFile() 89

6.15.2.2 writeCircuitToFile() 89

6.16 File-io.hpp 90

6.17 src/main.cpp File Reference 91

6.17.1 Function Documentation 92

6.17.1.1 main() 92

6.17.2 Variable Documentation 93

6.17.2.1 cnotControl 93

6.17.2.2 controlQubit 94

6.17.2.3 fileFilterPatterns 94

6.17.2.4 font 94

6.17.2.5 gates 94

6.17.2.6 gateSelected 94

6.17.2.7 qubits 94

6.17.2.8 windowHeight 94

6.17.2.9 windowWidth 94

6.18 src/readme.md File Reference 95

6.19 src/view/Button.hpp File Reference 95

6.20 Button.hpp 95

6.21 src/view/PlaceholderGate.hpp File Reference 96

6.22 PlaceholderGate.hpp 97

6.23 src/view/Result.hpp File Reference 98
6.24 Result.hpp 99
6.25 src/view/VisualCNOT.hpp File Reference 100
6.26 VisualCNOT.hpp 101
6.27 src/view/VisualGate.hpp File Reference 101
6.28 VisualGate.hpp 102
6.29 src/view/VisualGateAbstract.hpp File Reference 103
6.30 VisualGateAbstract.hpp 104
6.31 src/view/VisualQubit.hpp File Reference 105
6.32 VisualQubit.hpp 105

Index **109**

Chapter 1

Source content

This folder should contain only hpp/cpp files of your implementation. You can also place hpp files in a separate directory `include`.

You can create a summary of files here. It might be useful to describe file relations, and brief summary of their content.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

- Button 9
- Eigen::MatrixXcd
  - Gate 17
    - CNOT 14
    - H 21
    - RotationGate 48
      - PauliX 25
      - PauliY 28
      - PauliZ 32
- QuantumCircuit 42
- Result 46
- Simulator 52
- VisualCNOT 54
- VisualGateAbstract 63
  - PlaceholderGate 36
  - VisualGate 57
- VisualQubit 67

Chapter 3

Class Index

3.1 Class List

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

Button
A class visualizing a button in GUI 9

CNOT 14

Gate
A class representing a quantum gate, inheriting from Eigen::MatrixXcd 17

H 21

PauliX
A class representing a quantum rotation gate around the X axis axis by an angle of pi inheriting from the RotationGate class 25

PauliY
A class representing a quantum rotation gate around the Y axis axis by an angle of pi inheriting from the RotationGate class 28

PauliZ
A class representing a quantum rotation gate around the Z axis axis by an angle of pi inheriting from the RotationGate class Matrix form: (1,0) (0,0) (0,0) (-1,-0) 32

PlaceholderGate
A class visualizing a placeholder gate in GUI 36

QuantumCircuit
A class representing a quantum circuit with a set of quantum gates 42

Result
A class visualizing the result of the quantum computer simulation 46

RotationGate
A class representing a quantum rotation gate around a set axis by an arbitrary angle, inheriting from the Gate class 48

Simulator
Representing a simulator object, would be a quantum computer in the real world 52

VisualCNOT
A class visualizing a CNOT gate in GUI 54

VisualGate
A class visualizing a quantum gate in GUI 57

VisualGateAbstract
An abstract class for visual gate-like classes 63

VisualQubit
A class visualizing a qubit in GUI 67

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

src/main.cpp 91
src/controller/QuantumCircuit.hpp 75
src/controller/Simulator.hpp 77
src/gates/CNOT.hpp 78
src/gates/Gate.hpp 79
src/gates/Hadamard.hpp 81
src/gates/PauliGates.hpp 83
src/gates/RotationGate.hpp 85
src/io/File-io.hpp 88
src/view/Button.hpp 95
src/view/PlaceholderGate.hpp 96
src/view/Result.hpp 98
src/view/VisualCNOT.hpp 100
src/view/VisualGate.hpp 101
src/view/VisualGateAbstract.hpp 103
src/view/VisualQubit.hpp 105

Chapter 5

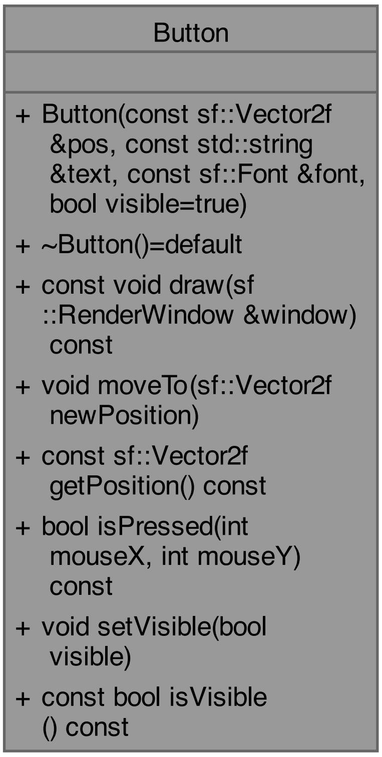
Class Documentation

5.1 Button Class Reference

A class visualizing a button in GUI.

```
#include <Button.hpp>
```

Collaboration diagram for Button:


Button
+ Button(const sf::Vector2f &pos, const std::string &text, const sf::Font &font, bool visible=true)
+ ~Button()=default
+ const void draw(sf ::RenderWindow &window) const
+ void moveTo(sf::Vector2f newPosition)
+ const sf::Vector2f getPosition() const
+ bool isPressed(int mouseX, int mouseY) const
+ void setVisible(bool visible)
+ const bool isVisible () const

Public Member Functions

- Button (const sf::Vector2f &pos, const std::string &text, const sf::Font &font, bool visible=true)

  Constructs a Button with a specified position and text of the button.
- ∼Button ()=default

  Default destructor for the Button class.
- const void draw (sf::RenderWindow &window) const

  Draws the button with the text to the window.
- void moveTo (sf::Vector2f newPosition)

  Moves the button to a specified position.
- const sf::Vector2f getPosition () const

  Getter for the position of the button.
- bool isPressed (int mouseX, int mouseY) const

  Checks if the mouse click happened inside the button.
- void setVisible (bool visible)

  Setter for 'visible_' variable.
- const bool isVisible () const

  Getter for 'visible_' variable.

5.1.1 Detailed Description

A class visualizing a button in GUI.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Button()

```
Button::Button (
            const sf::Vector2f & pos,
            const std::string & text,
            const sf::Font & font,
            bool visible = true)  [inline]
```

Constructs a Button with a specified position and text of the button.

Initializes a rectangleShape to specified position with specified text inside of the rectangleShape.

Parameters

pos	Position where the button should be drawn in GUI.
text	Text that will be visible inside the button.
font	The font that will be used for the button texts.
visible	Determines whether the button will be visible in UI, delauts to true.

5.1.2.2 ∼Button()

```
Button::∼Button ()  [default]
```

Default destructor for the Button class.

5.1.3 Member Function Documentation

5.1.3.1 draw()

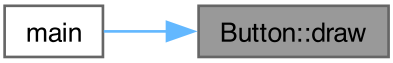
```
const void Button::draw (
             sf::RenderWindow & window) const  [inline]
```

Draws the button with the text to the window.

Parameters

window	Window where the button will be drawn.

Here is the caller graph for this function:



5.1.3.2 getPosition()

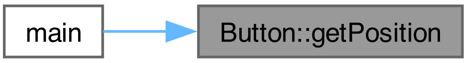
```
const sf::Vector2f Button::getPosition () const  [inline]
```

Getter for the position of the button.

Returns

2D vector of the position.

Here is the caller graph for this function:



5.1.3.3 isPressed()

```
bool Button::isPressed (
             int mouseX,
             int mouseY) const  [inline]
```

Checks if the mouse click happened inside the button.

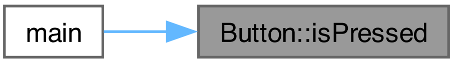
Parameters

mouseX	Mouse position on x axis.
mouseY	Mouse position on y axis.

Returns

True if the click was inside the button rectangle, otherwise false.

Here is the caller graph for this function:



5.1.3.4 isVisible()

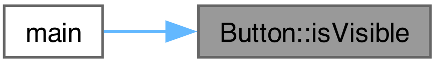
```
const bool Button::isVisible () const  [inline]
```

Getter for 'visible_' variable.

Returns

True or false, depending on the value of 'visible_'.

Here is the caller graph for this function:



5.1.3.5 moveTo()

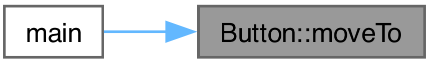
```
void Button::moveTo (
             sf::Vector2f newPosition)  [inline]
```

Moves the button to a specified position.

Parameters

newPosition	Vector of the position the button will be moved.

Here is the caller graph for this function:



5.1.3.6 setVisible()

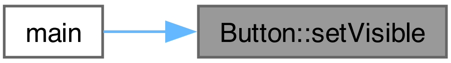
```
void Button::setVisible (
             bool visible)  [inline]
```

Setter for 'visible_' variable.

Parameters

visible	the value to be set

Here is the caller graph for this function:



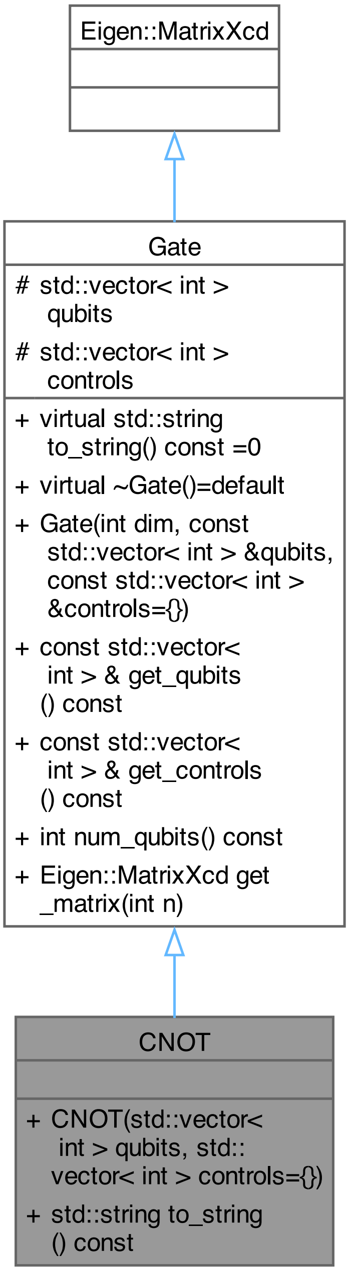
The documentation for this class was generated from the following file:

- src/view/Button.hpp

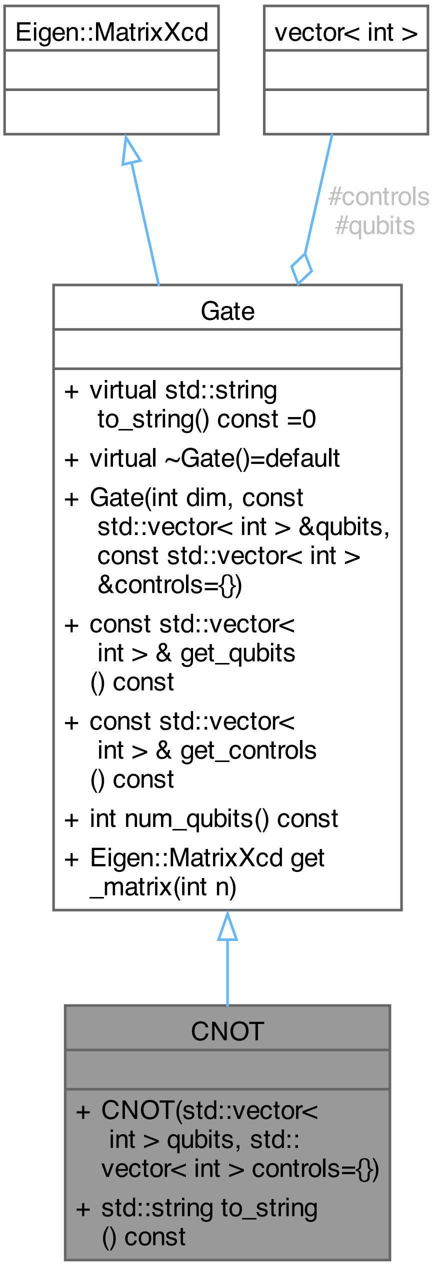
5.2 CNOT Class Reference

```
#include <CNOT.hpp>
```

Inheritance diagram for CNOT:


Eigen::MatrixXcd
Gate
# std::vector< int > qubits
# std::vector< int > controls
+ virtual std::string to_string() const =0
+ virtual ~Gate()=default
+ Gate(int dim, const std::vector< int > &qubits, const std::vector< int > &controls={})
+ const std::vector< int > & get_qubits () const
+ const std::vector< int > & get_controls () const
+ int num_qubits() const
+ Eigen::MatrixXcd get _matrix(int n)
CNOT
+ CNOT(std::vector< int > qubits, std:: vector< int > controls={})
+ std::string to_string () const

Collaboration diagram for CNOT:


Eigen::MatrixXcd
vector< int >
#controls
#qubits
Gate
+ virtual std::string to_string() const =0
+ virtual ~Gate()=default
+ Gate(int dim, const std::vector< int > &qubits, const std::vector< int > &controls={})
+ const std::vector< int > & get_qubits () const
+ const std::vector< int > & get_controls () const
+ int num_qubits() const
+ Eigen::MatrixXcd get _matrix(int n)
CNOT
+ CNOT(std::vector< int > qubits, std:: vector< int > controls={})
+ std::string to_string () const

Public Member Functions

- CNOT (std::vector< int > qubits, std::vector< int > controls={})

  Constructs a CNOT gate with specified target and control qubits.
- std::string to_string () const

Public Member Functions inherited from Gate

- virtual ∼Gate ()=default

    Virtual destructor for the Gate class.
- Gate (int dim, const std::vector< int > &qubits, const std::vector< int > &controls={})

    Constructs a Gate with a specified dimension.
- const std::vector< int > & get_qubits () const

    Returns the qubits the gate acts on.
- const std::vector< int > & get_controls () const

    Returns the control qubits for the gate.
- int num_qubits () const

    Returns the number of qubits the gate acts on.
- Eigen::MatrixXcd get_matrix (int n)

    Generates the unitary matrix for the gate, supporting multi-control gates.

Additional Inherited Members

Protected Attributes inherited from Gate

- std::vector< int > qubits
- std::vector< int > controls

5.2.1 Constructor & Destructor Documentation

5.2.1.1 CNOT()

```
CNOT::CNOT (
            std::vector< int > qubits,
            std::vector< int > controls = {})  [inline]
```

Constructs a CNOT gate with specified target and control qubits.

Initializes a 2x2 matrix representing the CNOT gate with the specified target and control qubits.

Parameters

qubits	The target and control qubits.
controls	The control qubits, defaults to an empty vector.

5.2.2 Member Function Documentation

5.2.2.1 to_string()

```
std::string CNOT::to_string () const  [inline], [virtual]
```

Implements Gate.

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

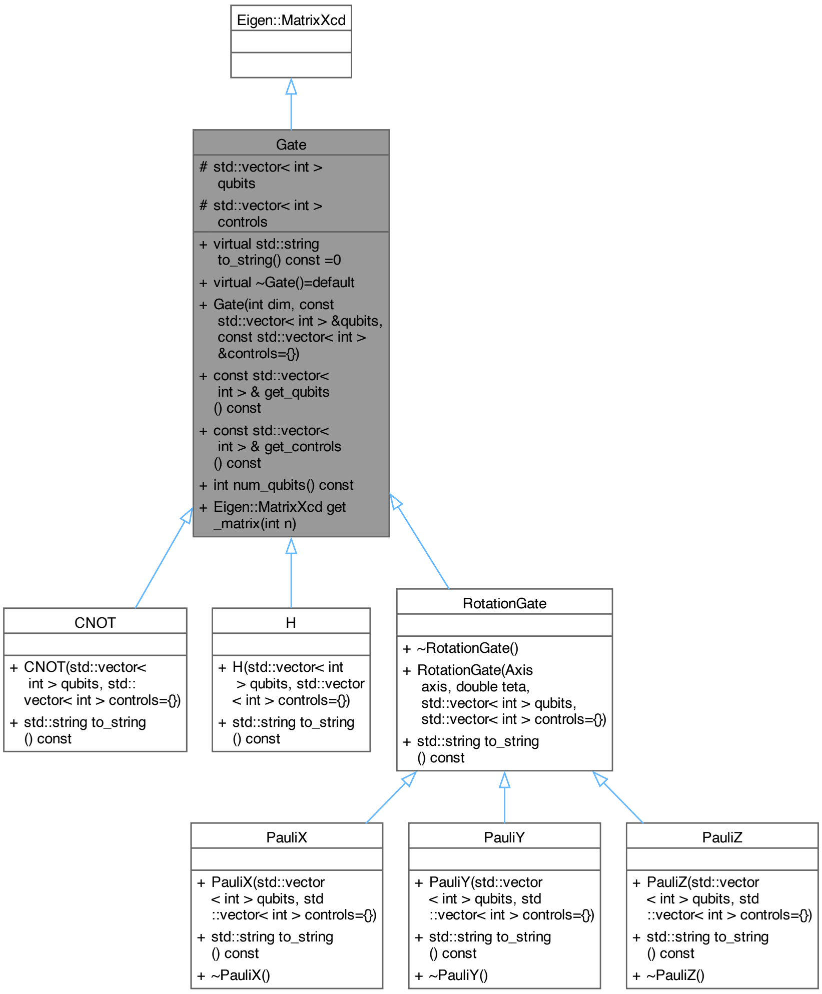
- src/gates/CNOT.hpp

5.3 Gate Class Reference

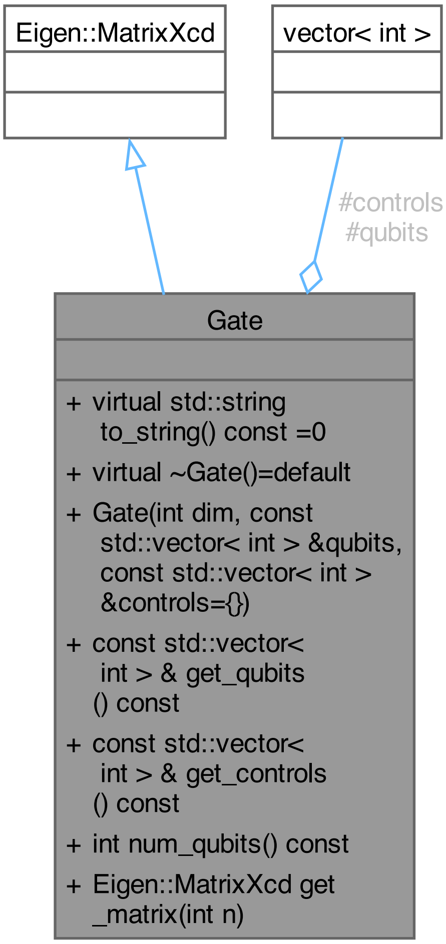
A class representing a quantum gate, inheriting from Eigen::MatrixXcd.

```
#include <Gate.hpp>
```

Inheritance diagram for Gate:


Eigen::MatrixXcd
Gate
# std::vector< int > qubits
# std::vector< int > controls
+ virtual std::string to_string() const =0
+ virtual ~Gate()=default
+ Gate(int dim, const std::vector< int > &qubits, const std::vector< int > &controls={})
+ const std::vector< int > & get_qubits () const
+ const std::vector< int > & get_controls () const
+ int num_qubits() const
+ Eigen::MatrixXcd get _matrix(int n)
CNOT
+ CNOT(std::vector< int > qubits, std:: vector< int > controls={})
+ std::string to_string () const
H
+ H(std::vector< int > qubits, std::vector < int > controls={})
+ std::string to_string () const
RotationGate
+ ~RotationGate()
+ RotationGate(Axis axis, double teta, std::vector< int > qubits, std::vector< int > controls={})
+ std::string to_string () const
PauliX
+ PauliX(std::vector < int > qubits, std ::vector< int > controls={})
+ std::string to_string () const
+ ~PauliX()
PauliY
+ PauliY(std::vector < int > qubits, std ::vector< int > controls={})
+ std::string to_string () const
+ ~PauliY()
PauliZ
+ PauliZ(std::vector < int > qubits, std ::vector< int > controls={})
+ std::string to_string () const
+ ~PauliZ()

Collaboration diagram for Gate:


Eigen::MatrixXcd
vector< int >
#controls
#qubits
Gate
+ virtual std::string to_string() const =0
+ virtual ~Gate()=default
+ Gate(int dim, const std::vector< int > &qubits, const std::vector< int > &controls={})
+ const std::vector< int > & get_qubits () const
+ const std::vector< int > & get_controls () const
+ int num_qubits() const
+ Eigen::MatrixXcd get _matrix(int n)

Public Member Functions

- virtual std::string to_string () const =0
- virtual ∼Gate ()=default

    Virtual destructor for the Gate class.

- Gate (int dim, const std::vector< int > &qubits, const std::vector< int > &controls={})

    Constructs a Gate with a specified dimension.

- const std::vector< int > & get_qubits () const

    Returns the qubits the gate acts on.

- const std::vector< int > & get_controls () const

    Returns the control qubits for the gate.

- int num_qubits () const

    Returns the number of qubits the gate acts on.

- Eigen::MatrixXcd get_matrix (int n)

    Generates the unitary matrix for the gate, supporting multi-control gates.

Protected Attributes

- std::vector< int > qubits
- std::vector< int > controls

5.3.1 Detailed Description

A class representing a quantum gate, inheriting from Eigen::MatrixXcd.

The Gate class extends Eigen's `MatrixXcd` class to represent complex-valued square matrices. This class initializes a matrix with a specified dimension and sets all elements to zero by default.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 ∼Gate()

```
virtual Gate::∼Gate ()  [virtual], [default]
```

Virtual destructor for the Gate class.

5.3.2.2 Gate()

```
Gate::Gate (
              int dim,
              const std::vector< int > & qubits,
              const std::vector< int > & controls = {})  [inline]
```

Constructs a Gate with a specified dimension.

Initializes a square matrix of complex numbers with the specified dimension and sets all elements to zero.

Parameters

dim	The dimension of the square matrix (number of rows and columns).
qubits	The target and control qubits.
controls	The control qubits, defaults to an empty vector.

5.3.3 Member Function Documentation

5.3.3.1 get_controls()

```
const std::vector< int > & Gate::get_controls () const  [inline]
```

Returns the control qubits for the gate.

Returns

The vector of control qubits.

5.3.3.2 get_matrix()

```
Eigen::MatrixXcd Gate::get_matrix (
              int n)  [inline]
```

Generates the unitary matrix for the gate, supporting multi-control gates.

Parameters

n	The total number of qubits in the system.

Returns

The resulting operator matrix as an Eigen::MatrixXcd.

5.3.3.3 get_qubits()

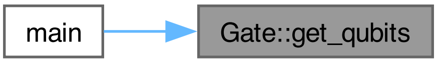
```
const std::vector< int > & Gate::get_qubits () const  [inline]
```

Returns the qubits the gate acts on.

Returns

The vector of target qubits.

Here is the caller graph for this function:



5.3.3.4 num_qubits()

```
int Gate::num_qubits () const  [inline]
```

Returns the number of qubits the gate acts on.

Returns

The number of target qubits.

5.3.3.5 to_string()

```
virtual std::string Gate::to_string () const  [pure virtual]
```

Implemented in CNOT, H, PauliX, PauliY, PauliZ, and RotationGate.

5.3.4 Member Data Documentation

5.3.4.1 controls

```
std::vector<int> Gate::controls  [protected]
```

5.3.4.2 qubits

```
std::vector<int> Gate::qubits  [protected]
```

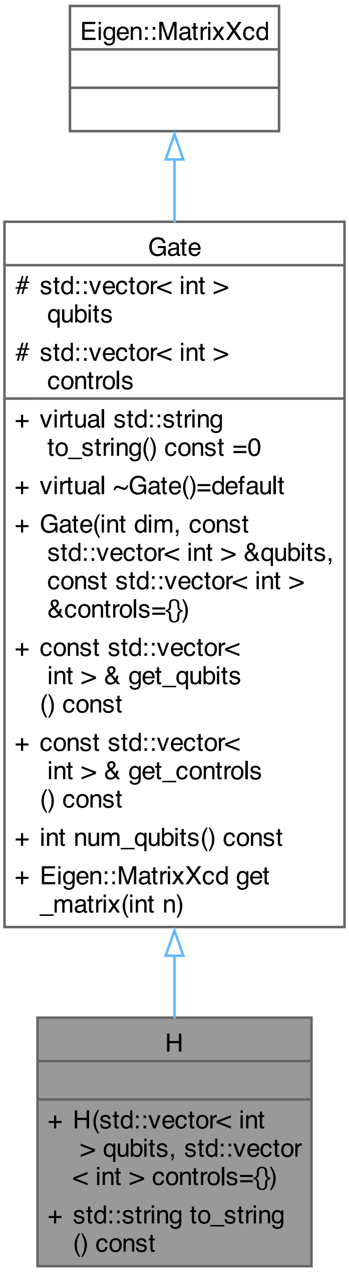
The documentation for this class was generated from the following file:

- src/gates/Gate.hpp

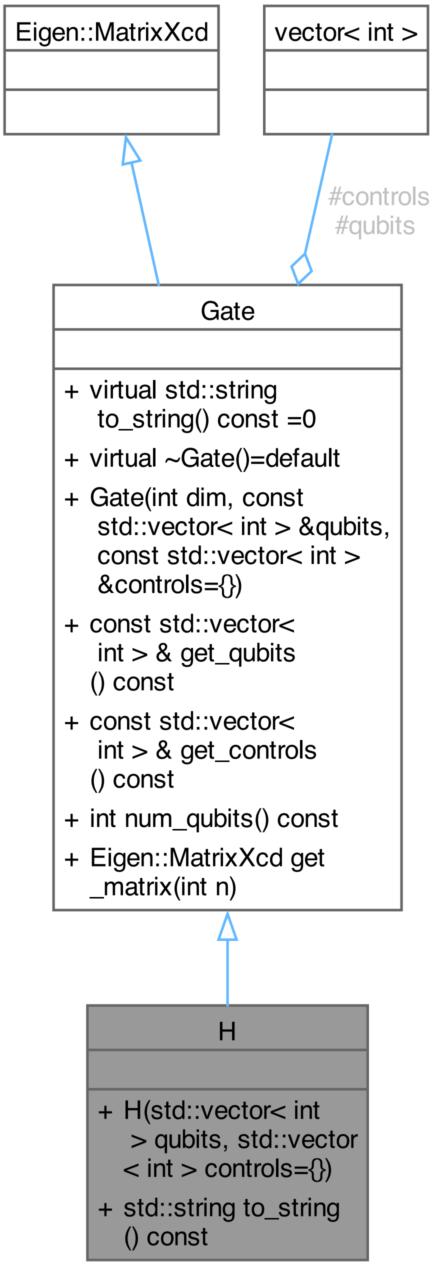
5.4 H Class Reference

```
#include <Hadamard.hpp>
```

Inheritance diagram for H:


Eigen::MatrixXcd
Gate
# std::vector< int > qubits
# std::vector< int > controls
+ virtual std::string to_string() const =0
+ virtual ~Gate()=default
+ Gate(int dim, const std::vector< int > &qubits, const std::vector< int > &controls={})
+ const std::vector< int > & get_qubits () const
+ const std::vector< int > & get_controls () const
+ int num_qubits() const
+ Eigen::MatrixXcd get _matrix(int n)
H
+ H(std::vector< int > qubits, std::vector < int > controls={})
+ std::string to_string () const

Collaboration diagram for H:


Eigen::MatrixXcd
vector< int >
#controls
#qubits
Gate
+ virtual std::string to_string() const =0
+ virtual ~Gate()=default
+ Gate(int dim, const std::vector< int > &qubits, const std::vector< int > &controls={})
+ const std::vector< int > & get_qubits () const
+ const std::vector< int > & get_controls () const
+ int num_qubits() const
+ Eigen::MatrixXcd get _matrix(int n)
H
+ H(std::vector< int > qubits, std::vector < int > controls={})
+ std::string to_string () const

Public Member Functions

- H (std::vector< int > qubits, std::vector< int > controls={})

  Constructs a Hadamard gate with specified target and control qubits.
- std::string to_string () const

Public Member Functions inherited from Gate

- virtual ~Gate ()=default

    Virtual destructor for the Gate class.
- Gate (int dim, const std::vector< int > &qubits, const std::vector< int > &controls={})

    Constructs a Gate with a specified dimension.
- const std::vector< int > & get_qubits () const

    Returns the qubits the gate acts on.
- const std::vector< int > & get_controls () const

    Returns the control qubits for the gate.
- int num_qubits () const

    Returns the number of qubits the gate acts on.
- Eigen::MatrixXcd get_matrix (int n)

    Generates the unitary matrix for the gate, supporting multi-control gates.

Additional Inherited Members

Protected Attributes inherited from Gate

- std::vector< int > qubits
- std::vector< int > controls

5.4.1 Constructor & Destructor Documentation

5.4.1.1 H()

```
H::H (
            std::vector< int > qubits,
            std::vector< int > controls = {})  [inline]
```

Constructs a Hadamard gate with specified target and control qubits.

Initializes a 2x2 matrix representing the Hadamard gate with the specified target and control qubits.

Parameters

qubits	The target and control qubits.
controls	The control qubits.

5.4.2 Member Function Documentation

5.4.2.1 to_string()

```
std::string H::to_string () const  [inline], [virtual]
```

Implements Gate.

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

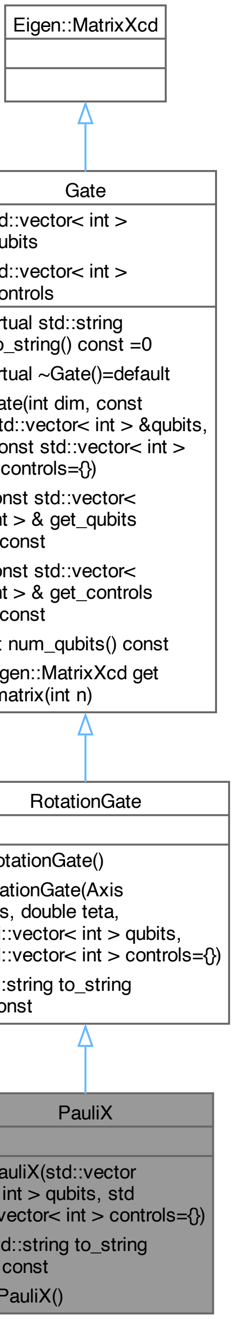
- src/gates/Hadamard.hpp

5.5 PauliX Class Reference

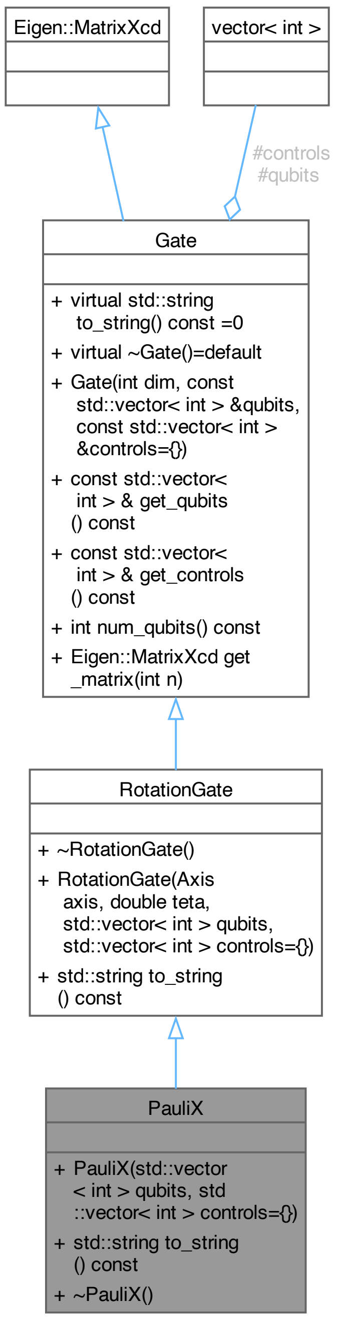
A class representing a quantum rotation gate around the X axis axis by an angle of pi inheriting from the RotationGate class.

```
#include <PauliGates.hpp>
```

Inheritance diagram for PauliX:


Eigen::MatrixXcd
Gate
# std::vector< int > qubits
# std::vector< int > controls
+ virtual std::string to_string() const =0
+ virtual ~Gate()=default
+ Gate(int dim, const std::vector< int > &qubits, const std::vector< int > &controls={})
+ const std::vector< int > & get_qubits () const
+ const std::vector< int > & get_controls () const
+ int num_qubits() const
+ Eigen::MatrixXcd get _matrix(int n)
RotationGate
+ ~RotationGate()
+ RotationGate(Axis axis, double teta, std::vector< int > qubits, std::vector< int > controls={})
+ std::string to_string () const
PauliX
+ PauliX(std::vector < int > qubits, std ::vector< int > controls={})
+ std::string to_string () const
+ ~PauliX()

Collaboration diagram for PauliX:


Eigen::MatrixXcd
vector< int >
#controls
#qubits
Gate
+ virtual std::string to_string() const =0
+ virtual ~Gate()=default
+ Gate(int dim, const std::vector< int > &qubits, const std::vector< int > &controls={})
+ const std::vector< int > & get_qubits () const
+ const std::vector< int > & get_controls () const
+ int num_qubits() const
+ Eigen::MatrixXcd get _matrix(int n)
RotationGate
+ ~RotationGate()
+ RotationGate(Axis axis, double teta, std::vector< int > qubits, std::vector< int > controls={})
+ std::string to_string () const
PauliX
+ PauliX(std::vector < int > qubits, std ::vector< int > controls={})
+ std::string to_string () const
+ ~PauliX()

Public Member Functions

- PauliX (std::vector< int > qubits, std::vector< int > controls={})

  Constructs a PauliX gate, by calling the parent constructor with the set parameters.
- std::string to_string () const

  Gives the string representation of the gate.
- ∼PauliX ()

  Destructor for the PauliX class.

Public Member Functions inherited from RotationGate

- ∼RotationGate ()

  Destructor for the RotationGate class.
- RotationGate (Axis axis, double teta, std::vector< int > qubits, std::vector< int > controls={})

  Constructs a RotationGate representing rotation around a given axis, with a given teta angle. Sets the elements of the complex 2×2 matrix.
- std::string to_string () const

  Gives the string representation of the gate.

Public Member Functions inherited from Gate

- virtual ∼Gate ()=default

  Virtual destructor for the Gate class.
- Gate (int dim, const std::vector< int > &qubits, const std::vector< int > &controls={})

  Constructs a Gate with a specified dimension.
- const std::vector< int > & get_qubits () const

  Returns the qubits the gate acts on.
- const std::vector< int > & get_controls () const

  Returns the control qubits for the gate.
- int num_qubits () const

  Returns the number of qubits the gate acts on.
- Eigen::MatrixXcd get_matrix (int n)

  Generates the unitary matrix for the gate, supporting multi-control gates.

Additional Inherited Members

Protected Attributes inherited from Gate

- std::vector< int > qubits
- std::vector< int > controls

5.5.1 Detailed Description

A class representing a quantum rotation gate around the X axis axis by an angle of pi inheriting from the RotationGate class.

Matrix form: (0,0) (1,0) (1,0) (0,0)

5.5.2 Constructor & Destructor Documentation

5.5.2.1 PauliX()

```
PauliX::PauliX (
             std::vector< int > qubits,
             std::vector< int > controls = {})  [inline]
```

Constructs a PauliX gate, by calling the parent constructor with the set parameters.

Parameters

qubits	The target and control qubits.
controls	The control qubits, defaults to an empty vector.

5.5.2.2 ∼PauliX()

```
PauliX::∼PauliX ()  [inline]
```

Destructor for the PauliX class.

5.5.3 Member Function Documentation

5.5.3.1 to_string()

```
std::string PauliX::to_string () const  [inline], [virtual]
```

Gives the string representation of the gate.

Returns

> The string representation of the gate, in format "X".

Implements Gate.

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

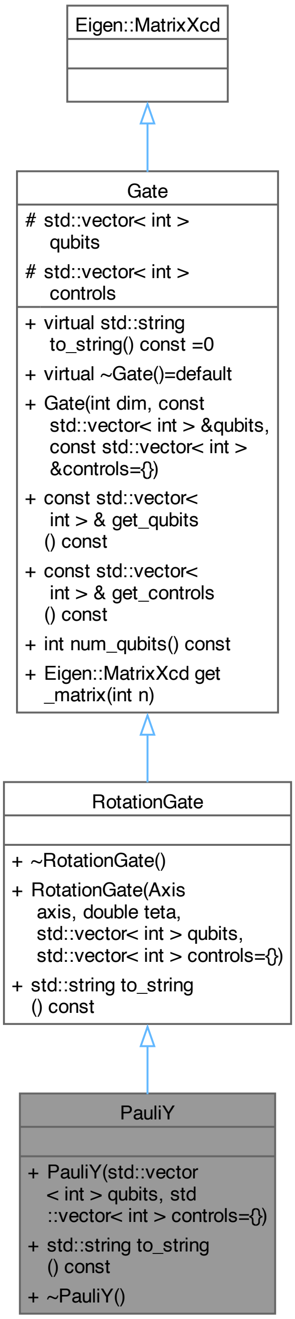
- src/gates/PauliGates.hpp

5.6 PauliY Class Reference

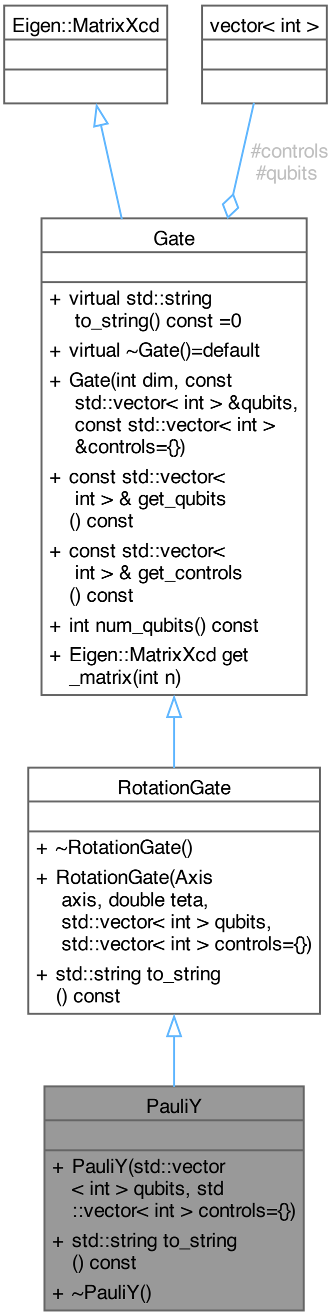
A class representing a quantum rotation gate around the Y axis axis by an angle of pi inheriting from the RotationGate class.

```
#include <PauliGates.hpp>
```

Inheritance diagram for PauliY:


Eigen::MatrixXcd
Gate
# std::vector< int > qubits
# std::vector< int > controls
+ virtual std::string to_string() const =0
+ virtual ~Gate()=default
+ Gate(int dim, const std::vector< int > &qubits, const std::vector< int > &controls={})
+ const std::vector< int > & get_qubits () const
+ const std::vector< int > & get_controls () const
+ int num_qubits() const
+ Eigen::MatrixXcd get _matrix(int n)
RotationGate
+ ~RotationGate()
+ RotationGate(Axis axis, double teta, std::vector< int > qubits, std::vector< int > controls={})
+ std::string to_string () const
PauliY
+ PauliY(std::vector < int > qubits, std ::vector< int > controls={})
+ std::string to_string () const
+ ~PauliY()

Collaboration diagram for PauliY:


Eigen::MatrixXcd
vector< int >
#controls
#qubits
Gate
+ virtual std::string to_string() const =0
+ virtual ~Gate()=default
+ Gate(int dim, const std::vector< int > &qubits, const std::vector< int > &controls={})
+ const std::vector< int > & get_qubits () const
+ const std::vector< int > & get_controls () const
+ int num_qubits() const
+ Eigen::MatrixXcd get _matrix(int n)
RotationGate
+ ~RotationGate()
+ RotationGate(Axis axis, double teta, std::vector< int > qubits, std::vector< int > controls={})
+ std::string to_string () const
PauliY
+ PauliY(std::vector < int > qubits, std ::vector< int > controls={})
+ std::string to_string () const
+ ~PauliY()

Public Member Functions

- PauliY (std::vector< int > qubits, std::vector< int > controls={})

  Constructs a PauliY gate, by calling the parent constructor with the set parameters.
- std::string to_string () const

  Gives the string representation of the gate.
- ~PauliY ()

  Destructor for the PauliY class.

Public Member Functions inherited from RotationGate

- ∼RotationGate ()

  Destructor for the RotationGate class.
- RotationGate (Axis axis, double teta, std::vector< int > qubits, std::vector< int > controls={})

  Constructs a RotationGate representing rotation around a given axis, with a given teta angle. Sets the elements of the complex 2×2 matrix.
- std::string to_string () const

  Gives the string representation of the gate.

Public Member Functions inherited from Gate

- virtual ∼Gate ()=default

  Virtual destructor for the Gate class.
- Gate (int dim, const std::vector< int > &qubits, const std::vector< int > &controls={})

  Constructs a Gate with a specified dimension.
- const std::vector< int > & get_qubits () const

  Returns the qubits the gate acts on.
- const std::vector< int > & get_controls () const

  Returns the control qubits for the gate.
- int num_qubits () const

  Returns the number of qubits the gate acts on.
- Eigen::MatrixXcd get_matrix (int n)

  Generates the unitary matrix for the gate, supporting multi-control gates.

Additional Inherited Members

Protected Attributes inherited from Gate

- std::vector< int > qubits
- std::vector< int > controls

5.6.1 Detailed Description

A class representing a quantum rotation gate around the Y axis axis by an angle of pi inheriting from the RotationGate class.

Matrix form: (0,0) (0,-1) (0,1) (0,0)

5.6.2 Constructor & Destructor Documentation

5.6.2.1 PauliY()

```
PauliY::PauliY (
            std::vector< int > qubits,
            std::vector< int > controls = {})  [inline]
```

Constructs a PauliY gate, by calling the parent constructor with the set parameters.

Parameters

qubits	The target and control qubits.
controls	The control qubits, defaults to an empty vector.

5.6.2.2 ∼PauliY()

```
PauliY::∼PauliY () [inline]
```

Destructor for the PauliY class.

5.6.3 Member Function Documentation

5.6.3.1 to_string()

```
std::string PauliY::to_string () const [inline], [virtual]
```

Gives the string representation of the gate.

Returns

> The string representation of the gate, in format "Y".

Implements Gate.

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

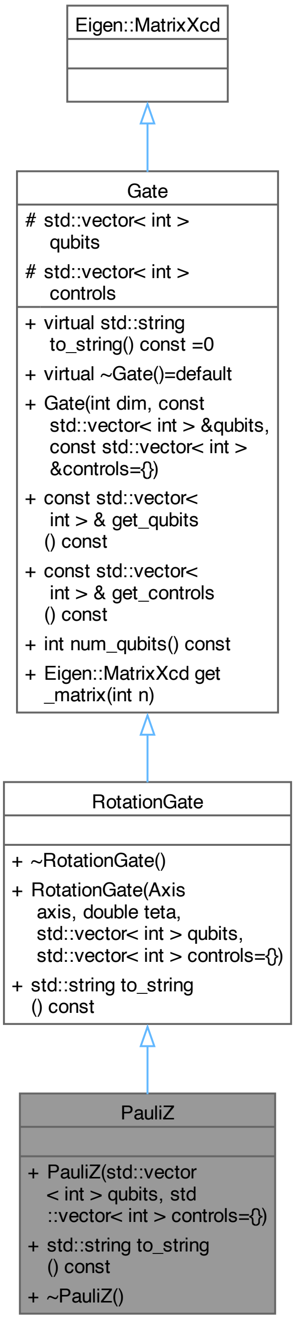
- src/gates/PauliGates.hpp

5.7 PauliZ Class Reference

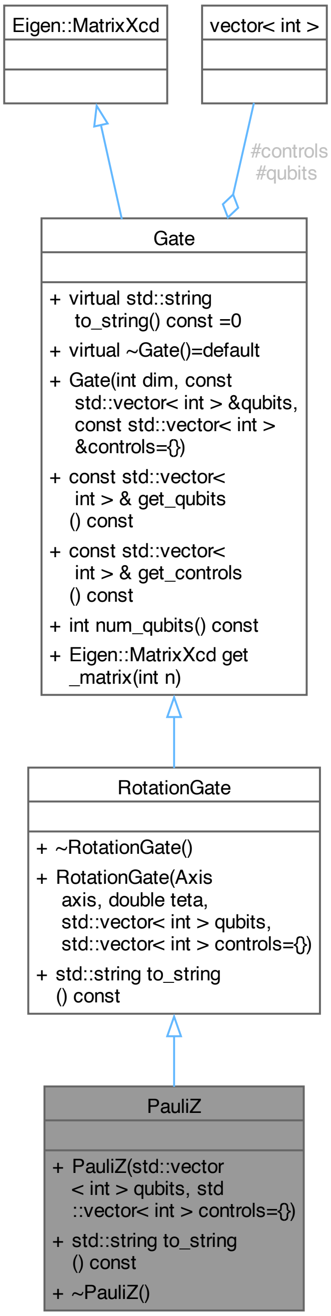
A class representing a quantum rotation gate around the Z axis axis by an angle of pi inheriting from the RotationGate class Matrix form: (1,0) (0,0) (0,0) (-1,-0)

```
#include <PauliGates.hpp>
```

Inheritance diagram for PauliZ:


Eigen::MatrixXcd
Gate
# std::vector< int > qubits
# std::vector< int > controls
+ virtual std::string to_string() const =0
+ virtual ~Gate()=default
+ Gate(int dim, const std::vector< int > &qubits, const std::vector< int > &controls={})
+ const std::vector< int > & get_qubits () const
+ const std::vector< int > & get_controls () const
+ int num_qubits() const
+ Eigen::MatrixXcd get _matrix(int n)
RotationGate
+ ~RotationGate()
+ RotationGate(Axis axis, double teta, std::vector< int > qubits, std::vector< int > controls={})
+ std::string to_string () const
PauliZ
+ PauliZ(std::vector < int > qubits, std ::vector< int > controls={})
+ std::string to_string () const
+ ~PauliZ()

Collaboration diagram for PauliZ:


Eigen::MatrixXcd
vector< int >
#controls
#qubits
Gate
+ virtual std::string to_string() const =0
+ virtual ~Gate()=default
+ Gate(int dim, const std::vector< int > &qubits, const std::vector< int > &controls={})
+ const std::vector< int > & get_qubits () const
+ const std::vector< int > & get_controls () const
+ int num_qubits() const
+ Eigen::MatrixXcd get _matrix(int n)
RotationGate
+ ~RotationGate()
+ RotationGate(Axis axis, double teta, std::vector< int > qubits, std::vector< int > controls={})
+ std::string to_string () const
PauliZ
+ PauliZ(std::vector < int > qubits, std ::vector< int > controls={})
+ std::string to_string () const
+ ~PauliZ()

Public Member Functions

- PauliZ (std::vector< int > qubits, std::vector< int > controls={})

  Constructs a PauliZ gate, by calling the parent constructor with the set parameters.
- std::string to_string () const

  Gives the string representation of the gate.
- ~PauliZ ()

  Destructor for the PauliZ class.

Public Member Functions inherited from RotationGate

- ∼RotationGate ()

  Destructor for the RotationGate class.
- RotationGate (Axis axis, double teta, std::vector< int > qubits, std::vector< int > controls={})

  Constructs a RotationGate representing rotation around a given axis, with a given teta angle. Sets the elements of the complex 2×2 matrix.
- std::string to_string () const

  Gives the string representation of the gate.

Public Member Functions inherited from Gate

- virtual ∼Gate ()=default

  Virtual destructor for the Gate class.
- Gate (int dim, const std::vector< int > &qubits, const std::vector< int > &controls={})

  Constructs a Gate with a specified dimension.
- const std::vector< int > & get_qubits () const

  Returns the qubits the gate acts on.
- const std::vector< int > & get_controls () const

  Returns the control qubits for the gate.
- int num_qubits () const

  Returns the number of qubits the gate acts on.
- Eigen::MatrixXcd get_matrix (int n)

  Generates the unitary matrix for the gate, supporting multi-control gates.

Additional Inherited Members

Protected Attributes inherited from Gate

- std::vector< int > qubits
- std::vector< int > controls

5.7.1 Detailed Description

A class representing a quantum rotation gate around the Z axis axis by an angle of pi inheriting from the RotationGate class Matrix form: (1,0) (0,0) (0,0) (-1,-0)

5.7.2 Constructor & Destructor Documentation

5.7.2.1 PauliZ()

```
PauliZ::PauliZ (
            std::vector< int > qubits,
            std::vector< int > controls = {})  [inline]
```

Constructs a PauliZ gate, by calling the parent constructor with the set parameters.

Parameters

qubits	The target and control qubits.
controls	The control qubits, defaults to an empty vector.

5.7.2.2 ∼PauliZ()

```
PauliZ::∼PauliZ ( ) [inline]
```

Destructor for the PauliZ class.

5.7.3 Member Function Documentation

5.7.3.1 to_string()

```
std::string PauliZ::to_string ( ) const [inline], [virtual]
```

Gives the string representation of the gate.

Returns

> The string representation of the gate, in format "Z".

Implements Gate.

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

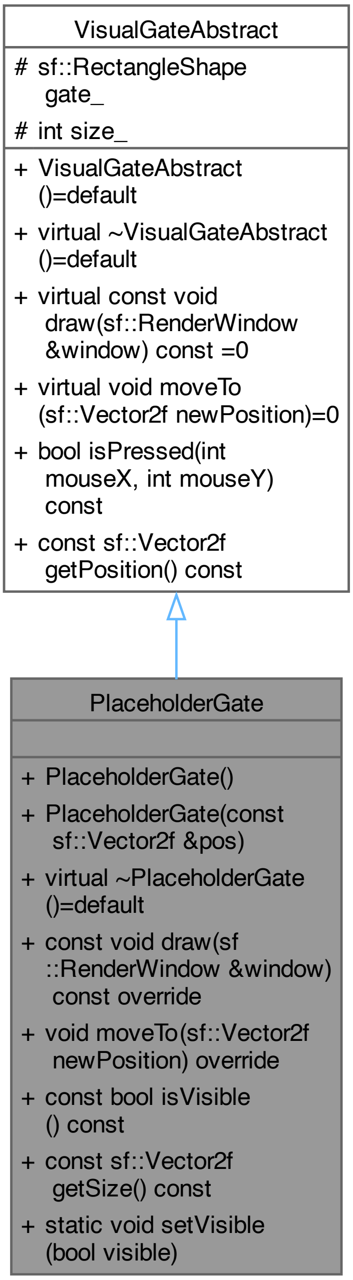
- src/gates/PauliGates.hpp

5.8 PlaceholderGate Class Reference

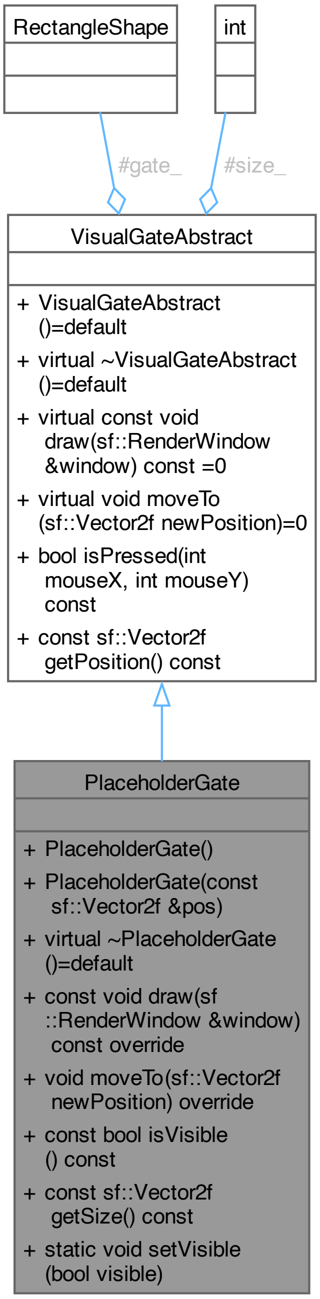
A class visualizing a placeholder gate in GUI.

```
#include <PlaceholderGate.hpp>
```

Inheritance diagram for PlaceholderGate:


VisualGateAbstract
# sf::RectangleShape gate_
# int size_
+ VisualGateAbstract ()=default
+ virtual ~VisualGateAbstract ()=default
+ virtual const void draw(sf::RenderWindow &window) const =0
+ virtual void moveTo (sf::Vector2f newPosition)=0
+ bool isPressed(int mouseX, int mouseY) const
+ const sf::Vector2f getPosition() const
PlaceholderGate
+ PlaceholderGate()
+ PlaceholderGate(const sf::Vector2f &pos)
+ virtual ~PlaceholderGate ()=default
+ const void draw(sf ::RenderWindow &window) const override
+ void moveTo(sf::Vector2f newPosition) override
+ const bool isVisible () const
+ const sf::Vector2f getSize() const
+ static void setVisible (bool visible)

Collaboration diagram for PlaceholderGate:


RectangleShape
int
#gate_
#size_
VisualGateAbstract
+ VisualGateAbstract ()=default
+ virtual ~VisualGateAbstract ()=default
+ virtual const void draw(sf::RenderWindow &window) const =0
+ virtual void moveTo (sf::Vector2f newPosition)=0
+ bool isPressed(int mouseX, int mouseY) const
+ const sf::Vector2f getPosition() const
PlaceholderGate
+ PlaceholderGate()
+ PlaceholderGate(const sf::Vector2f &pos)
+ virtual ~PlaceholderGate ()=default
+ const void draw(sf ::RenderWindow &window) const override
+ void moveTo(sf::Vector2f newPosition) override
+ const bool isVisible () const
+ const sf::Vector2f getSize() const
+ static void setVisible (bool visible)

Public Member Functions

- PlaceholderGate ()

  Default constructor for PlaceholderGate.
- PlaceholderGate (const sf::Vector2f &pos)

  Constructs a PlaceholderGate with a specified position and abbreviation of the gate.
- virtual ~PlaceholderGate ()=default

Default destructor for the PlaceholderGate class.
- const void draw (sf::RenderWindow &window) const override

  Draws the placeholder gate to the window if it is set to be visible.
- void moveTo (sf::Vector2f newPosition) override

  Moves the placeholder gate to a specified position.
- const bool isVisible () const

  Getter for 'visible_' variable.
- const sf::Vector2f getSize () const

  Get the size of a gate object.

Public Member Functions inherited from VisualGateAbstract

- VisualGateAbstract ()=default

  Default constructor for VisualGateAbstract class.
- virtual ∼VisualGateAbstract ()=default

  Default destructor for the VisualGate class.
- bool isPressed (int mouseX, int mouseY) const

  Checks if the mouse click happened inside the abstract gate.
- const sf::Vector2f getPosition () const

  Getter for the position of the gate.

Static Public Member Functions

- static void setVisible (bool visible)

  Static setter for 'visible_' variable.

Additional Inherited Members

Protected Attributes inherited from VisualGateAbstract

- sf::RectangleShape gate_
- int size_ = 90

5.8.1 Detailed Description

A class visualizing a placeholder gate in GUI.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 PlaceholderGate() [1/2]

```
PlaceholderGate::PlaceholderGate ()  [inline]
```

Default constructor for PlaceholderGate.

5.8.2.2 PlaceholderGate() [2/2]

```
PlaceholderGate::PlaceholderGate (
             const sf::Vector2f & pos)  [inline]
```

Constructs a PlaceholderGate with a specified position and abbreviation of the gate.

Initializes a square to specified position.

Parameters

pos	Position where the gate should be drawn in GUI.

5.8.2.3 ∼PlaceholderGate()

```
virtual PlaceholderGate::∼PlaceholderGate () [virtual], [default]
```

Default destructor for the PlaceholderGate class.

5.8.3 Member Function Documentation

5.8.3.1 draw()

```
const void PlaceholderGate::draw (
             sf::RenderWindow & window) const  [inline], [override], [virtual]
```

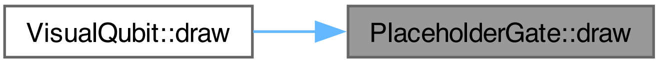
Draws the placeholder gate to the window if it is set to be visible.

Parameters

window	Window where the gate will be drawn.

Implements VisualGateAbstract.

Here is the caller graph for this function:



5.8.3.2 getSize()

```
const sf::Vector2f PlaceholderGate::getSize () const  [inline]
```

Get the size of a gate object.

Returns

const sf::Vector2f size of the gate

5.8.3.3 isVisible()

```
const bool PlaceholderGate::isVisible () const  [inline]
```

Getter for 'visible_' variable.

Returns

True or false, depending on the value of 'visible_'.

5.8.3.4 moveTo()

```
void PlaceholderGate::moveTo (
             sf::Vector2f newPosition)  [inline], [override], [virtual]
```

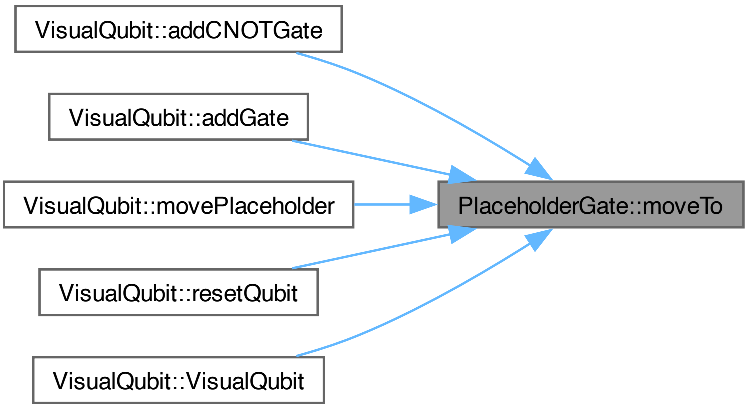
Moves the placeholder gate to a specified position.

Parameters

newPosition	Vector of the position the gate will be moved.

Implements VisualGateAbstract.

Here is the caller graph for this function:



5.8.3.5 setVisible()

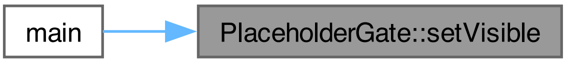
```
static void PlaceholderGate::setVisible (
             bool visible)  [inline], [static]
```

Static setter for 'visible_' variable.

Parameters

visible	the value to be set

Here is the caller graph for this function:



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

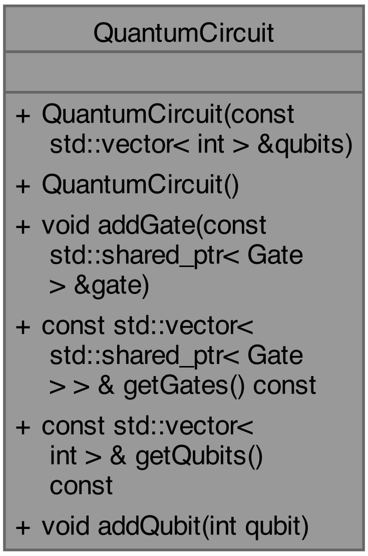
- src/view/PlaceholderGate.hpp

5.9 QuantumCircuit Class Reference

A class representing a quantum circuit with a set of quantum gates.

```
#include <QuantumCircuit.hpp>
```

Collaboration diagram for QuantumCircuit:


QuantumCircuit
+ QuantumCircuit(const std::vector< int > &qubits)
+ QuantumCircuit()
+ void addGate(const std::shared_ptr< Gate > &gate)
+ const std::vector< std::shared_ptr< Gate > > & getGates() const
+ const std::vector< int > & getQubits() const
+ void addQubit(int qubit)

Public Member Functions

- QuantumCircuit (const std::vector< int > &qubits)

  Constructs a QuantumCircuit with qubits with their initial states.
- QuantumCircuit ()

  Default constructor for a new Quantum Circuit object.
- void addGate (const std::shared_ptr< Gate > &gate)

  Adds a gate to the quantum circuit.
- const std::vector< std::shared_ptr< Gate > > & getGates () const

  Returns the list of gates in the quantum circuit.
- const std::vector< int > & getQubits () const

  Returns the vector of qubits in the quantum circuit.
- void addQubit (int qubit)

  Adds a new qubit to the circuit.

5.9.1 Detailed Description

A class representing a quantum circuit with a set of quantum gates.

The QuantumCircuit class holds a collection of gates that operate on qubits. A quantum circuit can be constructed by specifying the number of qubits with specific states and then adding gates to the circuit.

The quantum circuit supports operations like adding gates to the circuit and retrieving the list of gates in the circuit.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 QuantumCircuit() [1/2]

```
QuantumCircuit::QuantumCircuit (
             const std::vector< int > & qubits)  [inline]
```

Constructs a QuantumCircuit with qubits with their initial states.

Parameters

qubits	The vector of qubits in the circuit with their initial states.

5.9.2.2 QuantumCircuit() [2/2]

```
QuantumCircuit::QuantumCircuit ()  [inline]
```

Default constructor for a new Quantum Circuit object.

5.9.3 Member Function Documentation

5.9.3.1 addGate()

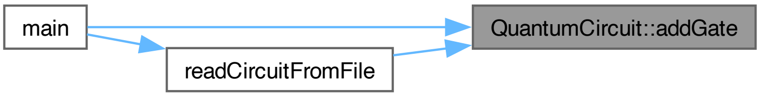
```
void QuantumCircuit::addGate (
             const std::shared_ptr< Gate > & gate)  [inline]
```

Adds a gate to the quantum circuit.

Parameters

gate	A shared pointer to the gate to be added to the circuit.

Here is the caller graph for this function:



5.9.3.2 addQubit()

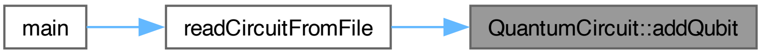
```
void QuantumCircuit::addQubit (
             int qubit)  [inline]
```

Adds a new qubit to the circuit.

Parameters

qubit	The qubit state to be added.

Here is the caller graph for this function:



5.9.3.3 getGates()

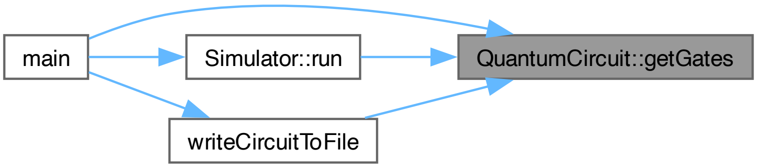
```
const std::vector< std::shared_ptr< Gate > > & QuantumCircuit::getGates () const  [inline]
```

Returns the list of gates in the quantum circuit.

Returns

A const reference to the vector of gates.

Here is the caller graph for this function:



5.9.3.4 getQubits()

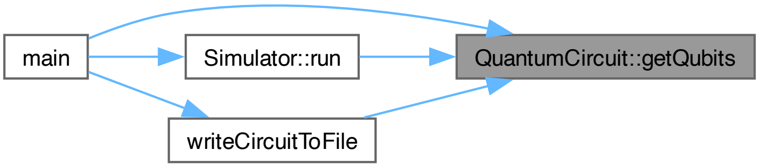
```
const std::vector< int > & QuantumCircuit::getQubits () const  [inline]
```

Returns the vector of qubits in the quantum circuit.

Returns

The vector of qubits.

Here is the caller graph for this function:



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

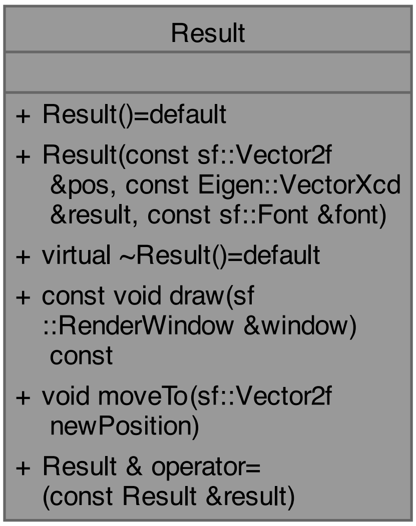
- src/controller/QuantumCircuit.hpp

5.10 Result Class Reference

A class visualizing the result of the quantum computer simulation.

```
#include <Result.hpp>
```

Collaboration diagram for Result:


Result
+ Result()=default
+ Result(const sf::Vector2f &pos, const Eigen::VectorXcd &result, const sf::Font &font)
+ virtual ~Result()=default
+ const void draw(sf ::RenderWindow &window) const
+ void moveTo(sf::Vector2f newPosition)
+ Result & operator= (const Result &result)

Public Member Functions

- Result ()=default

    Default constructor for Result.
- Result (const sf::Vector2f &pos, const Eigen::VectorXcd &result, const sf::Font &font)

    Constructs a Result object with a specified position and computation result.
- virtual ∼Result ()=default

    Default destructor for the Result class.
- const void draw (sf::RenderWindow &window) const

    Draws the result texts to the window.
- void moveTo (sf::Vector2f newPosition)

    Moves the result to a specified position.
- Result & operator= (const Result &result)

    Overloaded = operator.

5.10.1 Detailed Description

A class visualizing the result of the quantum computer simulation.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 Result() [1/2]

```
Result::Result ()  [default]
```

Default constructor for Result.

5.10.2.2 Result() [2/2]

```
Result::Result (
             const sf::Vector2f & pos,
             const Eigen::VectorXcd & result,
             const sf::Font & font)  [inline]
```

Constructs a Result object with a specified position and computation result.

Initializes two text blocks with "Result:" and the result provided by the quantum computer simulation.

Parameters

pos	Position where the "Result:" text should be drawn in GUI.
result	Result that will be visible next to text.
font	The font that will be used for the texts.

5.10.2.3 ∼Result()

```
virtual Result::∼Result ()  [virtual], [default]
```

Default destructor for the Result class.

5.10.3 Member Function Documentation

5.10.3.1 draw()

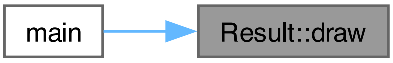
```
const void Result::draw (
             sf::RenderWindow & window) const  [inline]
```

Draws the result texts to the window.

Parameters

window	Window where the texts will be drawn.

Here is the caller graph for this function:



5.10.3.2 moveTo()

```
void Result::moveTo (
             sf::Vector2f newPosition)  [inline]
```

Moves the result to a specified position.

Parameters

newPosition	Vector of the position the result will be moved.

5.10.3.3 operator=()

```
Result & Result::operator= (
             const Result & result)  [inline]
```

Overloaded = operator.

Parameters

result	Result instance to be copied

Returns

Result& copied Result object

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

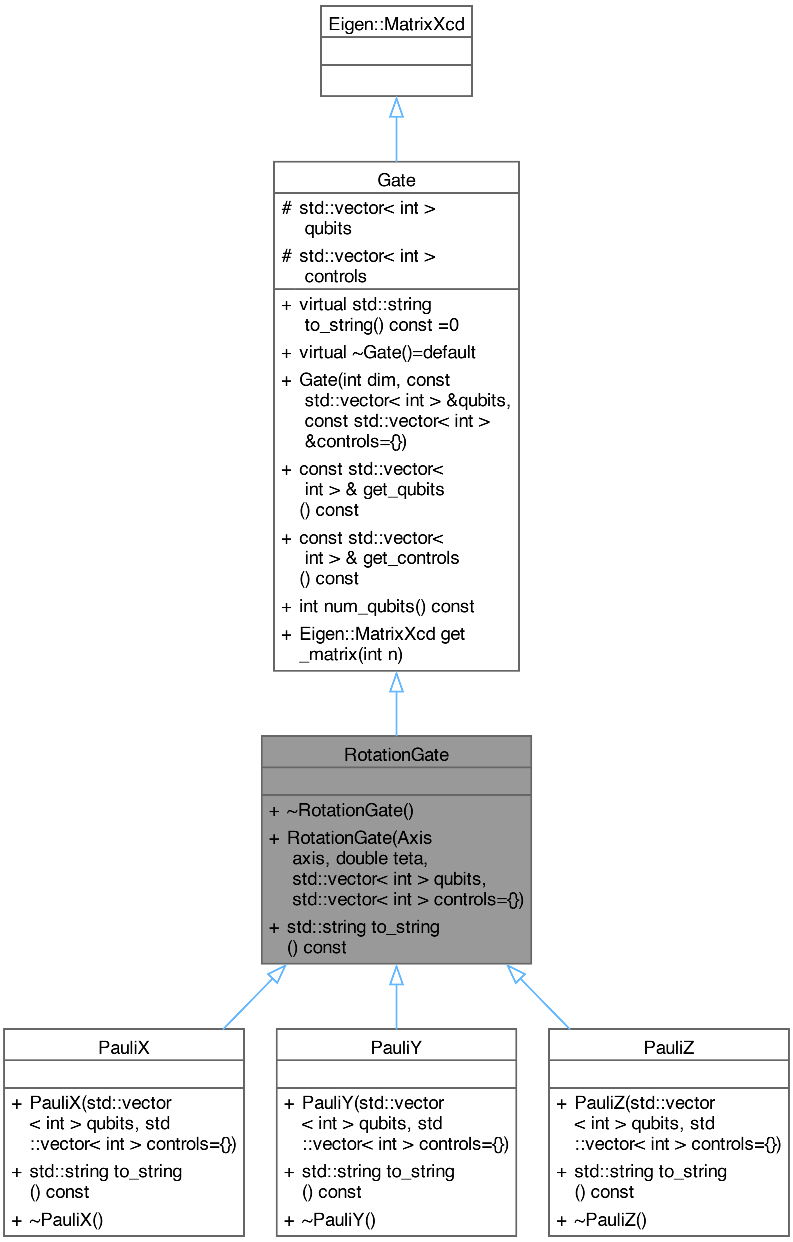
- src/view/Result.hpp

5.11 RotationGate Class Reference

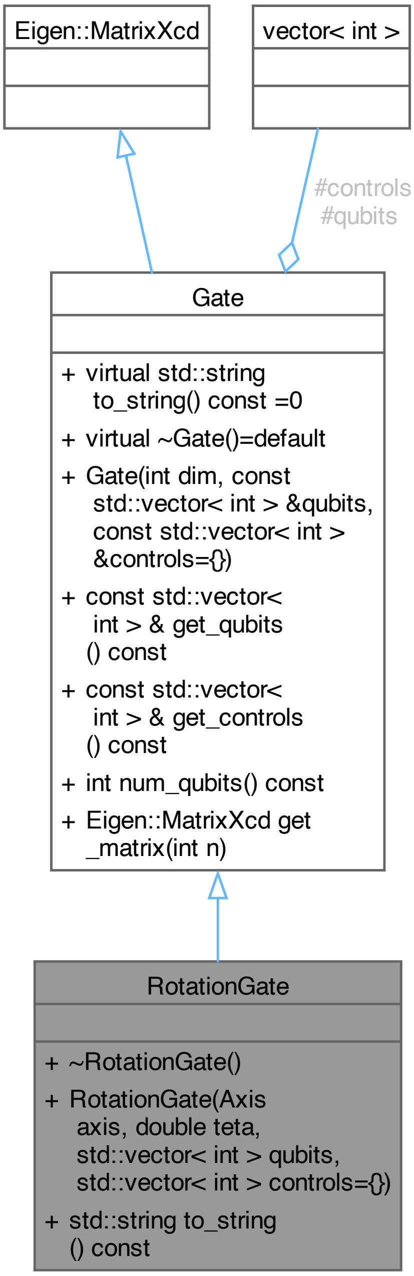
A class representing a quantum rotation gate around a set axis by an arbitrary angle, inheriting from the Gate class.

```
#include <RotationGate.hpp>
```

Inheritance diagram for RotationGate:


Eigen::MatrixXcd
Gate
# std::vector< int > qubits
# std::vector< int > controls
+ virtual std::string to_string() const =0
+ virtual ~Gate()=default
+ Gate(int dim, const std::vector< int > &qubits, const std::vector< int > &controls={})
+ const std::vector< int > & get_qubits () const
+ const std::vector< int > & get_controls () const
+ int num_qubits() const
+ Eigen::MatrixXcd get _matrix(int n)
RotationGate
+ ~RotationGate()
+ RotationGate(Axis axis, double teta, std::vector< int > qubits, std::vector< int > controls={})
+ std::string to_string () const
PauliX
+ PauliX(std::vector < int > qubits, std ::vector< int > controls={})
+ std::string to_string () const
+ ~PauliX()
PauliY
+ PauliY(std::vector < int > qubits, std ::vector< int > controls={})
+ std::string to_string () const
+ ~PauliY()
PauliZ
+ PauliZ(std::vector < int > qubits, std ::vector< int > controls={})
+ std::string to_string () const
+ ~PauliZ()

Collaboration diagram for RotationGate:


Eigen::MatrixXcd
vector< int >
#controls
#qubits
Gate
+ virtual std::string to_string() const =0
+ virtual ~Gate()=default
+ Gate(int dim, const std::vector< int > &qubits, const std::vector< int > &controls={})
+ const std::vector< int > & get_qubits () const
+ const std::vector< int > & get_controls () const
+ int num_qubits() const
+ Eigen::MatrixXcd get _matrix(int n)
RotationGate
+ ~RotationGate()
+ RotationGate(Axis axis, double teta, std::vector< int > qubits, std::vector< int > controls={})
+ std::string to_string () const

Public Member Functions

- ∼RotationGate ()

    Destructor for the RotationGate class.
- RotationGate (Axis axis, double teta, std::vector< int > qubits, std::vector< int > controls={})

    Constructs a RotationGate representing rotation around a given axis, with a given teta angle. Sets the elements of the complex 2×2 matrix.

- std::string to_string () const

  Gives the string representation of the gate.

Public Member Functions inherited from Gate

- virtual ∼Gate ()=default

  Virtual destructor for the Gate class.
- Gate (int dim, const std::vector< int > &qubits, const std::vector< int > &controls={})

  Constructs a Gate with a specified dimension.
- const std::vector< int > & get_qubits () const

  Returns the qubits the gate acts on.
- const std::vector< int > & get_controls () const

  Returns the control qubits for the gate.
- int num_qubits () const

  Returns the number of qubits the gate acts on.
- Eigen::MatrixXcd get_matrix (int n)

  Generates the unitary matrix for the gate, supporting multi-control gates.

Additional Inherited Members

Protected Attributes inherited from Gate

- std::vector< int > qubits
- std::vector< int > controls

5.11.1 Detailed Description

A class representing a quantum rotation gate around a set axis by an arbitrary angle, inheriting from the Gate class.

The RotationGate class is a gate with dimensions of 2×2 This class initializes the matrix representation of the rotationgate based on its axis and angle teta

5.11.2 Constructor & Destructor Documentation

5.11.2.1 ∼RotationGate()

```
RotationGate::∼RotationGate ()  [inline]
```

Destructor for the RotationGate class.

5.11.2.2 RotationGate()

```
RotationGate::RotationGate (
            Axis axis,
            double teta,
            std::vector< int > qubits,
            std::vector< int > controls = {})  [inline]
```

Constructs a RotationGate representing rotation around a given axis, with a given teta angle. Sets the elements of the complex 2×2 matrix.

Parameters

axis	The axis of the rotation operation. One of x, y, z.
teta	The angle of the rotation in radians (has a period of 4∗pi)
qubits	The target and control qubits.
controls	The control qubits, defaults to an empty vector.

5.11.3 Member Function Documentation

5.11.3.1 to_string()

```
std::string RotationGate::to_string () const  [inline], [virtual]
```

Gives the string representation of the gate.

Returns

The string representation of the gate, in format "R"+"axis"+"(angle)".

Implements Gate.

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

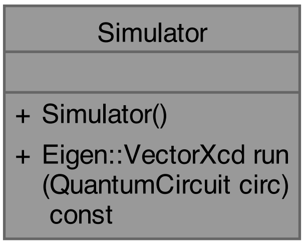
- src/gates/RotationGate.hpp

5.12 Simulator Class Reference

Representing a simulator object, would be a quantum computer in the real world.

```
#include <Simulator.hpp>
```

Collaboration diagram for Simulator:



Public Member Functions

- Simulator ()

  Constructs a Simulator object.
- Eigen::VectorXcd run (QuantumCircuit circ) const

  Evaluates the quantum circuit and returns the final state in little endian ordering meaning the least significant bit is the first element eg. H|00> = |00> + |01> / sqrt(2)

5.12.1 Detailed Description

Representing a simulator object, would be a quantum computer in the real world.

A simulator object can be constructed with a quantum circuit as input, and is able to evaluate the circuit with the given parameters such as noise models etc.

5.12.2 Constructor & Destructor Documentation

5.12.2.1 Simulator()

```
Simulator::Simulator ()  [inline]
```

Constructs a Simulator object.

5.12.3 Member Function Documentation

5.12.3.1 run()

```
Eigen::VectorXcd Simulator::run (
            QuantumCircuit circ) const  [inline]
```

Evaluates the quantum circuit and returns the final state in little endian ordering meaning the least significant bit is the first element eg. H|00> = |00> + |01> / sqrt(2)

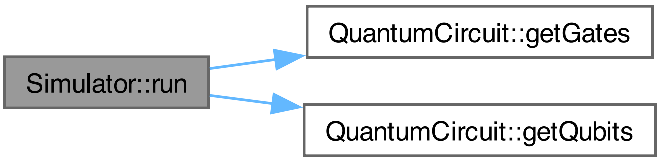
Parameters

circ	The quantum circuit to be simulated.

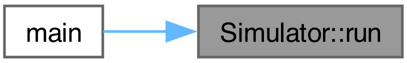
Returns

The final state of the quantum circuit in little endian ordering.

Here is the call graph for this function:



Here is the caller graph for this function:



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

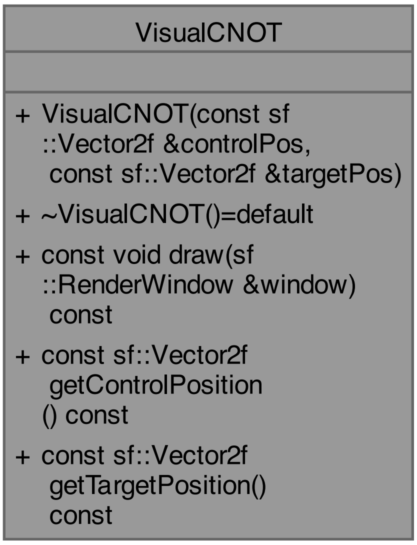
- src/controller/Simulator.hpp

5.13 VisualCNOT Class Reference

A class visualizing a CNOT gate in GUI.

```
#include <VisualCNOT.hpp>
```

Collaboration diagram for VisualCNOT:


VisualCNOT
+ VisualCNOT(const sf ::Vector2f &controlPos, const sf::Vector2f &targetPos)
+ ~VisualCNOT()=default
+ const void draw(sf ::RenderWindow &window) const
+ const sf::Vector2f getControlPosition () const
+ const sf::Vector2f getTargetPosition() const

Public Member Functions

- VisualCNOT (const sf::Vector2f &controlPos, const sf::Vector2f &targetPos)

  Constructs a VisualCNOT with a specified control and target qubit positions.
- ∼VisualCNOT ()=default

  Default destructor for the VisualCNOT class.
- const void draw (sf::RenderWindow &window) const

  Draws the gate to the window.
- const sf::Vector2f getControlPosition () const

  Get the position of the control dot.
- const sf::Vector2f getTargetPosition () const

  Get the position of the target dot.

5.13.1 Detailed Description

A class visualizing a CNOT gate in GUI.

5.13.2 Constructor & Destructor Documentation

5.13.2.1 VisualCNOT()

```
VisualCNOT::VisualCNOT (
            const sf::Vector2f & controlPos,
            const sf::Vector2f & targetPos)  [inline]
```

Constructs a VisualCNOT with a specified control and target qubit positions.

Parameters

controlPos	Position where the control end of the gate should be.
targetPos	Position where the target end of the gate should be.

5.13.2.2 ∼VisualCNOT()

```
VisualCNOT::∼VisualCNOT ()  [default]
```

Default destructor for the VisualCNOT class.

5.13.3 Member Function Documentation

5.13.3.1 draw()

```
const void VisualCNOT::draw (
             sf::RenderWindow & window) const  [inline]
```

Draws the gate to the window.

Parameters

window	Window where the gate will be drawn.

5.13.3.2 getControlPosition()

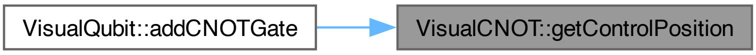
```
const sf::Vector2f VisualCNOT::getControlPosition () const  [inline]
```

Get the position of the control dot.

Returns

2D vector of the position.

Here is the caller graph for this function:



5.13.3.3 getTargetPosition()

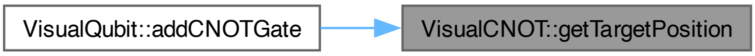
```
const sf::Vector2f VisualCNOT::getTargetPosition () const  [inline]
```

Get the position of the target dot.

Returns

2D vector of the position.

Here is the caller graph for this function:



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

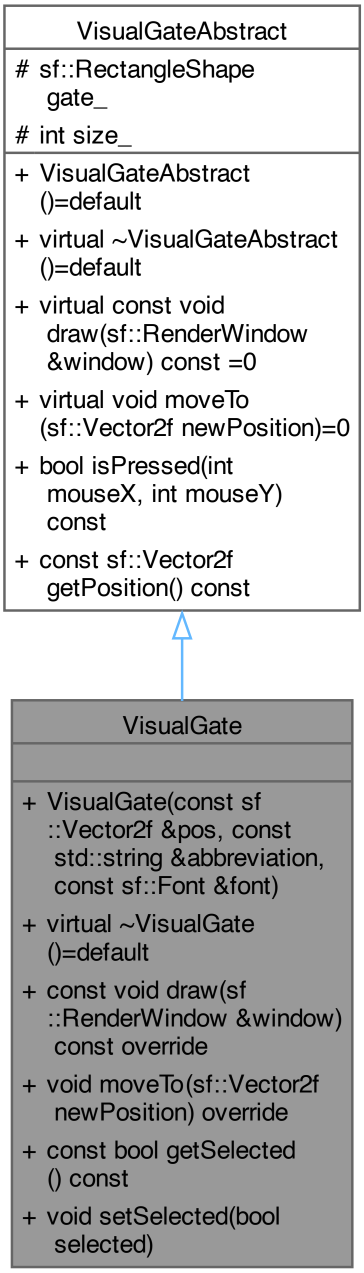
- src/view/VisualCNOT.hpp

5.14 VisualGate Class Reference

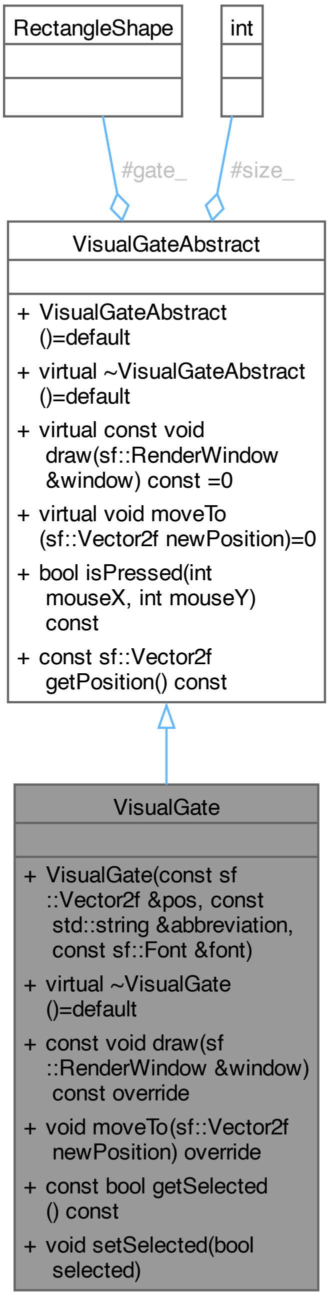
A class visualizing a quantum gate in GUI.

```
#include <VisualGate.hpp>
```

Inheritance diagram for VisualGate:


VisualGateAbstract
# sf::RectangleShape gate_
# int size_
+ VisualGateAbstract ()=default
+ virtual ~VisualGateAbstract ()=default
+ virtual const void draw(sf::RenderWindow &window) const =0
+ virtual void moveTo (sf::Vector2f newPosition)=0
+ bool isPressed(int mouseX, int mouseY) const
+ const sf::Vector2f getPosition() const
VisualGate
+ VisualGate(const sf ::Vector2f &pos, const std::string &abbreviation, const sf::Font &font)
+ virtual ~VisualGate ()=default
+ const void draw(sf ::RenderWindow &window) const override
+ void moveTo(sf::Vector2f newPosition) override
+ const bool getSelected () const
+ void setSelected(bool selected)

Collaboration diagram for VisualGate:


RectangleShape
int
#gate_
#size_
VisualGateAbstract
+ VisualGateAbstract ()=default
+ virtual ~VisualGateAbstract ()=default
+ virtual const void draw(sf::RenderWindow &window) const =0
+ virtual void moveTo (sf::Vector2f newPosition)=0
+ bool isPressed(int mouseX, int mouseY) const
+ const sf::Vector2f getPosition() const
VisualGate
+ VisualGate(const sf ::Vector2f &pos, const std::string &abbreviation, const sf::Font &font)
+ virtual ~VisualGate ()=default
+ const void draw(sf ::RenderWindow &window) const override
+ void moveTo(sf::Vector2f newPosition) override
+ const bool getSelected () const
+ void setSelected(bool selected)

Public Member Functions

- VisualGate (const sf::Vector2f &pos, const std::string &abbreviation, const sf::Font &font)

  Constructs a VisualGate with a specified position and abbreviation of the gate.
- virtual ~VisualGate ()=default

  Default destructor for the VisualGate class.
- const void draw (sf::RenderWindow &window) const override

Draws the gate to the window.

- void moveTo (sf::Vector2f newPosition) override

  Moves the gate to a specified position.
- const bool getSelected () const

  Getter for 'selected_' variable.
- void setSelected (bool selected)

  Setter for 'selected_' variable.

Public Member Functions inherited from VisualGateAbstract

- VisualGateAbstract ()=default

  Default constructor for VisualGateAbstract class.
- virtual ~VisualGateAbstract ()=default

  Default destructor for the VisualGate class.
- bool isPressed (int mouseX, int mouseY) const

  Checks if the mouse click happened inside the abstract gate.
- const sf::Vector2f getPosition () const

  Getter for the position of the gate.

Additional Inherited Members

Protected Attributes inherited from VisualGateAbstract

- sf::RectangleShape gate_
- int size_ = 90

5.14.1 Detailed Description

A class visualizing a quantum gate in GUI.

5.14.2 Constructor & Destructor Documentation

5.14.2.1 VisualGate()

```
VisualGate::VisualGate (
            const sf::Vector2f & pos,
            const std::string & abbreviation,
            const sf::Font & font)  [inline]
```

Constructs a VisualGate with a specified position and abbreviation of the gate.

Initializes a square to specified position with specified abbreviation inside of the square.

Parameters

pos	Position where the gate should be drawn in GUI.
abbreviation	Text that will be visible inside the gate square.
font	The font that will be used for the button texts.

5.14.2.2 ∼VisualGate()

```
virtual VisualGate::∼VisualGate ()  [virtual], [default]
```

Default destructor for the VisualGate class.

5.14.3 Member Function Documentation

5.14.3.1 draw()

```
const void VisualGate::draw (
             sf::RenderWindow & window) const  [inline], [override], [virtual]
```

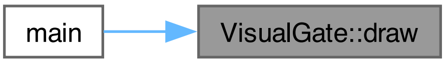
Draws the gate to the window.

Parameters

window	Window where the gate will be drawn.

Implements VisualGateAbstract.

Here is the caller graph for this function:



5.14.3.2 getSelected()

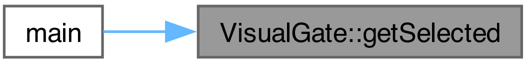
```
const bool VisualGate::getSelected () const  [inline]
```

Getter for 'selected_' variable.

Returns

True or false, depending on the value of 'selected_'.

Here is the caller graph for this function:



5.14.3.3 moveTo()

```
void VisualGate::moveTo (
            sf::Vector2f newPosition)  [inline], [override], [virtual]
```

Moves the gate to a specified position.

Parameters

newPosition	Vector of the position the gate will be moved.

Implements VisualGateAbstract.

5.14.3.4 setSelected()

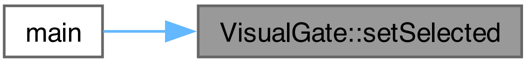
```
void VisualGate::setSelected (
            bool selected)  [inline]
```

Setter for 'selected_' variable.

Parameters

selected	The value to be set.

Here is the caller graph for this function:



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

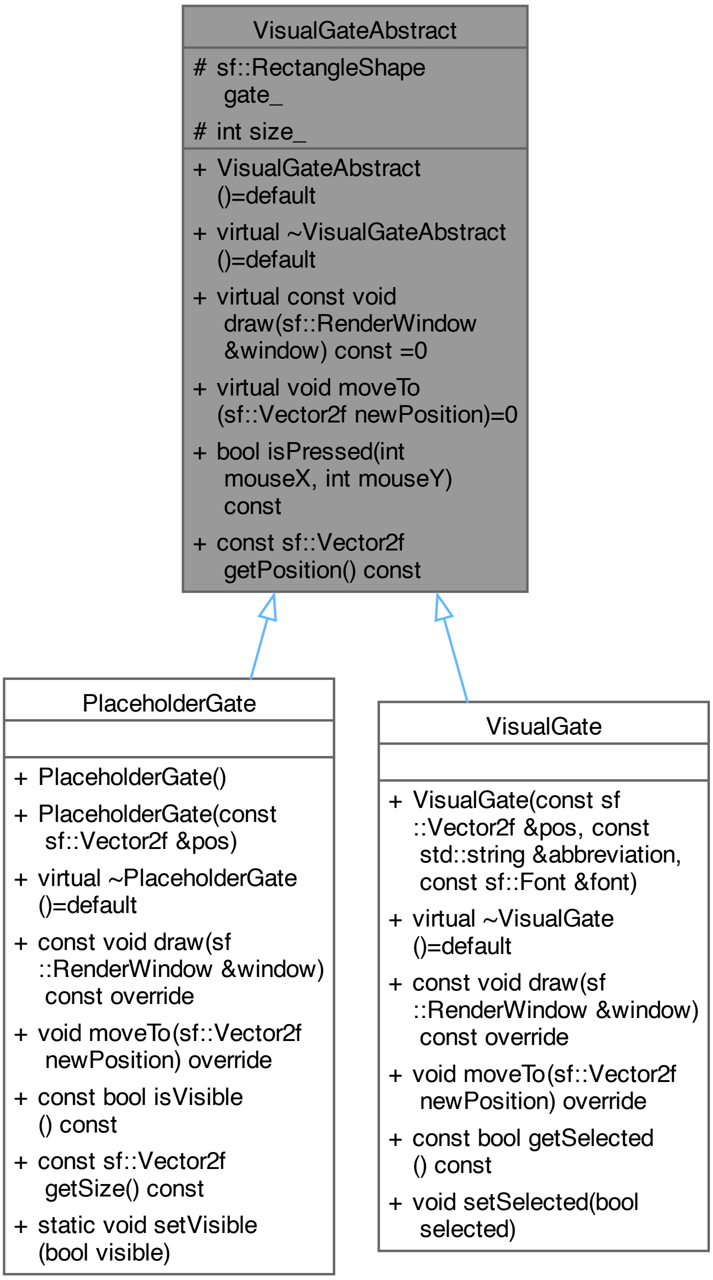
- src/view/VisualGate.hpp

5.15 VisualGateAbstract Class Reference

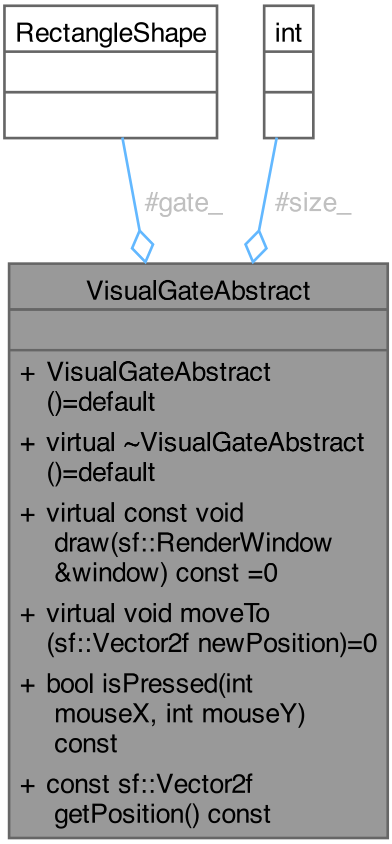
An abstract class for visual gate-like classes.

```
#include <VisualGateAbstract.hpp>
```

Inheritance diagram for VisualGateAbstract:


VisualGateAbstract
# sf::RectangleShape gate_
# int size_
+ VisualGateAbstract ()=default
+ virtual ~VisualGateAbstract ()=default
+ virtual const void draw(sf::RenderWindow &window) const =0
+ virtual void moveTo (sf::Vector2f newPosition)=0
+ bool isPressed(int mouseX, int mouseY) const
+ const sf::Vector2f getPosition() const
PlaceholderGate
+ PlaceholderGate()
+ PlaceholderGate(const sf::Vector2f &pos)
+ virtual ~PlaceholderGate ()=default
+ const void draw(sf ::RenderWindow &window) const override
+ void moveTo(sf::Vector2f newPosition) override
+ const bool isVisible () const
+ const sf::Vector2f getSize() const
+ static void setVisible (bool visible)
VisualGate
+ VisualGate(const sf ::Vector2f &pos, const std::string &abbreviation, const sf::Font &font)
+ virtual ~VisualGate ()=default
+ const void draw(sf ::RenderWindow &window) const override
+ void moveTo(sf::Vector2f newPosition) override
+ const bool getSelected () const
+ void setSelected(bool selected)

Collaboration diagram for VisualGateAbstract:


RectangleShape
int
#gate_
#size_
VisualGateAbstract
+ VisualGateAbstract ()=default
+ virtual ~VisualGateAbstract ()=default
+ virtual const void draw(sf::RenderWindow &window) const =0
+ virtual void moveTo (sf::Vector2f newPosition)=0
+ bool isPressed(int mouseX, int mouseY) const
+ const sf::Vector2f getPosition() const

Public Member Functions

- VisualGateAbstract ()=default

  Default constructor for VisualGateAbstract class.
- virtual ∼VisualGateAbstract ()=default

  Default destructor for the VisualGate class.
- virtual const void draw (sf::RenderWindow &window) const =0

  Pure virtual function for drawing the VisualGateAbstract to the screen.
- virtual void moveTo (sf::Vector2f newPosition)=0

  Pure virtual function for moving the gate to a specified location.
- bool isPressed (int mouseX, int mouseY) const

  Checks if the mouse click happened inside the abstract gate.
- const sf::Vector2f getPosition () const

  Getter for the position of the gate.

Protected Attributes

- sf::RectangleShape gate_
- int size_ = 90

5.15.1 Detailed Description

An abstract class for visual gate-like classes.

5.15.2 Constructor & Destructor Documentation

5.15.2.1 VisualGateAbstract()

```
VisualGateAbstract::VisualGateAbstract ()  [default]
```

Default constructor for VisualGateAbstract class.

5.15.2.2 ∼VisualGateAbstract()

```
virtual VisualGateAbstract::~VisualGateAbstract ()  [virtual], [default]
```

Default destructor for the VisualGate class.

5.15.3 Member Function Documentation

5.15.3.1 draw()

```
virtual const void VisualGateAbstract::draw (
             sf::RenderWindow & window) const  [pure virtual]
```

Pure virtual function for drawing the VisualGateAbstract to the screen.

Parameters

window	Window where the VisualGateAbstract will be drawn.

Implemented in PlaceholderGate, and VisualGate.

5.15.3.2 getPosition()

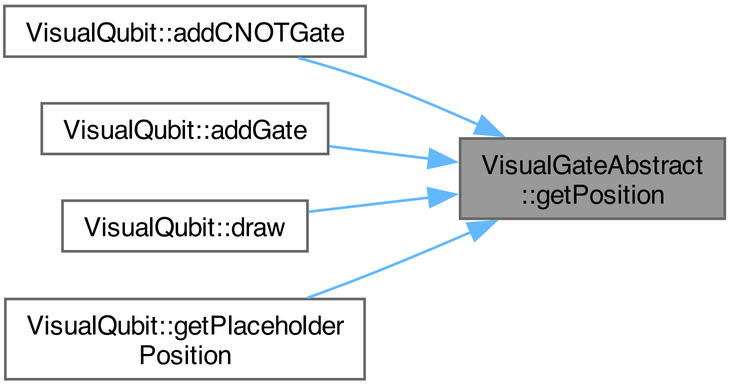
```
const sf::Vector2f VisualGateAbstract::getPosition () const  [inline]
```

Getter for the position of the gate.

Returns

2D vector of the position.

Here is the caller graph for this function:


VisualQubit::addCNOTGate
VisualQubit::addGate
VisualQubit::draw
VisualQubit::getPlaceholder Position
VisualGateAbstract ::getPosition

5.15.3.3 isPressed()

```
bool VisualGateAbstract::isPressed (
             int mouseX,
             int mouseY) const  [inline]
```

Checks if the mouse click happened inside the abstract gate.

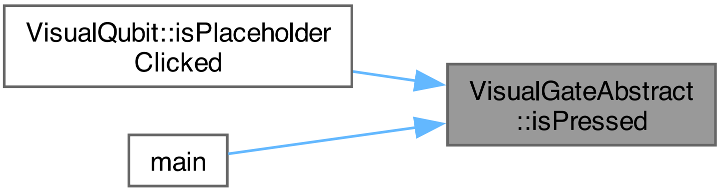
Parameters

mouseX	Mouse position on x axis.
mouseY	Mouse position on y axis.

Returns

True if the click was inside the gate, otherwise false.

Here is the caller graph for this function:


VisualQubit::isPlaceholder Clicked
main
VisualGateAbstract ::isPressed

5.15.3.4 moveTo()

```
virtual void VisualGateAbstract::moveTo (
             sf::Vector2f newPosition)  [pure virtual]
```

Pure virtual function for moving the gate to a specified location.

Parameters

newPosition	Vector of the position the gate will be moved.

Implemented in PlaceholderGate, and VisualGate.

5.15.4 Member Data Documentation

5.15.4.1 gate_

```
sf::RectangleShape VisualGateAbstract::gate_  [protected]
```

5.15.4.2 size_

```
int VisualGateAbstract::size_ = 90  [protected]
```

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

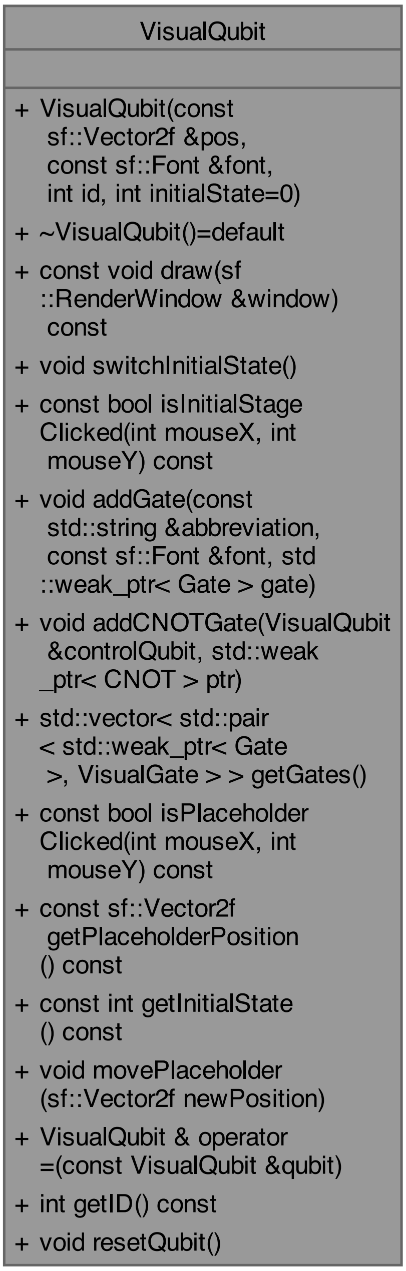
- src/view/VisualGateAbstract.hpp

5.16 VisualQubit Class Reference

A class visualizing a qubit in GUI.

```
#include <VisualQubit.hpp>
```

Collaboration diagram for VisualQubit:


VisualQubit
+ VisualQubit(const sf::Vector2f &pos, const sf::Font &font, int id, int initialState=0)
+ ~VisualQubit()=default
+ const void draw(sf ::RenderWindow &window) const
+ void switchInitialState()
+ const bool isInitialStage Clicked(int mouseX, int mouseY) const
+ void addGate(const std::string &abbreviation, const sf::Font &font, std ::weak_ptr< Gate > gate)
+ void addCNOTGate(VisualQubit &controlQubit, std::weak _ptr< CNOT > ptr)
+ std::vector< std::pair < std::weak_ptr< Gate >, VisualGate > > getGates()
+ const bool isPlaceholder Clicked(int mouseX, int mouseY) const
+ const sf::Vector2f getPlaceholderPosition () const
+ const int getInitialState () const
+ void movePlaceholder (sf::Vector2f newPosition)
+ VisualQubit & operator =(const VisualQubit &qubit)
+ int getID() const
+ void resetQubit()

Public Member Functions

- VisualQubit (const sf::Vector2f &pos, const sf::Font &font, int id, int initialState=0)

  Constructs a VisualQubit with a specified position of the qubit.
- ∼VisualQubit ()=default

  Default destructor for the VisualQubit class.
- const void draw (sf::RenderWindow &window) const

Draws the initial state, the qubit and it's gates to the window.

- void switchInitialState ()

  Switches the initial state of the qubit to be either 0 or 1.
- const bool isInitialStageClicked (int mouseX, int mouseY) const

  Determines if the mouse click happened inside the initial state text.
- void addGate (const std::string &abbreviation, const sf::Font &font, std::weak_ptr< Gate > gate)

  Adds a new visual-logical gate pair to gates_.
- void addCNOTGate (VisualQubit &controlQubit, std::weak_ptr< CNOT > ptr)

  Adds a new CNOT gate to multiQubitGates_ that are drawn on screen.
- std::vector< std::pair< std::weak_ptr< Gate >, VisualGate > > getGates ()

  Get the Gates vector.
- const bool isPlaceholderClicked (int mouseX, int mouseY) const

  Determines if the mouse click happened inside the placeholder gate.
- const sf::Vector2f getPlaceholderPosition () const

  Get the Placeholder Position.
- const int getInitialState () const

  Get the Initial State of the qubit.
- void movePlaceholder (sf::Vector2f newPosition)

  Move placeholder gate to specified position.
- VisualQubit & operator= (const VisualQubit &qubit)

  Overloaded = operator.
- int getID () const

  Get the ID of the qubit.
- void resetQubit ()

  Clears all gates from the qubit and moves placeholder to leftmost position.

5.16.1 Detailed Description

A class visualizing a qubit in GUI.

5.16.2 Constructor & Destructor Documentation

5.16.2.1 VisualQubit()

```
VisualQubit::VisualQubit (
            const sf::Vector2f & pos,
            const sf::Font & font,
            int id,
            int initialState = 0)  [inline]
```

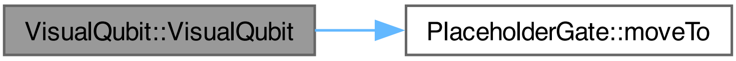
Constructs a VisualQubit with a specified position of the qubit.

Initializes a line to specified position with initial state of the qubit before the line.

Parameters

pos	Position where the qubit should be drawn in GUI.
font	The font that will be used for the texts.
id	The id of the qubit.
initialState	Initial state of the qubit, defaults to 0.

Here is the call graph for this function:



5.16.2.2 ∼VisualQubit()

```
VisualQubit::∼VisualQubit ()  [default]
```

Default destructor for the VisualQubit class.

5.16.3 Member Function Documentation

5.16.3.1 addCNOTGate()

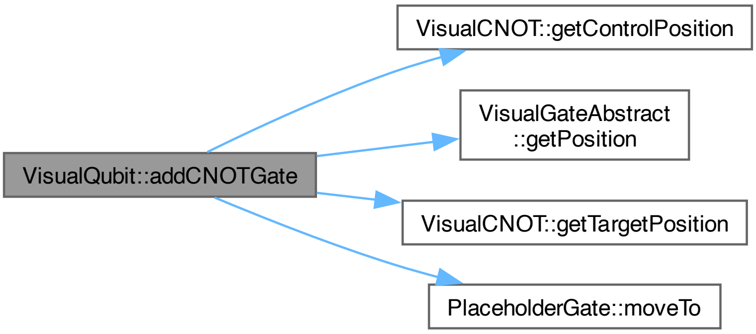
```
void VisualQubit::addCNOTGate (
            VisualQubit & controlQubit,
            std::weak_ptr< CNOT > ptr)  [inline]
```

Adds a new CNOT gate to multiQubitGates_ that are drawn on screen.

Parameters

controlQubit	Reference to the control qubit.
ptr	weak_ptr of the logical gate that will be paired with the visual one.

Here is the call graph for this function:


VisualQubit::addCNOTGate
VisualCNOT::getControlPosition
VisualGateAbstract ::getPosition
VisualCNOT::getTargetPosition
PlaceholderGate::moveTo

5.16.3.2 addGate()

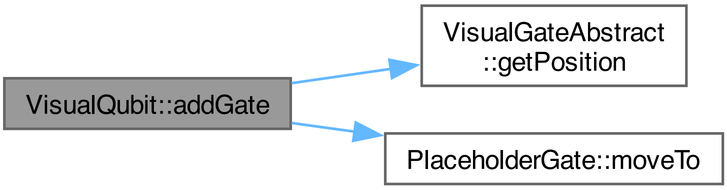
```
void VisualQubit::addGate (
            const std::string & abbreviation,
            const sf::Font & font,
            std::weak_ptr< Gate > gate)  [inline]
```

Adds a new visual-logical gate pair to gates_.

Parameters

abbreviation	Text that will be visible in the gate.
font	The font that will be used inside the gate.
gate	weak_ptr of the logical gate that will be paired with the visual one.

Here is the call graph for this function:



5.16.3.3 draw()

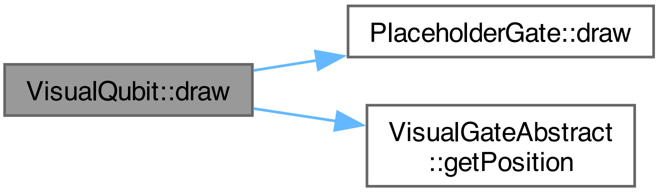
```
const void VisualQubit::draw (
            sf::RenderWindow & window) const  [inline]
```

Draws the initial state, the qubit and it's gates to the window.

Parameters

window	Window where everything will be drawn.

Here is the call graph for this function:



5.16.3.4 getGates()

```
std::vector< std::pair< std::weak_ptr< Gate >, VisualGate > > VisualQubit::getGates () [inline]
```

Get the Gates vector.

Returns

std::vector<std::pair<Gate, VisualGate>> vector of single-qubit gates

5.16.3.5 getID()

```
int VisualQubit::getID () const [inline]
```

Get the ID of the qubit.

Returns

int the ID of the qubit

5.16.3.6 getInitialState()

```
const int VisualQubit::getInitialState () const [inline]
```

Get the Initial State of the qubit.

Returns

const int 0 or 1 depending on the state

5.16.3.7 getPlaceholderPosition()

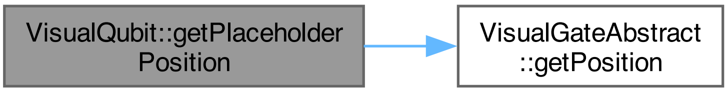
```
const sf::Vector2f VisualQubit::getPlaceholderPosition () const [inline]
```

Get the Placeholder Position.

Returns

const sf::Vector2f the position of the placeholder.

Here is the call graph for this function:


VisualQubit::getPlaceholder Position
VisualGateAbstract ::getPosition

5.16.3.8 isInitialStageClicked()

```
const bool VisualQubit::isInitialStageClicked (
            int mouseX,
            int mouseY) const [inline]
```

Determines if the mouse click happened inside the initial state text.

Parameters

mouseX	Mouse position on x axis.
mouseY	Mouse position on y axis.

Returns

True if initial state is clicked, false otherwise.

5.16.3.9 isPlaceholderClicked()

```
const bool VisualQubit::isPlaceholderClicked (
            int mouseX,
            int mouseY) const  [inline]
```

Determines if the mouse click happened inside the placeholder gate.

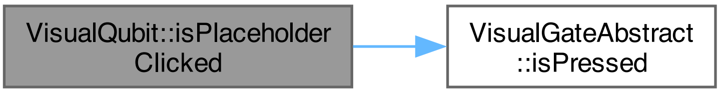
Parameters

mouseX	Mouse position on x axis.
mouseY	Mouse position on y axis.

Returns

True if the placeholder gate is clicked, false otherwise.

Here is the call graph for this function:


VisualQubit::isPlaceholder Clicked
VisualGateAbstract ::isPressed

5.16.3.10 movePlaceholder()

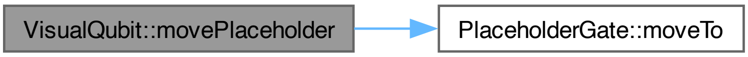
```
void VisualQubit::movePlaceholder (
            sf::Vector2f newPosition)  [inline]
```

Move placeholder gate to specified position.

Parameters

newPosition	Vector of the position the placeholder will be moved.

Here is the call graph for this function:



5.16.3.11 operator=()

```
VisualQubit & VisualQubit::operator= (
             const VisualQubit & qubit)  [inline]
```

Overloaded = operator.

Parameters

qubit	VisualQubit instance to be copied

Returns

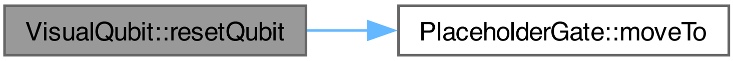
VisualQubit&

5.16.3.12 resetQubit()

```
void VisualQubit::resetQubit ()  [inline]
```

Clears all gates from the qubit and moves placeholder to leftmost position.

Here is the call graph for this function:



5.16.3.13 switchInitialState()

```
void VisualQubit::switchInitialState ()  [inline]
```

Switches the initial state of the qubit to be either 0 or 1.

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

- src/view/VisualQubit.hpp

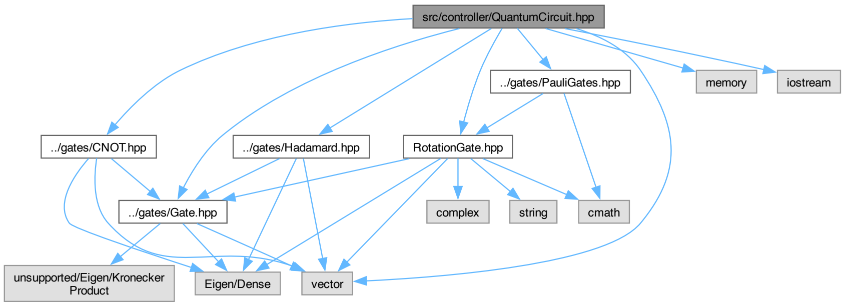
Chapter 6

File Documentation

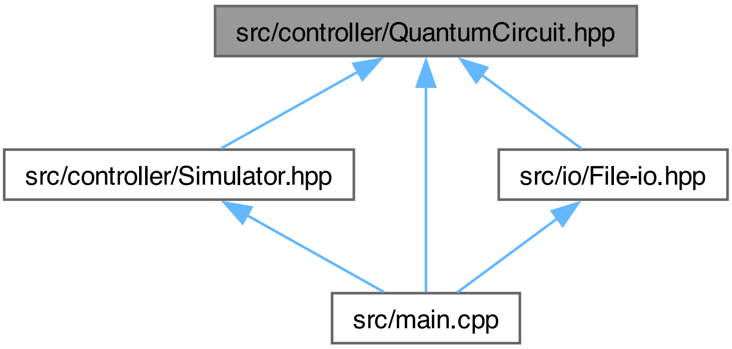
6.1 src/controller/QuantumCircuit.hpp File Reference

```
#include "../gates/Gate.hpp"
#include "../gates/PauliGates.hpp"
#include "../gates/RotationGate.hpp"
#include "../gates/CNOT.hpp"
#include "../gates/Hadamard.hpp"
#include <vector>
#include <memory>
#include <iostream>
```

Include dependency graph for QuantumCircuit.hpp:



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


src/controller/QuantumCircuit.hpp
src/controller/Simulator.hpp
src/io/File-io.hpp
src/main.cpp

Classes

- class QuantumCircuit

  A class representing a quantum circuit with a set of quantum gates.

6.2 QuantumCircuit.hpp

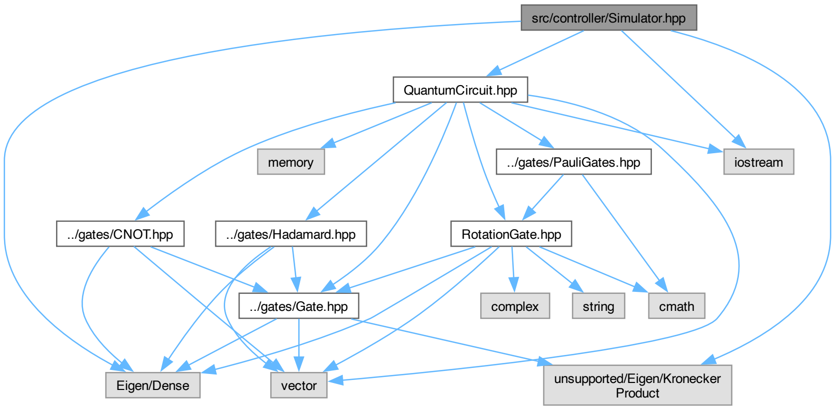
Go to the documentation of this file.

```
#ifndef QUANTUMCIRCUIT_HPP
#define QUANTUMCIRCUIT_HPP

#include "../gates/Gate.hpp"
#include "../gates/PauliGates.hpp"
#include "../gates/RotationGate.hpp"
#include "../gates/CNOT.hpp"
#include "../gates/Hadamard.hpp"
#include <vector>
#include <memory>
#include <iostream>

class QuantumCircuit {
public:
    QuantumCircuit(const std::vector<int>& qubits) : qubits_(qubits) {}

    QuantumCircuit() : qubits_({}) {}

    void addGate(const std::shared_ptr<Gate>& gate) {
        gates_.push_back(gate);
    }

    const std::vector<std::shared_ptr<Gate»& getGates() const {
        return gates_;
    }

    const std::vector<int>& getQubits() const {
        return qubits_;
    }

    void addQubit(int qubit) {
        qubits_.push_back(qubit);
    }

private:
    std::vector<int> qubits_;
    std::vector<std::shared_ptr<Gate» gates_;
};

#endif // QUANTUMCIRCUIT_HPP
```

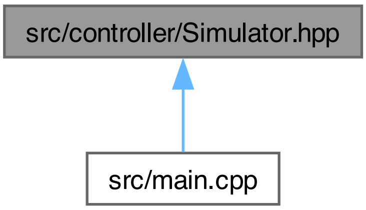
6.3 src/controller/Simulator.hpp File Reference

```
#include "Eigen/Dense"
#include "QuantumCircuit.hpp"
#include "unsupported/Eigen/KroneckerProduct"
#include <iostream>
```

Include dependency graph for Simulator.hpp:



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



Classes

- class Simulator

  Representing a simulator object, would be a quantum computer in the real world.

6.4 Simulator.hpp

Go to the documentation of this file.

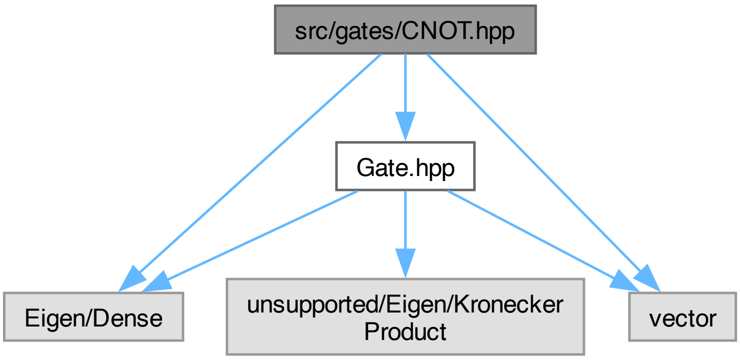
```
00001 #ifndef SIMULATOR_HPP
00002 #define SIMULATOR_HPP
00003
```

```
#include "Eigen/Dense"
#include "QuantumCircuit.hpp"
#include "unsupported/Eigen/KroneckerProduct"
#include <iostream>

class Simulator {

public:
  Simulator() {}

  Eigen::VectorXcd run(QuantumCircuit circ) const {
    std::vector<int> qubits = circ.getQubits();
    Eigen::VectorXcd mult(qubits.size());
    mult.setOnes();

    // Initialize the state of qubit(s)
    Eigen::VectorXcd state(2);
    Eigen::VectorXcd zero(2);
    Eigen::VectorXcd one(2);
    zero « 1, 0;
    one « 0, 1;

    if (qubits[0] == 1)
      state « one;
    else
      state « zero;

    // Create the initial state vector which is |0>^n
    for (int i = 1; i < qubits.size(); i++)
      if (qubits[i] == 1)
        state = Eigen::kroneckerProduct(state, one).eval();
      else
        state = Eigen::kroneckerProduct(state, zero).eval();

    // Evaluate gates
    for (const auto &gate : circ.getGates()) {
      state = gate->get_matrix(qubits.size()) * state;
    }
    return state;
  }

private:
};

#endif
```

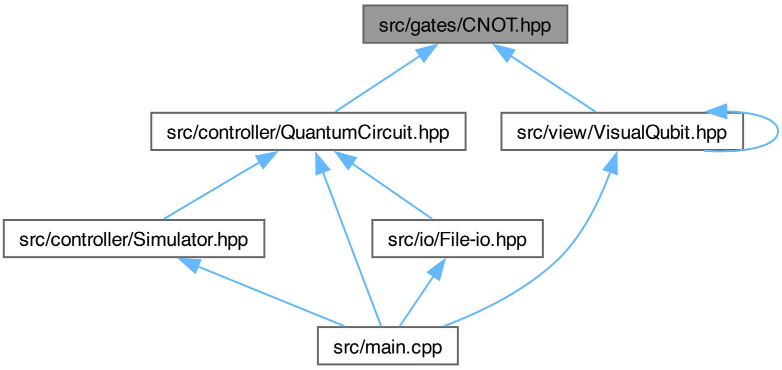
6.5 src/gates/CNOT.hpp File Reference

```
#include "Gate.hpp"
#include <Eigen/Dense>
#include <vector>
```

Include dependency graph for CNOT.hpp:



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



Classes

- class CNOT

6.6 CNOT.hpp

Go to the documentation of this file.

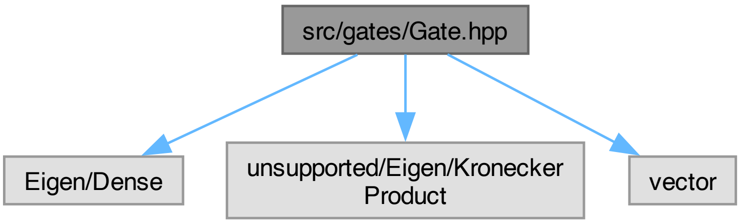
```
#ifndef CNOT_HPP
#define CNOT_HPP

#include "Gate.hpp"
#include <Eigen/Dense>
#include <vector>

class CNOT : public Gate {
public:
  CNOT(std::vector<int> qubits, std::vector<int> controls = {})
      : Gate(2, qubits, controls) {
    (*this)(0, 0) = 0;
    (*this)(0, 1) = 1;
    (*this)(1, 0) = 1;
    (*this)(1, 1) = 0;
  }

  std::string to_string() const { return "CNOT"; }
};

#endif
```

6.7 src/gates/Gate.hpp File Reference

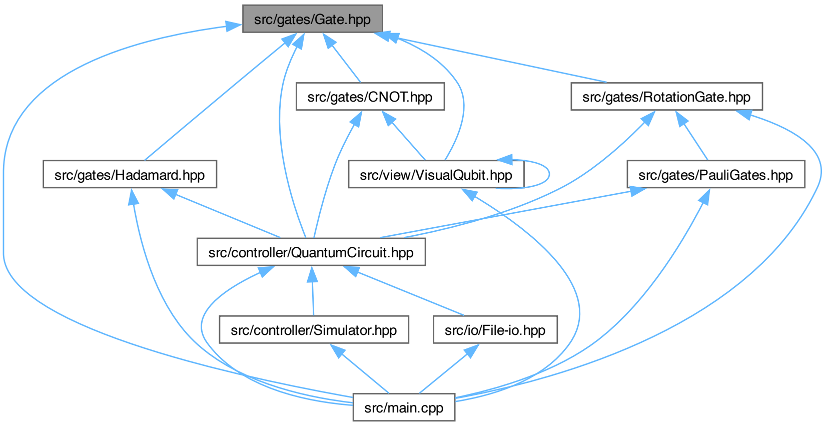
```
#include <Eigen/Dense>
#include <unsupported/Eigen/KroneckerProduct>
```

`#include <vector>`

Include dependency graph for Gate.hpp:



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



Classes

- class Gate

  A class representing a quantum gate, inheriting from Eigen::MatrixXcd.

6.8 Gate.hpp

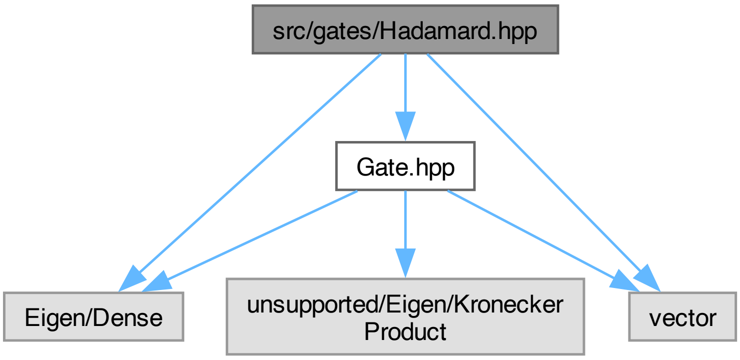
Go to the documentation of this file.

```
#ifndef GATE_HPP
#define GATE_HPP

#include <Eigen/Dense>
#include <unsupported/Eigen/KroneckerProduct>
#include <vector>

class Gate : public Eigen::MatrixXcd {
public:
  virtual std::string to_string() const = 0; // Pure virtual function

  virtual ~Gate() = default;

  Gate(int dim, const std::vector<int> &qubits,
       const std::vector<int> &controls = {})
      : Eigen::MatrixXcd(dim, dim), qubits(qubits), controls(controls) {
    this->setZero();
    P0x0 « 1, 0, 0, 0;
    P1x1 « 0, 0, 0, 1;
  }

  const std::vector<int> &get_qubits() const { return qubits; }

  const std::vector<int> &get_controls() const { return controls; }

  int num_qubits() const { return qubits.size(); }

  Eigen::MatrixXcd get_matrix(int n) {
    Eigen::MatrixXcd op = Eigen::MatrixXcd::Identity(1, 1);
    Eigen::MatrixXcd op2 = Eigen::MatrixXcd::Identity(1, 1);

    if (!controls.empty()) { // Multi-control gate logic
      for (int i = 0; i < n; ++i) {
        if (std::find(controls.begin(), controls.end(), i) != controls.end()) {
          op = Eigen::kroneckerProduct(op, P0x0).eval();
          op2 = Eigen::kroneckerProduct(op2, P1x1).eval();
        } else if (std::find(qubits.begin(), qubits.end(), i) != qubits.end()) {
          op = Eigen::kroneckerProduct(op, I).eval();
          op2 = Eigen::kroneckerProduct(op2, *this).eval();
        } else {
          op = Eigen::kroneckerProduct(op, I).eval();
          op2 = Eigen::kroneckerProduct(op2, I).eval();
        }
      }
      op += op2;
    } else { // Regular gates
      for (int i = 0; i < n; ++i) {
        if (std::find(qubits.begin(), qubits.end(), i) != qubits.end()) {
          op = Eigen::kroneckerProduct(op, *this).eval();
        } else {
          op = Eigen::kroneckerProduct(op, I).eval();
        }
      }
    }

    return op;
  }

private:
  // Projection operators
  Eigen::MatrixXcd P0x0 = Eigen::MatrixXcd(2, 2);
  Eigen::MatrixXcd P1x1 = Eigen::MatrixXcd(2, 2);
  Eigen::MatrixXcd I = Eigen::MatrixXcd::Identity(2, 2);

protected:
  std::vector<int> qubits;   // Target qubits for the gate
  std::vector<int> controls; // Control qubits for the gate
};

#endif // GATE_HPP
```

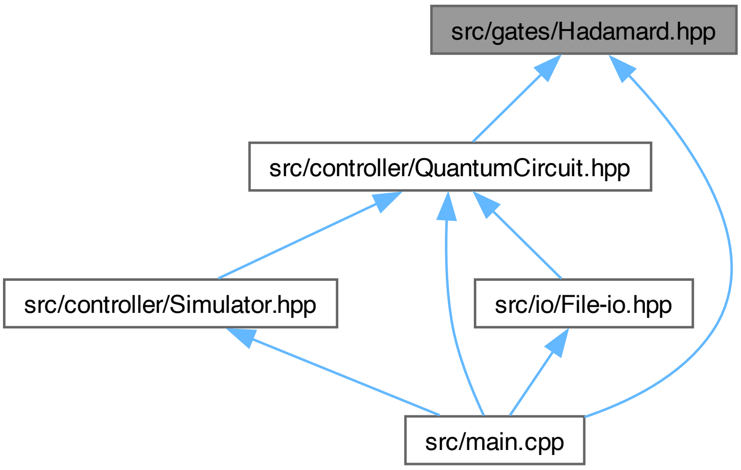
6.9 src/gates/Hadamard.hpp File Reference

```
#include "Gate.hpp"
#include <Eigen/Dense>
#include <vector>
```

Include dependency graph for Hadamard.hpp:



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


src/gates/Hadamard.hpp
src/controller/QuantumCircuit.hpp
src/controller/Simulator.hpp
src/io/File-io.hpp
src/main.cpp

Classes

- class H

6.10 Hadamard.hpp

Go to the documentation of this file.

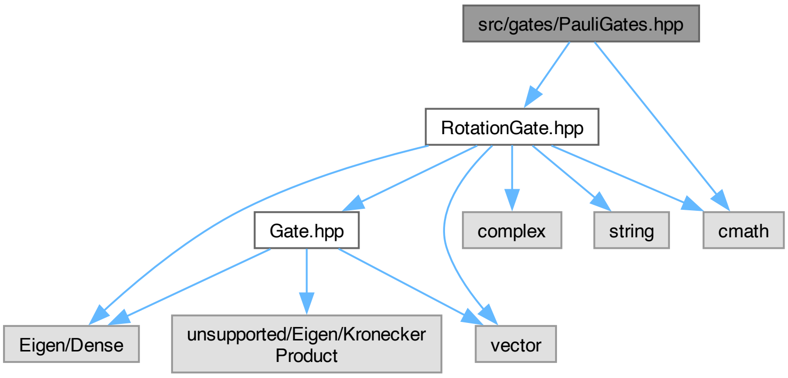
```
00001 #ifndef H_HPP
```

```
#define H_HPP

#include "Gate.hpp"
#include <Eigen/Dense>
#include <vector>

class H : public Gate {
public:
  H(std::vector<int> qubits, std::vector<int> controls = {})
      : Gate(2, qubits, controls) {
    (*this)(0, 0) = 1;
    (*this)(0, 1) = 1;
    (*this)(1, 0) = 1;
    (*this)(1, 1) = -1;
    (*this) *= 1 / std::sqrt(2);
  }

  std::string to_string() const { return "H"; }
};

#endif
```

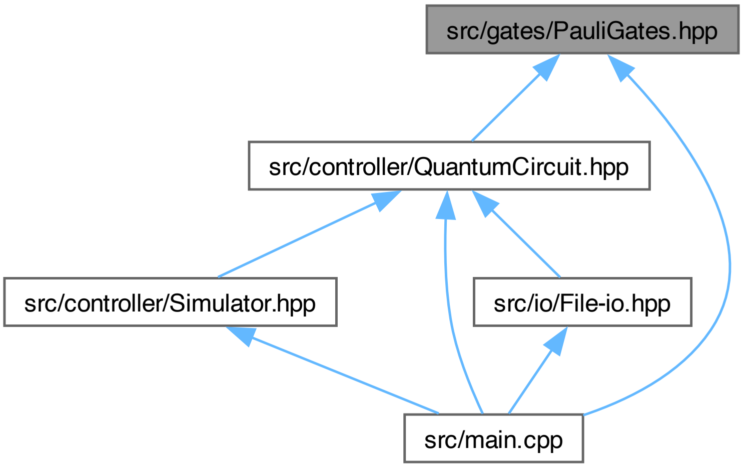
6.11 src/gates/PauliGates.hpp File Reference

```
#include "RotationGate.hpp"
#include <cmath>
```

Include dependency graph for PauliGates.hpp:



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


src/gates/PauliGates.hpp
src/controller/QuantumCircuit.hpp
src/controller/Simulator.hpp
src/io/File-io.hpp
src/main.cpp

Classes

- class PauliX

  A class representing a quantum rotation gate around the X axis axis by an angle of pi inheriting from the RotationGate class.
- class PauliY

  A class representing a quantum rotation gate around the Y axis axis by an angle of pi inheriting from the RotationGate class.
- class PauliZ

  A class representing a quantum rotation gate around the Z axis axis by an angle of pi inheriting from the RotationGate class Matrix form: (1,0) (0,0) (0,0) (-1,-0)

Macros

- #define _USE_MATH_DEFINES

6.11.1 Macro Definition Documentation

6.11.1.1 _USE_MATH_DEFINES

```
#define _USE_MATH_DEFINES
```

6.12 PauliGates.hpp

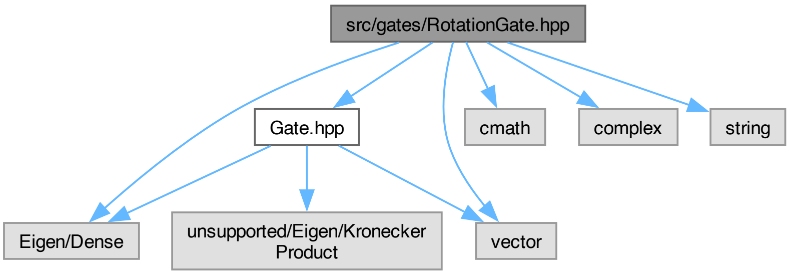
Go to the documentation of this file.

```
#ifndef PAULIGATES_HPP
#define PAULIGATES_HPP

#include "RotationGate.hpp"
#define _USE_MATH_DEFINES
#include <cmath>

class PauliX : public RotationGate {
public:
  PauliX(std::vector<int> qubits, std::vector<int> controls = {})
      : RotationGate(X, -M_PI, qubits, controls) {
    std::complex<double> i_minus(0.0, -1.0);
    (*this) « (i_minus * (*this));
  };
  std::string to_string() const { return "X"; }
  ~PauliX(){};
};
class PauliY : public RotationGate {
public:
  PauliY(std::vector<int> qubits, std::vector<int> controls = {})
      : RotationGate(Y, -M_PI, qubits, controls) {
    std::complex<double> i_minus(0.0, -1.0);
    (*this) « (i_minus * (*this));
  };
  std::string to_string() const { return "Y"; }
  ~PauliY(){};
};
class PauliZ : public RotationGate {
public:
  PauliZ(std::vector<int> qubits, std::vector<int> controls = {})
      : RotationGate(Z, -M_PI, qubits, controls) {
    std::complex<double> i_minus(0.0, -1.0);
    (*this) « (i_minus * (*this));
  };
  std::string to_string() const { return "Z"; }
  ~PauliZ(){};
};
#endif
```

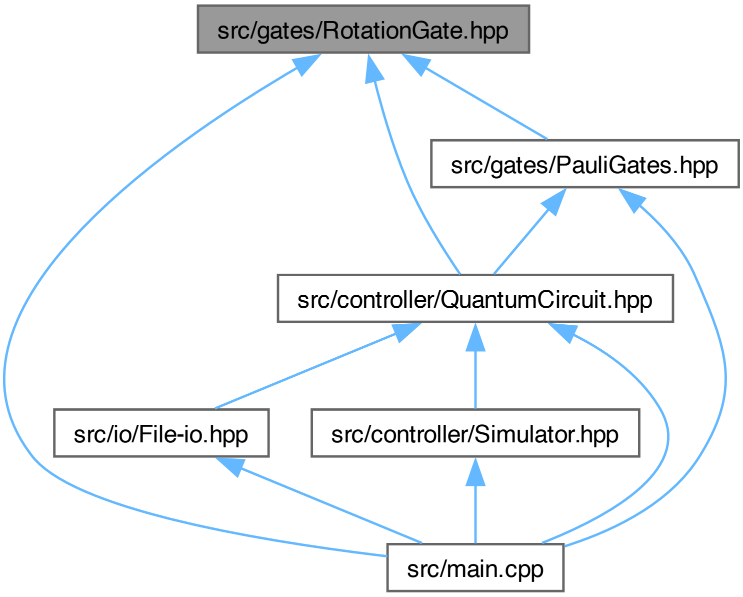
6.13 src/gates/RotationGate.hpp File Reference

```
#include "Gate.hpp"
#include <Eigen/Dense>
#include <cmath>
#include <complex>
#include <string>
#include <vector>
```

Include dependency graph for RotationGate.hpp:


src/gates/RotationGate.hpp
Gate.hpp
cmath
complex
string
Eigen/Dense
unsupported/Eigen/Kronecker Product
vector

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


src/gates/RotationGate.hpp
src/gates/PauliGates.hpp
src/controller/QuantumCircuit.hpp
src/io/File-io.hpp
src/controller/Simulator.hpp
src/main.cpp

Classes

- class RotationGate

  A class representing a quantum rotation gate around a set axis by an arbitrary angle, inheriting from the Gate class.

Enumerations

- enum Axis { X = 0 , Y = 1 , Z = 2 }

Functions

- std::complex< double > roundWithPrecision (std::complex< double > c)

6.13.1 Enumeration Type Documentation

6.13.1.1 Axis

```
enum Axis
```

Enumerator

X	
Y	
Z	

6.13.2 Function Documentation

6.13.2.1 roundWithPrecision()

```
std::complex< double > roundWithPrecision (
              std::complex< double > c)  [inline]
```

6.14 RotationGate.hpp

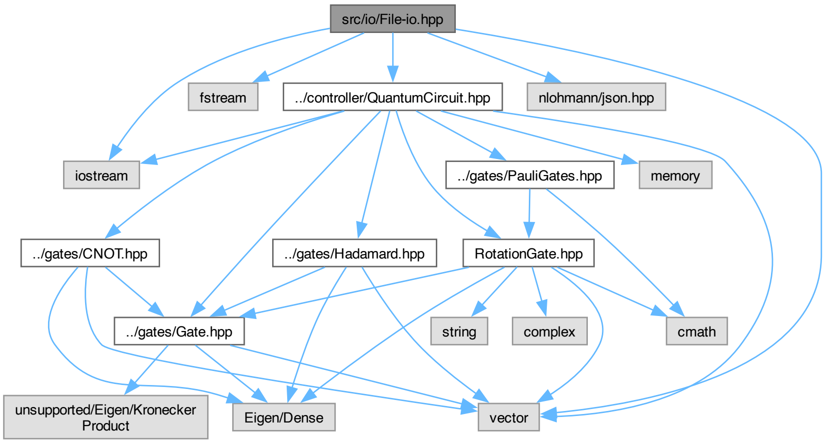
Go to the documentation of this file.

```
#ifndef ROTATIONGATE_HPP
#define ROTATIONGATE_HPP

#include "Gate.hpp"
#include <Eigen/Dense>
#include <cmath>
#include <complex>
#include <string>
#include <vector>

enum Axis {
  X = 0,
  Y = 1,
  Z = 2,
};

inline std::complex<double> roundWithPrecision(std::complex<double> c) {
  double n = 4.0;
  std::complex<double> result(
      std::round(c.real() * pow(10.0, n)) / pow(10.0, n),
      std::round(c.imag() * pow(10.0, n)) / pow(10.0, n));
  return result;
}
class RotationGate : public Gate {
public:
  ~RotationGate(){};

  RotationGate(Axis axis, double teta, std::vector<int> qubits,
               std::vector<int> controls = {})
      : Gate(2, qubits, controls) {
    teta_ = teta;
    std::complex<double> i(0.0, 1.0);
    std::complex<double> teta_comp(teta / 2, 0.0);
    switch (axis) {
    case Z:
      (*this)(0, 0) = roundWithPrecision(std::exp(-1.0 * i * teta_comp));
      (*this)(1, 1) = roundWithPrecision(std::exp(i * teta_comp));
      axis_ = "Z";
      break;
    case X:
      (*this)(0, 0) = roundWithPrecision(std::cos(teta_comp));
      (*this)(0, 1) = roundWithPrecision(i * -1.0 * std::sin(teta_comp));
      (*this)(1, 0) = roundWithPrecision(i * -1.0 * std::sin(teta_comp));
      (*this)(1, 1) = roundWithPrecision(std::cos(teta_comp));
      axis_ = "X";
      break;
    case Y:
      (*this)(0, 0) = roundWithPrecision(std::cos(teta_comp));
      (*this)(0, 1) = roundWithPrecision(-1.0 * std::sin(teta_comp));
      (*this)(1, 0) = roundWithPrecision(std::sin(teta_comp));
      (*this)(1, 1) = roundWithPrecision(std::cos(teta_comp));
      axis_ = "Y";
      break;
    default:
      axis_ = "reee";
      throw std::invalid_argument("Invalid axis");
    }
  }
  std::string to_string() const {
    std::string r = "R" + this->axis_ + "(" + std::to_string(teta_) + ")";
    return r;
  }

private:
  std::string axis_;
  double teta_;
};

#endif // ROTATIONGATE_HPP
```

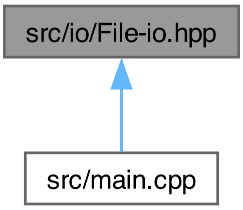
6.15 src/io/File-io.hpp File Reference

```
#include <iostream>
#include <fstream>
#include <vector>
#include <nlohmann/json.hpp>
#include "../controller/QuantumCircuit.hpp"
```

Include dependency graph for File-io.hpp:



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



Typedefs

- using json = nlohmann::json

Functions

- void writeCircuitToFile (const QuantumCircuit &circuit, const std::string &filename)

  Writes a quantum circuit to a JSON file.
- void readCircuitFromFile (QuantumCircuit &circuit, const std::string &filename)

  Reads a quantum circuit from a JSON file.

6.15.1 Typedef Documentation

6.15.1.1 json

```
using json = nlohmann::json
```

6.15.2 Function Documentation

6.15.2.1 readCircuitFromFile()

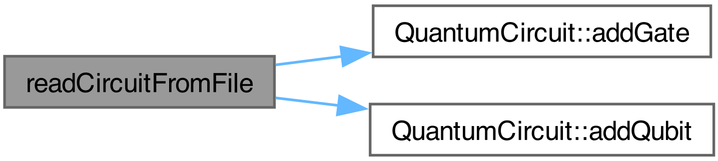
```
void readCircuitFromFile (
             QuantumCircuit & circuit,
             const std::string & filename)
```

Reads a quantum circuit from a JSON file.

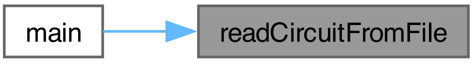
Parameters

circuit	QuantumCircuit where the contents of the file are added.
filename	Name of the file.

Here is the call graph for this function:



Here is the caller graph for this function:



6.15.2.2 writeCircuitToFile()

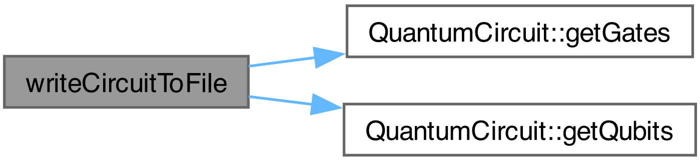
```
void writeCircuitToFile (
             const QuantumCircuit & circuit,
             const std::string & filename)
```

Writes a quantum circuit to a JSON file.

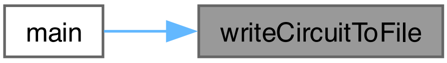
Parameters

circuit	QuantumCircuit that will be written into file.
filename	Name of the file.

Here is the call graph for this function:



Here is the caller graph for this function:



6.16 File-io.hpp

Go to the documentation of this file.

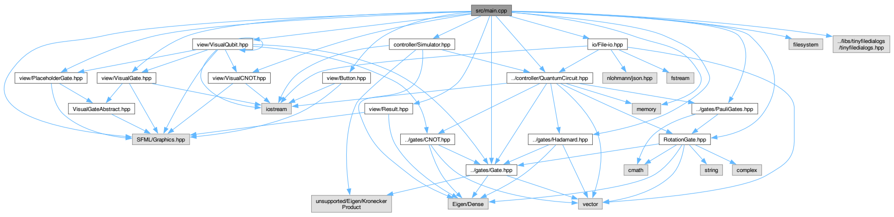
```
#ifndef FILE_IO_HPP
#define FILE_IO_HPP

#include <iostream>
#include <fstream>
#include <vector>
#include <nlohmann/json.hpp>
#include "../controller/QuantumCircuit.hpp"

using json = nlohmann::json;

void writeCircuitToFile(const QuantumCircuit& circuit, const std::string& filename) {
    json j;

    // Add number of qubits to JSON
    j["qubitStates"] = circuit.getQubits();

    // Store the gates
    for (const auto& gate : circuit.getGates()) {
        json gateJson;

        // Determine gate type and add it to JSON
        if (dynamic_cast<const PauliX*>(gate.get())) {
            gateJson["gate"] = "X";
        } else if (dynamic_cast<const PauliY*>(gate.get())) {
            gateJson["gate"] = "Y";
        } else if (dynamic_cast<const PauliZ*>(gate.get())) {
            gateJson["gate"] = "Z";
        } else if (dynamic_cast<const H*>(gate.get())) {
            gateJson["gate"] = "H";
        } else if (dynamic_cast<const CNOT*>(gate.get())) {
            gateJson["gate"] = "CNOT";
        }

        gateJson["qubits"] = gate->get_qubits();
        if (!gate->get_controls().empty()) {
            gateJson["controls"] = gate->get_controls();
        }

        j["gates"].push_back(gateJson);
    }

    // Write JSON to file
    std::ofstream file(filename);
    if (!file.is_open()) {
        throw std::ios_base::failure("Could not open file for writing.");
    }
    file « j.dump(4);
}

void readCircuitFromFile(QuantumCircuit& circuit, const std::string& filename) {
    std::ifstream file(filename);
    json j;
    file » j;

    for (int i = 0; i < j["qubitStates"].size(); ++i) {
        circuit.addQubit(j["qubitStates"].at(i));
    }

    for (const auto& gateJson : j["gates"]) {
        std::string gateType = gateJson["gate"];
        std::vector<int> qubits = gateJson["qubits"].get<std::vector<int»();
        std::vector<int> controls = gateJson.contains("controls") ?
gateJson["controls"].get<std::vector<int»() : std::vector<int>();

        if (gateType == "X") {
            circuit.addGate(std::make_shared<PauliX>(qubits, controls));
        } else if (gateType == "Y") {
            circuit.addGate(std::make_shared<PauliY>(qubits, controls));
        } else if (gateType == "Z") {
            circuit.addGate(std::make_shared<PauliZ>(qubits, controls));
        } else if (gateType == "H") {
            circuit.addGate(std::make_shared<H>(qubits, controls));
        } else if (gateType == "CNOT") {
            circuit.addGate(std::make_shared<CNOT>(qubits, controls));
        }
    }
}

#endif // FILE_IO_HPP
```

6.17 src/main.cpp File Reference

```
#include <SFML/Graphics.hpp>
#include <filesystem>
#include <memory>
#include "../libs/tinyfiledialogs/tinyfiledialogs.hpp"
#include "io/File-io.hpp"
#include "view/Button.hpp"
#include "view/PlaceholderGate.hpp"
#include "view/Result.hpp"
#include "view/VisualCNOT.hpp"
#include "view/VisualGate.hpp"
#include "view/VisualQubit.hpp"
#include "controller/QuantumCircuit.hpp"
#include "controller/Simulator.hpp"
#include "gates/Gate.hpp"
#include "gates/Hadamard.hpp"
#include "gates/PauliGates.hpp"
```

`#include "gates/RotationGate.hpp"`

Include dependency graph for main.cpp:



Functions

- int main ()

Variables

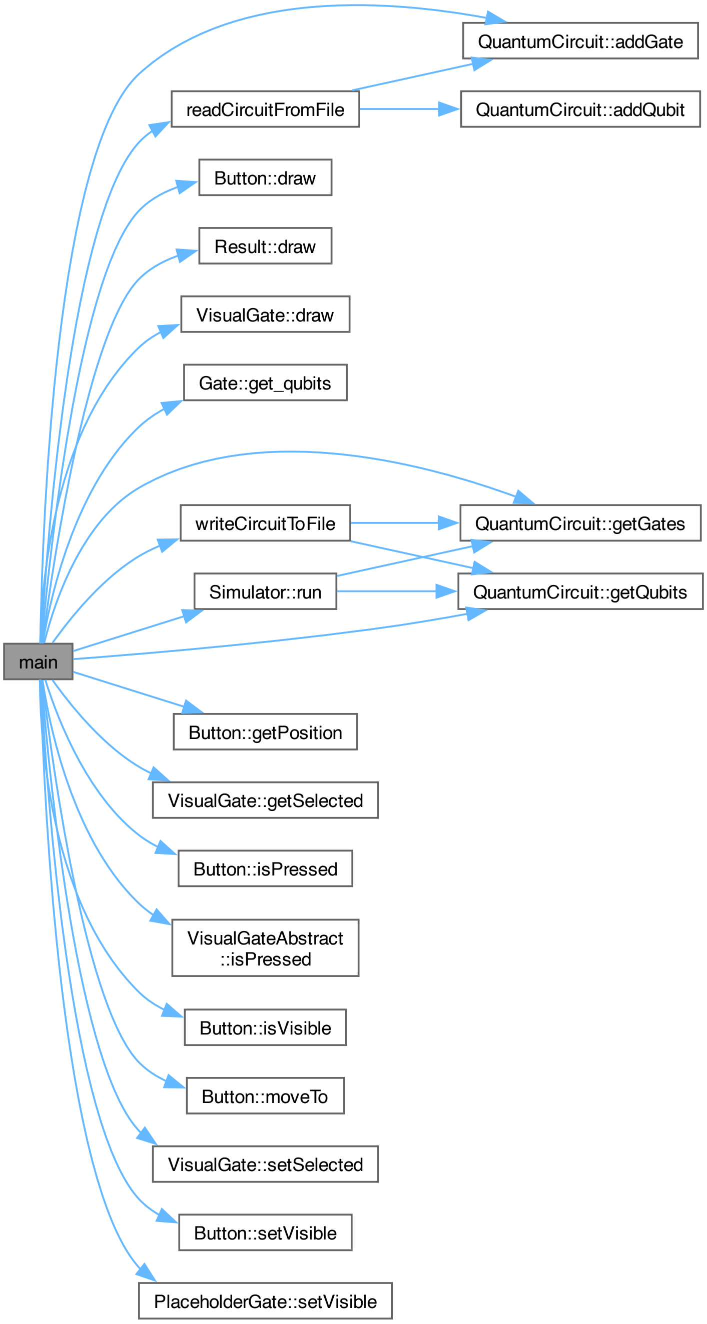
- int windowHeight = 800
- int windowWidth = 1400
- bool gateSelected = false
- char const ∗ fileFilterPatterns [1] = { "∗.json" }
- std::vector< std::shared_ptr< Gate > > gates
- sf::Font font
- std::vector< VisualQubit > qubits
- bool cnotControl = false
- std::vector< VisualQubit >::iterator controlQubit

6.17.1 Function Documentation

6.17.1.1 main()

```
int main ()
```

Here is the call graph for this function:



6.17.2 Variable Documentation

6.17.2.1 cnotControl

```
bool cnotControl = false
```

6.17.2.2 controlQubit

```
std::vector<VisualQubit>::iterator controlQubit
```

6.17.2.3 fileFilterPatterns

```
char const* fileFilterPatterns[1] = { "*.json" }
```

6.17.2.4 font

```
sf::Font font
```

6.17.2.5 gates

```
std::vector<std::shared_ptr<Gate> > gates
```

6.17.2.6 gateSelected

```
bool gateSelected = false
```

6.17.2.7 qubits

```
std::vector<VisualQubit> qubits
```

6.17.2.8 windowHeight

```
int windowHeight = 800
```

6.17.2.9 windowWidth

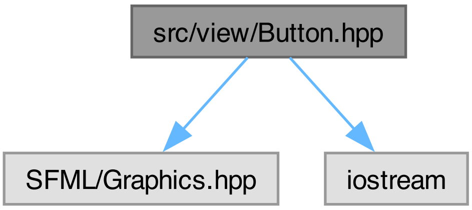
```
int windowWidth = 1400
```

6.18 src/readme.md File Reference

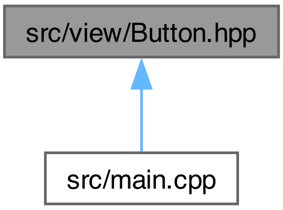
6.19 src/view/Button.hpp File Reference

```
#include <SFML/Graphics.hpp>
#include <iostream>
```

Include dependency graph for Button.hpp:



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



Classes

- class Button

  A class visualizing a button in GUI.

6.20 Button.hpp

Go to the documentation of this file.

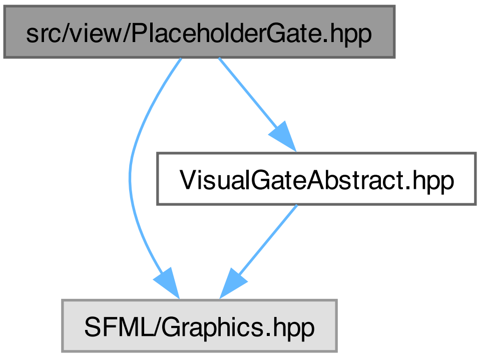
```
#ifndef BUTTON_HPP
#define BUTTON_HPP

#include <SFML/Graphics.hpp>
#include <iostream>
```

```

class Button {
  private:
    sf::RectangleShape button_;
    sf::Text text_;
    sf::Vector2f size_;
    bool visible_;

  public:
    Button(const sf::Vector2f& pos, const std::string& text, const sf::Font& font, bool visible = true) {
      text_.setFont(font);
      text_.setString(text);
      text_.setCharacterSize(24);
      text_.setFillColor(sf::Color::Black);
      text_.setOrigin(text_.getGlobalBounds().width / 2.f + text_.getLocalBounds().left, text_.getGlobalBounds().height / 2.f + text_.getLocalBounds().top);

      size_ = sf::Vector2f(text_.getGlobalBounds().width, text_.getGlobalBounds().height) + sf::Vector2f(15, 15);

      button_.setSize(size_);
      button_.setFillColor(sf::Color::White);
      button_.setPosition(pos);
      button_.setOutlineThickness(3.f);
      button_.setOutlineColor(sf::Color::Black);

      text_.setPosition(button_.getPosition() + (size_ / 2.f));

      visible_ = visible;
    }

    ~Button() = default;

    const void draw(sf::RenderWindow& window) const {
      if (visible_) {
        window.draw(button_);
        window.draw(text_);
      }
    }

    void moveTo(sf::Vector2f newPosition) {
      button_.setPosition(newPosition);
      text_.setPosition(newPosition + (size_ / 2.f));
    }

    const sf::Vector2f getPosition() const {
      return button_.getPosition();
    }

    bool isPressed(int mouseX, int mouseY) const {
      int gateX = button_.getPosition().x;
      int gateY = button_.getPosition().y;

      if ((gateX <= mouseX && mouseX <= (gateX + size_.x)) && (gateY <= mouseY && mouseY <= (gateY + size_.y)))
        return true;
      else
        return false;
    }

    void setVisible(bool visible) {
      visible_ = visible;
    }

    const bool isVisible() const {
      return visible_;
    }
};

#endif // BUTTON_HPP
```

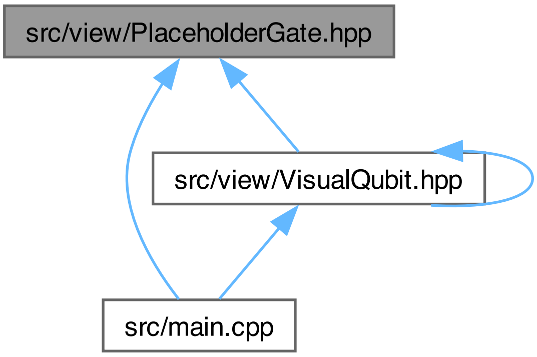
6.21 src/view/PlaceholderGate.hpp File Reference

```
#include <SFML/Graphics.hpp>
#include "VisualGateAbstract.hpp"
```

Include dependency graph for PlaceholderGate.hpp:



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



Classes

- class PlaceholderGate

  A class visualizing a placeholder gate in GUI.

6.22 PlaceholderGate.hpp

Go to the documentation of this file.

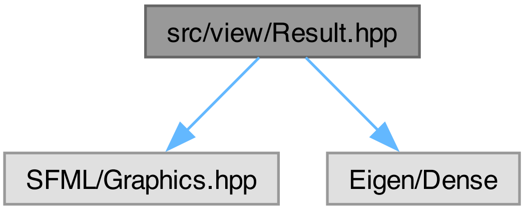
```
00001 #ifndef PLACEHOLDER_GATE_HPP
00002 #define PLACEHOLDER_GATE_HPP
00003
```

```
#include <SFML/Graphics.hpp>

#include "VisualGateAbstract.hpp"

class PlaceholderGate : public VisualGateAbstract {
  private:
    static bool visible_;

  public:
    PlaceholderGate() {
      gate_.setSize(sf::Vector2f(size_, size_));
      gate_.setFillColor(sf::Color(255, 0 , 0 , 100));
      gate_.setOrigin(gate_.getSize() / 2.f);
    }

    PlaceholderGate(const sf::Vector2f& pos) {
      gate_.setSize(sf::Vector2f(size_, size_));
      gate_.setFillColor(sf::Color(255, 0 , 0 , 0));
      gate_.setPosition(pos);
      gate_.setOrigin(gate_.getSize() / 2.f);
    }

    virtual ~PlaceholderGate() = default;

    const void draw(sf::RenderWindow& window) const override {
      if (visible_)
        window.draw(gate_);
    }

    void moveTo(sf::Vector2f newPosition) override {
      gate_.setPosition(newPosition);
    }

    static void setVisible(bool visible) {
      visible_ = visible;
    }

    const bool isVisible() const {
      return visible_;
    }

    const sf::Vector2f getSize() const {
      return gate_.getSize();
    }
};

bool PlaceholderGate::visible_ = false;

#endif // PLACEHOLDER_GATE_HPP
```

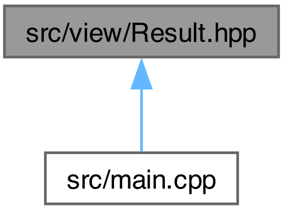
6.23 src/view/Result.hpp File Reference

```
#include <SFML/Graphics.hpp>
#include <Eigen/Dense>
```

Include dependency graph for Result.hpp:



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



Classes

- class Result

  A class visualizing the result of the quantum computer simulation.

6.24 Result.hpp

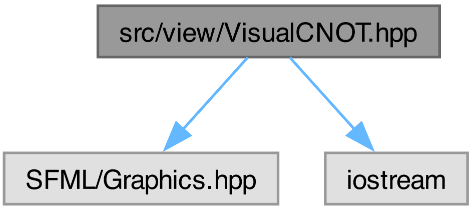
Go to the documentation of this file.

```
#ifndef RESULT_HPP
#define RESULT_HPP

#include <SFML/Graphics.hpp>
#include <Eigen/Dense>

class Result {
  private:
    sf::Text text_;
    sf::Text result_;

  public:
    Result() = default;

    Result(const sf::Vector2f& pos, const Eigen::VectorXcd& result, const sf::Font& font) {
      text_.setFont(font);
      text_.setString("Result:");
      text_.setCharacterSize(32);
      text_.setFillColor(sf::Color::Black);
      text_.setPosition(pos);

      std::stringstream ss;
      ss « result;

      result_.setFont(font);
      result_.setString(ss.str());
      result_.setCharacterSize(32);
      result_.setFillColor(sf::Color::Black);
      result_.setPosition(pos + sf::Vector2f(120, 0));
    }

    virtual ~Result() = default;

    const void draw(sf::RenderWindow& window) const {
      window.draw(text_);
      window.draw(result_);
    }

    void moveTo(sf::Vector2f newPosition) {
      text_.setPosition(newPosition);
      result_.setPosition(newPosition + sf::Vector2f(120, 0));
    }

    Result& operator=(const Result& result) {
      if (this != &result) {
          text_ = result.text_;
          result_ = result.result_;
      }
      return *this;
    }
};

#endif // RESULT_HPP
```

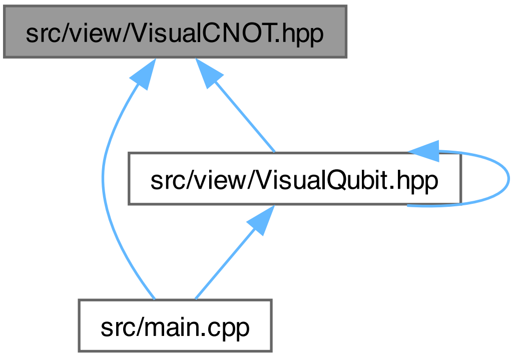
6.25 src/view/VisualCNOT.hpp File Reference

```
#include <SFML/Graphics.hpp>
#include <iostream>
```

Include dependency graph for VisualCNOT.hpp:



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



Classes

- class VisualCNOT

  A class visualizing a CNOT gate in GUI.

6.26 VisualCNOT.hpp

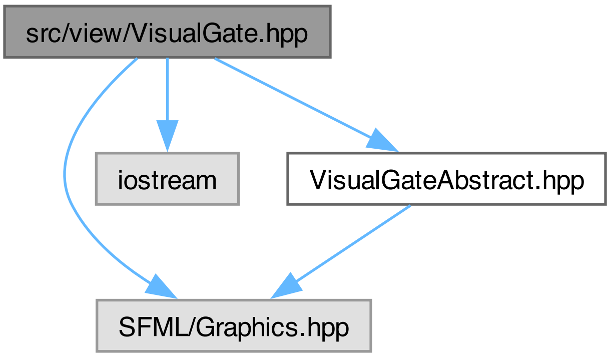
Go to the documentation of this file.

```
#ifndef VISUAL_CNOT_HPP
#define VISUAL_CNOT_HPP

#include <SFML/Graphics.hpp>
#include <iostream>

class VisualCNOT {
  private:
    sf::CircleShape control_;
    sf::CircleShape target_;
    sf::RectangleShape connector_;

  public:
    VisualCNOT(const sf::Vector2f& controlPos, const sf::Vector2f& targetPos) {
      control_.setRadius(15);
      control_.setFillColor(sf::Color::Black);
      control_.setPosition(controlPos);
      control_.setOrigin(control_.getGlobalBounds().width / 2.f + control_.getLocalBounds().left, control_.getGlobalBounds().height / 2.f + control_.getLocalBounds().top);

      target_.setRadius(20);
      target_.setFillColor(sf::Color::White);
      target_.setPosition(targetPos);
      target_.setOutlineThickness(5);
      target_.setOutlineColor(sf::Color::Black);
      target_.setOrigin(target_.getGlobalBounds().width / 2.f + target_.getLocalBounds().left, target_.getGlobalBounds().height / 2.f + target_.getLocalBounds().top);

      connector_.setSize(sf::Vector2f((controlPos.y - targetPos.y), 4));
      connector_.setOrigin(connector_.getPosition() + sf::Vector2f(2, 0));
      connector_.setFillColor(sf::Color::Black);
      connector_.setPosition(controlPos);
      connector_.rotate(-90.f);
    }

    ~VisualCNOT() = default;

    const void draw(sf::RenderWindow& window) const {
      window.draw(connector_);
      window.draw(control_);
      window.draw(target_);
    }

    const sf::Vector2f getControlPosition() const {
      return control_.getPosition();
    }

    const sf::Vector2f getTargetPosition() const {
      return target_.getPosition();
    }
};

#endif // VISUAL_CNOT_HPP
```

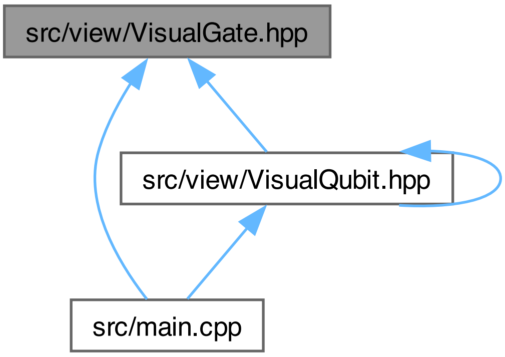
6.27 src/view/VisualGate.hpp File Reference

```
#include <SFML/Graphics.hpp>
#include <iostream>
#include "VisualGateAbstract.hpp"
```

Include dependency graph for VisualGate.hpp:



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



Classes

- class VisualGate

  A class visualizing a quantum gate in GUI.

6.28 VisualGate.hpp

Go to the documentation of this file.

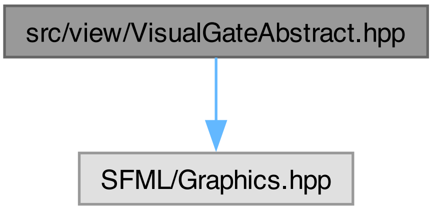
```
00001 #ifndef VISUAL_GATE_HPP
00002 #define VISUAL_GATE_HPP
00003
```

```
#include <SFML/Graphics.hpp>
#include <iostream>

#include "VisualGateAbstract.hpp"

class VisualGate : public VisualGateAbstract {
  private:
    sf::Text text_;
    bool selected_ = false;

  public:
    VisualGate(const sf::Vector2f& pos, const std::string& abbreviation, const sf::Font& font) {
      gate_.setSize(sf::Vector2f(size_, size_));
      gate_.setFillColor(sf::Color::White);
      gate_.setPosition(pos);
      gate_.setOutlineThickness(5.f);
      gate_.setOutlineColor(sf::Color::Black);
      gate_.setOrigin(gate_.getSize() / 2.f);

      text_.setFont(font);
      text_.setString(abbreviation);
      text_.setCharacterSize(32);
      text_.setFillColor(sf::Color::Black);
      text_.setOrigin(text_.getGlobalBounds().width / 2.f + text_.getLocalBounds().left, text_.getGlobalBounds().height / 2.f + text_.getLocalBounds().top);
      text_.setPosition(gate_.getPosition());
    }

    virtual ~VisualGate() = default;

    const void draw(sf::RenderWindow& window) const override {
      window.draw(gate_);
      window.draw(text_);
    }

    void moveTo(sf::Vector2f newPosition) override {
      gate_.setPosition(newPosition);
      text_.setPosition(newPosition);
    }

    const bool getSelected() const {
      return selected_;
    }

    void setSelected(bool selected) {
      if (selected)
        gate_.setFillColor(sf::Color::Red);
      else
        gate_.setFillColor(sf::Color::White);

      selected_ = selected;
    }
};

#endif // VISUAL_GATE_HPP
```

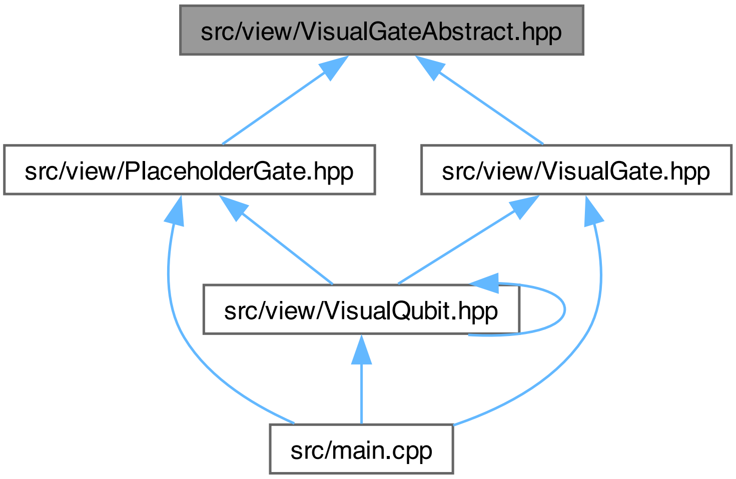
6.29 src/view/VisualGateAbstract.hpp File Reference

`#include <SFML/Graphics.hpp>`

Include dependency graph for VisualGateAbstract.hpp:



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



Classes

- class VisualGateAbstract

  An abstract class for visual gate-like classes.

6.30 VisualGateAbstract.hpp

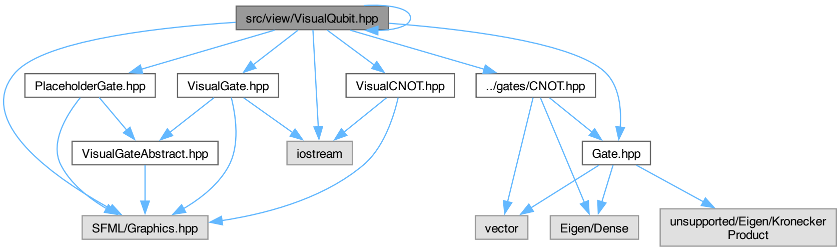
Go to the documentation of this file.

```
#ifndef VISUAL_GATE_ABSTRACT_HPP
#define VISUAL_GATE_ABSTRACT_HPP

#include <SFML/Graphics.hpp>

class VisualGateAbstract {
  protected:
    sf::RectangleShape gate_;
    int size_ = 90;

  public:
    VisualGateAbstract() = default;

    virtual ~VisualGateAbstract() = default;

    virtual const void draw(sf::RenderWindow& window) const = 0;

    virtual void moveTo(sf::Vector2f newPosition) = 0;

    bool isPressed(int mouseX, int mouseY) const {
      int gateX = gate_.getPosition().x;
      int gateY = gate_.getPosition().y;

      if (((gateX - (size_ / 2)) <= mouseX && mouseX <= (gateX + (size_ / 2))) && ((gateY - (size_ / 2)) <= mouseY && mouseY <= (gateY + (size_ / 2))))
        return true;
      else
        return false;
    }

    const sf::Vector2f getPosition() const {
      return gate_.getPosition();
    }
};

#endif // VISUAL_GATE_ABSTRACT_HPP
```

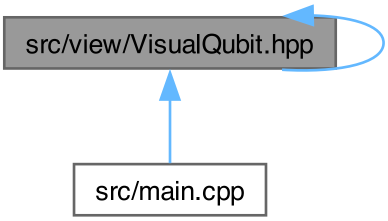
6.31 src/view/VisualQubit.hpp File Reference

```
#include <SFML/Graphics.hpp>
#include <iostream>
#include "PlaceholderGate.hpp"
#include "VisualCNOT.hpp"
#include "VisualGate.hpp"
#include "VisualQubit.hpp"
#include "../gates/CNOT.hpp"
#include "../gates/Gate.hpp"
```

Include dependency graph for VisualQubit.hpp:



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



Classes

- class VisualQubit

  A class visualizing a qubit in GUI.

6.32 VisualQubit.hpp

Go to the documentation of this file.

```
00001 #ifndef VISUAL_QUBIT_HPP
00002 #define VISUAL_QUBIT_HPP
00003
00004 #include <SFML/Graphics.hpp>
```

```
#include <iostream>

#include "PlaceholderGate.hpp"
#include "VisualCNOT.hpp"
#include "VisualGate.hpp"
#include "VisualQubit.hpp"

#include "../gates/CNOT.hpp"
#include "../gates/Gate.hpp"

class VisualQubit {
  private:
    sf::RectangleShape qubit_;
    sf::Text text_;
    int id_;
    int initialState_;
    std::vector<std::pair<std::weak_ptr<Gate>, VisualGate» gates_;
    std::vector<std::pair<std::weak_ptr<CNOT>, VisualCNOT»
        multiQubitGates_; // if the gate is target and the gate
    PlaceholderGate placeholder_;

  public:
    VisualQubit(const sf::Vector2f &pos, const sf::Font &font, int id, int initialState = 0)
    : id_(id), initialState_(initialState) {
      qubit_.setSize(sf::Vector2f(800, 5));
      qubit_.setFillColor(sf::Color::Black);
      qubit_.setPosition(pos);

      std::stringstream ss;
      ss « "|" « initialState_ « ">";

      text_.setFont(font);
      text_.setString(ss.str());
      text_.setCharacterSize(32);
      text_.setFillColor(sf::Color::Black);
      text_.setOrigin(text_.getGlobalBounds().width + text_.getLocalBounds().left,
                      text_.getGlobalBounds().height / 2.f +
                          text_.getLocalBounds().top);
      text_.setPosition(qubit_.getPosition() - sf::Vector2f(20, -2));

      placeholder_.moveTo(pos + sf::Vector2f(65, 0));
    }

    ~VisualQubit() = default;

    const void draw(sf::RenderWindow &window) const {
      window.draw(qubit_);
      window.draw(text_);

      for (auto gate : gates_) {
        gate.second.draw(window);
      }

      for (auto gate : multiQubitGates_) {
        if (auto gateptr = gate.first.lock()) {
          if (gateptr->get_qubits().at(0) == id_)
            gate.second.draw(window);
        }
      }

      if (gates_.size() + multiQubitGates_.size() < 7 &&
          placeholder_.getPosition().x < 1060)
        placeholder_.draw(window);
    }

    void switchInitialState() {
      initialState_ == 0 ? initialState_ = 1 : initialState_ = 0;

      std::stringstream ss;
      ss « "|" « initialState_ « ">";

      text_.setString(ss.str());
    }

    const bool isInitialStageClicked(int mouseX, int mouseY) const {
      return text_.getGlobalBounds().contains(mouseX, mouseY);
    }

    void addGate(const std::string &abbreviation, const sf::Font &font,
                 std::weak_ptr<Gate> gate) {
      std::pair<std::weak_ptr<Gate>, VisualGate> newGate(
          gate, VisualGate(placeholder_.getPosition(), abbreviation, font));
      gates_.push_back(newGate);
      placeholder_.moveTo(newGate.second.getPosition() + sf::Vector2f(110, 0));
    }

    void addCNOTGate(VisualQubit &controlQubit, std::weak_ptr<CNOT> ptr) {
      if (controlQubit.getPlaceholderPosition() != placeholder_.getPosition()) {
        sf::Vector2f ctrlQubitPosition = controlQubit.getPlaceholderPosition();
        sf::Vector2f controlPosition;

        ctrlQubitPosition.x > placeholder_.getPosition().x
            ? controlPosition = ctrlQubitPosition
            : controlPosition =
                  sf::Vector2f(placeholder_.getPosition().x, ctrlQubitPosition.y);

        VisualCNOT gate(
            controlPosition,
            sf::Vector2f(controlPosition.x, placeholder_.getPosition().y));
        multiQubitGates_.push_back(std::make_pair(ptr, gate));
        placeholder_.moveTo(gate.getTargetPosition() + sf::Vector2f(110, 0));
        controlQubit.movePlaceholder(gate.getControlPosition() +
                                     sf::Vector2f(110, 0));
      }
    }

    std::vector<std::pair<std::weak_ptr<Gate>, VisualGate» getGates() {
      return gates_;
    }

    const bool isPlaceholderClicked(int mouseX, int mouseY) const {
      return gates_.size() + multiQubitGates_.size() < 7
                 ? placeholder_.isPressed(mouseX, mouseY)
                 : false;
    }

    const sf::Vector2f getPlaceholderPosition() const {
      return placeholder_.getPosition();
    }

    const int getInitialState() const { return initialState_; }

    void movePlaceholder(sf::Vector2f newPosition) {
      placeholder_.moveTo(newPosition);
    }

    VisualQubit &operator=(const VisualQubit &qubit) {
      if (this != &qubit) {
        qubit_ = qubit.qubit_;
        text_ = qubit.text_;
        initialState_ = qubit.initialState_;
        gates_ = qubit.gates_;
        multiQubitGates_ = qubit.multiQubitGates_;
        placeholder_ = qubit.placeholder_;
      }
      return *this;
    }

    int getID() const { return id_; }

    void resetQubit() {
      gates_.clear();
      initialState_ = 0;
      placeholder_.moveTo(qubit_.getPosition() + sf::Vector2f(65, 0));
    }
};

#endif // VISUAL_QUBIT_HPP
```

Index

- _USE_MATH_DEFINES
  - PauliGates.hpp, 84
- ∼Button
  - Button, 10
- ∼Gate
  - Gate, 19
- ∼PauliX
  - PauliX, 28
- ∼PauliY
  - PauliY, 32
- ∼PauliZ
  - PauliZ, 36
- ∼PlaceholderGate
  - PlaceholderGate, 40
- ∼Result
  - Result, 47
- ∼RotationGate
  - RotationGate, 51
- ∼VisualCNOT
  - VisualCNOT, 56
- ∼VisualGate
  - VisualGate, 60
- ∼VisualGateAbstract
  - VisualGateAbstract, 65
- ∼VisualQubit
  - VisualQubit, 70

- addCNOTGate
  - VisualQubit, 70
- addGate
  - QuantumCircuit, 43
  - VisualQubit, 70
- addQubit
  - QuantumCircuit, 44
- Axis
  - RotationGate.hpp, 86

- Button, 9
  - ∼Button, 10
  - Button, 10
  - draw, 11
  - getPosition, 11
  - isPressed, 11
  - isVisible, 12
  - moveTo, 12
  - setVisible, 13

- CNOT, 14
  - CNOT, 16
  - to_string, 16
- cnotControl
  - main.cpp, 93
- controlQubit
  - main.cpp, 93
- controls
  - Gate, 21

- draw
  - Button, 11
  - PlaceholderGate, 40
  - Result, 47
  - VisualCNOT, 56
  - VisualGate, 61
  - VisualGateAbstract, 65
  - VisualQubit, 71

- File-io.hpp
  - json, 89
  - readCircuitFromFile, 89
  - writeCircuitToFile, 89
- fileFilterPatterns
  - main.cpp, 94
- font
  - main.cpp, 94

- Gate, 17
  - ∼Gate, 19
  - controls, 21
  - Gate, 19
  - get_controls, 19
  - get_matrix, 19
  - get_qubits, 20
  - num_qubits, 20
  - qubits, 21
  - to_string, 20
- gate_
  - VisualGateAbstract, 67
- gates
  - main.cpp, 94
- gateSelected
  - main.cpp, 94
- get_controls
  - Gate, 19
- get_matrix
  - Gate, 19
- get_qubits
  - Gate, 20
- getControlPosition
  - VisualCNOT, 56
- getGates

QuantumCircuit, 44
VisualQubit, 72
getID
VisualQubit, 72
getInitialState
VisualQubit, 72
getPlaceholderPosition
VisualQubit, 72
getPosition
Button, 11
VisualGateAbstract, 65
getQubits
QuantumCircuit, 45
getSelected
VisualGate, 61
getSize
PlaceholderGate, 40
getTargetPosition
VisualCNOT, 56

H, 21
H, 24
to_string, 24

isInitialStageClicked
VisualQubit, 72
isPlaceholderClicked
VisualQubit, 73
isPressed
Button, 11
VisualGateAbstract, 66
isVisible
Button, 12
PlaceholderGate, 40

json
File-io.hpp, 89

main
main.cpp, 92
main.cpp
cnotControl, 93
controlQubit, 93
fileFilterPatterns, 94
font, 94
gates, 94
gateSelected, 94
main, 92
qubits, 94
windowHeight, 94
windowWidth, 94
movePlaceholder
VisualQubit, 73
moveTo
Button, 12
PlaceholderGate, 41
Result, 47
VisualGate, 61
VisualGateAbstract, 66

num_qubits
Gate, 20

operator=
Result, 48
VisualQubit, 74

PauliGates.hpp
_USE_MATH_DEFINES, 84
PauliX, 25
∼PauliX, 28
PauliX, 27
to_string, 28
PauliY, 28
∼PauliY, 32
PauliY, 31
to_string, 32
PauliZ, 32
∼PauliZ, 36
PauliZ, 35
to_string, 36
PlaceholderGate, 36
∼PlaceholderGate, 40
draw, 40
getSize, 40
isVisible, 40
moveTo, 41
PlaceholderGate, 39
setVisible, 41

QuantumCircuit, 42
addGate, 43
addQubit, 44
getGates, 44
getQubits, 45
QuantumCircuit, 43
qubits
Gate, 21
main.cpp, 94

readCircuitFromFile
File-io.hpp, 89
resetQubit
VisualQubit, 74
Result, 46
∼Result, 47
draw, 47
moveTo, 47
operator=, 48
Result, 47
RotationGate, 48
∼RotationGate, 51
RotationGate, 51
to_string, 52
RotationGate.hpp
Axis, 86
roundWithPrecision, 87
X, 86
Y, 86

Z, 86
roundWithPrecision
RotationGate.hpp, 87
run
Simulator, 53

setSelected
VisualGate, 62
setVisible
Button, 13
PlaceholderGate, 41
Simulator, 52
run, 53
Simulator, 53
size_
VisualGateAbstract, 67
Source content, 1
src/controller/QuantumCircuit.hpp, 75, 76
src/controller/Simulator.hpp, 77
src/gates/CNOT.hpp, 78, 79
src/gates/Gate.hpp, 79, 80
src/gates/Hadamard.hpp, 81, 82
src/gates/PauliGates.hpp, 83, 85
src/gates/RotationGate.hpp, 85, 87
src/io/File-io.hpp, 88, 90
src/main.cpp, 91
src/readme.md, 95
src/view/Button.hpp, 95
src/view/PlaceholderGate.hpp, 96, 97
src/view/Result.hpp, 98, 99
src/view/VisualCNOT.hpp, 100, 101
src/view/VisualGate.hpp, 101, 102
src/view/VisualGateAbstract.hpp, 103, 104
src/view/VisualQubit.hpp, 105
switchInitialState
VisualQubit, 74

to_string
CNOT, 16
Gate, 20
H, 24
PauliX, 28
PauliY, 32
PauliZ, 36
RotationGate, 52

VisualCNOT, 54
∼VisualCNOT, 56
draw, 56
getControlPosition, 56
getTargetPosition, 56
VisualCNOT, 55
VisualGate, 57
∼VisualGate, 60
draw, 61
getSelected, 61
moveTo, 61
setSelected, 62
VisualGate, 60
VisualGateAbstract, 63
∼VisualGateAbstract, 65
draw, 65
gate_, 67
getPosition, 65
isPressed, 66
moveTo, 66
size_, 67
VisualGateAbstract, 65
VisualQubit, 67
∼VisualQubit, 70
addCNOTGate, 70
addGate, 70
draw, 71
getGates, 72
getID, 72
getInitialState, 72
getPlaceholderPosition, 72
isInitialStageClicked, 72
isPlaceholderClicked, 73
movePlaceholder, 73
operator=, 74
resetQubit, 74
switchInitialState, 74
VisualQubit, 69

windowHeight
main.cpp, 94
windowWidth
main.cpp, 94
writeCircuitToFile
File-io.hpp, 89

X
RotationGate.hpp, 86

Y
RotationGate.hpp, 86

Z
RotationGate.hpp, 86