# 435L Project Group 5

Generated by Doxygen 1.8.13

# Contents

# Chapter 1

# Hierarchical Index

## 1.1 Class Hierarchy

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

# Chapter 2

# Class Index

## 2.1 Class List

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

# Chapter 3

# File Index

## 3.1 File List

Here is a list of all documented files with brief descriptions:

# Chapter 4

# Class Documentation

## 4.1 account Class Reference

contains account class definition

`#include <account.h>`

### Public Member Functions

- account (QString f, QString l, QString user, QString pass, QString imag)

    *account constructor that takes parameters.*
- account ()

    *account constructor that does not takes parameters.*
- void print ()

    *prints account details.*

### Public Attributes

- QString firstName

    *Stores the first name of the user.*
- QString lastName

    *Stores the last name of the user.*
- QString gender

    *Stores the gender of the user.*
- QString username

    *Stores the username of the user.*
- QString password

    *Stores the password of the user.*
- QString day

    *Stores the day of birth of the user.*
- QString month

    *Stores the month of birth of the user.*
- QString year

    *Stores the year of birth of the user.*
- QString imageloc

    *Stores the path to the user's profile picture.*

### 4.1.1 Detailed Description

contains account class definition

This class is responsible for storing user information.

### 4.1.2 Constructor & Destructor Documentation

#### 4.1.2.1 account() [1/2]

```
account::account (
            QString f,
            QString l,
            QString user,
            QString pass,
            QString imag )
```

account constructor that takes parameters.

account constructor that takes first name, last name, username, password and image path as parameters.

#### 4.1.2.2 account() [2/2]

```
account::account ( )
```

account constructor that does not takes parameters.

account constructor that doesn't take any parameters.

### 4.1.3 Member Function Documentation

#### 4.1.3.1 print()

```
void account::print ( )
```

prints account details.

prints account's first name, last name, username, password, gender, date of birth and image location.

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

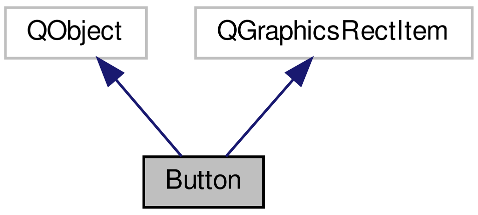
- account.h
- account.cpp

## 4.2 Button Class Reference

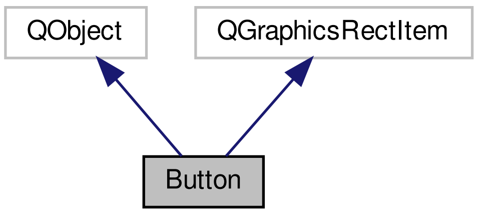
contains button class definition

`#include <button.h>`

Inheritance diagram for Button:



Collaboration diagram for Button:



### Signals

- void clicked ()

  *signal to be emitted.*

### Public Member Functions

- Button (QString name, QGraphicsItem ∗parent=NULL)

  *Button constructor.*
- void mousePressEvent (QGraphicsSceneMouseEvent ∗event)

  *mouse press event function.*
- void hoverEnterEvent (QGraphicsSceneHoverEvent ∗event)

  *mouse hover event function.*
- void hoverLeaveEvent (QGraphicsSceneHoverEvent ∗event)

  *mouse leave event function.*

### 4.2.1 Detailed Description

contains button class definition

This class is responsible for creating a button inside a game (QT scene)

### 4.2.2 Constructor & Destructor Documentation

#### 4.2.2.1 Button()

```
Button::Button (
            QString name,
            QGraphicsItem * parent = NULL )
```

Button constructor.

Button constructor that takes button name as parameter.

### 4.2.3 Member Function Documentation

#### 4.2.3.1 clicked

```
void Button::clicked ( )  [signal]
```

signal to be emitted.

signal that is emitted when the button is pressed.

#### 4.2.3.2 hoverEnterEvent()

```
void Button::hoverEnterEvent (
            QGraphicsSceneHoverEvent * event )
```

mouse hover event function.

function that lightens up the button when the mouse is hovering over it

#### 4.2.3.3 hoverLeaveEvent()

```
void Button::hoverLeaveEvent (
            QGraphicsSceneHoverEvent * event )
```

mouse leave event function.

function that dims back the button when the mouse is no longer hovering over it

#### 4.2.3.4 mousePressEvent()

```
void Button::mousePressEvent (
             QGraphicsSceneMouseEvent * event )
```

mouse press event function.

function that sends a signal whenever the button is pressed

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

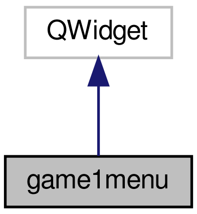
- button.h
- button.cpp

## 4.3 game1menu Class Reference

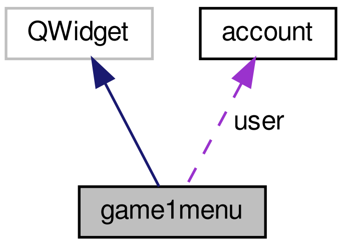
contains game 1 menu class definition

`#include <game1menu.h>`

Inheritance diagram for game1menu:



Collaboration diagram for game1menu:



## Public Slots

- void playb ()

  *function to be called when play is pressed.*
- void Backk ()

  *function to be called when back is pressed.*

## Public Member Functions

- game1menu (QWidget ∗parent=nullptr)

  *game 1 menu constructor.*

## Public Attributes

- QLabel ∗ title

  *Kill covid 19 image.*
- QPushButton ∗ play

  *Play button.*
- QPushButton ∗ back

  *Back button.*
- QLabel ∗ l

  *label asking to select game level*
- QRadioButton ∗ lvl1

  *level 1 radio button*
- QRadioButton ∗ lvl2

  *level 2 radio button*
- QRadioButton ∗ lvl3

  *level 3 radio button*
- QVBoxLayout ∗ Vbox

  *Vertical box used for layout.*
- account ∗ user

  *passed down user from sign in widget*

### 4.3.1 Detailed Description

contains game 1 menu class definition

This class is responsible for creating the main menu of game 1

### 4.3.2 Constructor & Destructor Documentation

#### 4.3.2.1 game1menu()

```
game1menu::game1menu (
            QWidget * parent = nullptr )  [explicit]
```

game 1 menu constructor.

constructs the class and sets up the layout. also links signals to slots.

### 4.3.3 Member Function Documentation

#### 4.3.3.1 Backk

```
void game1menu::Backk ( )  [slot]
```

function to be called when back is pressed.

exits the game menu and goes back to sign in widget

#### 4.3.3.2 playb

```
void game1menu::playb ( )  [slot]
```

function to be called when play is pressed.

checks which level is selected and goes to game 1 scene. each level has it's own difficulty and music audio

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

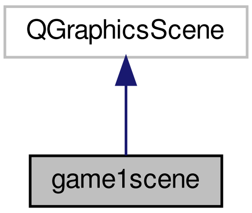
- game1menu.h
- game1menu.cpp

## 4.4 game1scene Class Reference

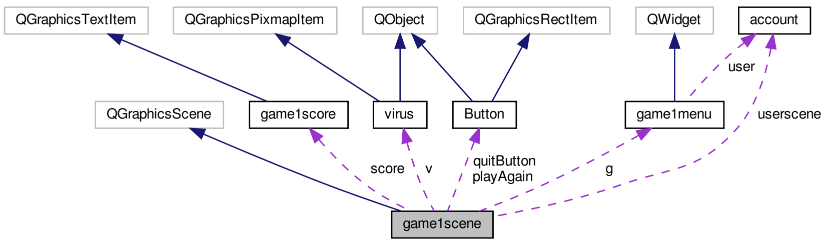
contains game 1 scene class definition

```
#include <game1scene.h>
```

Inheritance diagram for game1scene:



Collaboration diagram for game1scene:



### Public Slots

- void create_instance ()

  *creates a virus.*
- void update_counters ()

  *updates the current state of the game.*
- void restartGame ()

  *Function that restarts the game.*
- void quitGame ()

  *Function that quits the game.*

## Public Member Functions

- game1scene ()

  *game 1 scene constructor.*
- void lose ()

  *lose function to call when you lose.*
- void win ()

  *win function to call when you win.*
- void drawPanel (int x, int y, int width, int height, QColor color, double opacity)

  *function that draws a panel.*

## Public Attributes

- int i

  *itterator that keeps track of viruses*
- account ∗ userscene

  *user account passed down from game 1 menu*
- int speed

  *speed at which the game is running*
- QGraphicsView ∗ view

  *view used to create the scene*
- QMediaPlayer ∗ audio

  *music audio that is played during the game*
- int lossCount

  *stores the number of missed viruses required to lose the game*
- int scoreToWin

  *stores the score needed to win the game*
- int loss

  *keeps count of how many viruses were missed*
- game1menu ∗ g

  *pointer to game 1 menu to be able to exit the game*
- QFile ∗ file

  *files used to read level (the txt databasse)*
- QTextStream ∗ in

  *Text stream used to read file.*
- game1score ∗ score

  *score class that keeps track of the score and displays it on the scene*
- QGraphicsTextItem ∗ io

  *text item to display either a win or a loss*
- Button ∗ playAgain

  *button used for Play again when the game is over*
- Button ∗ quitButton

  *button used to quit the game when the game is over*
- QTimer ∗ inst

  *Timer used to call create instancce.*
- QTimer ∗ count

  *Timer used to call update counter.*
- QGraphicsRectItem ∗ panel

  *panel used to darken the screen for endgame*
- virus ∗ v [40]

  *array of viruses to be created*

### 4.4.1 Detailed Description

contains game 1 scene class definition

This class is responsible for creating and playing game 1 (Kill Covid-19)

### 4.4.2 Constructor & Destructor Documentation

#### 4.4.2.1 game1scene()

```
game1scene::game1scene ( )
```

game 1 scene constructor.

constructs game 1 scene and sets background image and initialized all elements inlucding timers and linking timers to slots.

### 4.4.3 Member Function Documentation

#### 4.4.3.1 create_instance

```
void game1scene::create_instance ( )  [slot]
```

creates a virus.

function that creates a virus called by the timer. It checks the game level to see the size and place at which the virus should be spawned.

#### 4.4.3.2 drawPanel()

```
void game1scene::drawPanel (
             int x,
             int y,
             int width,
             int height,
             QColor color,
             double opacity )
```

function that draws a panel.

function that draws pannels used to create a dark panel for the end game.

#### 4.4.3.3 lose()

```
void game1scene::lose ( )
```

lose function to call when you lose.

ends the game with a loss screen and plays a loss audio. Also writes to the account's history the results of the game.

#### 4.4.3.4 quitGame

```
void game1scene::quitGame ( ) [slot]
```

Function that quits the game.

Function that quits the game when the Quit game button is pressed. goes back to game 1 menu

#### 4.4.3.5 restartGame

```
void game1scene::restartGame ( ) [slot]
```

Function that restarts the game.

Function that restart the game when the Restart button is pressed.

#### 4.4.3.6 update_counters

```
void game1scene::update_counters ( ) [slot]
```

updates the current state of the game.

updates the current state of the game by updating the score, counts and checks for win/loss conditions.

#### 4.4.3.7 win()

```
void game1scene::win ( )
```

win function to call when you win.

ends the game with a win screen and plays a victory sound. Also writes to the account's history the results of the game.

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

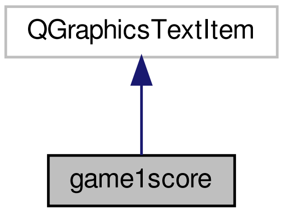
- game1scene.h
- game1scene.cpp

## 4.5 game1score Class Reference

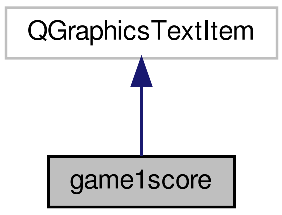
contains game 1 score class definition

```
#include <game1score.h>
```

Inheritance diagram for game1score:



Collaboration diagram for game1score:



### Public Member Functions

- game1score (QGraphicsItem ∗parent=nullptr)

    *game 1 score constructor.*
- void si ()

    *updates score when small virus is killed.*
- void mi ()

    *updates score when medium virus is killed.*
- void bi ()

    *updates score when big virus is killed.*

## Public Attributes

- int scount

  *stores small count*
- int mcount

  *stores medium count*
- int bcount

  *stores big count*
- int score

  *stores total game score*
- int sum

  *stores the sum of scount, mcount and bcount*
- QString text

  *text to be displayed on screen as score*

### 4.5.1 Detailed Description

contains game 1 score class definition

This class is responsible for keeping track of the score and displaying it in game 1

### 4.5.2 Constructor & Destructor Documentation

#### 4.5.2.1 game1score()

```
game1score::game1score (
            QGraphicsItem * parent = nullptr )  [explicit]
```

game 1 score constructor.

constructs the class and sets all scores to 0. additionally sets the base text to be displayed in game 1 scene.

### 4.5.3 Member Function Documentation

#### 4.5.3.1 bi()

```
void game1score::bi ( )
```

updates score when big virus is killed.

updates score when big virus is killed by incrementing big count by 1 and score by 7.

#### 4.5.3.2 mi()

```
void game1score::mi ( )
```

updates score when medium virus is killed.

updates score when medium virus is killed by incrementing medium count by 1 and score by 5.

#### 4.5.3.3 si()

```
void game1score::si ( )
```

updates score when small virus is killed.

updates score when small virus is killed by incrementing small count by 1 and score by 3.

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

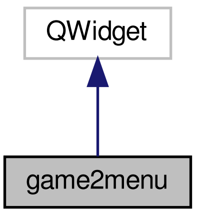
- game1score.h
- game1score.cpp

## 4.6 game2menu Class Reference

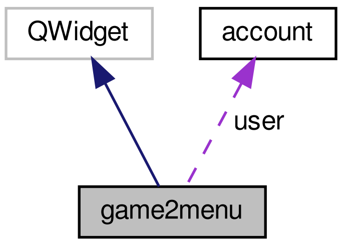
contains game 2 menu class definition

`#include <game2menu.h>`

Inheritance diagram for game2menu:



Collaboration diagram for game2menu:



## Public Slots

- void playb ()

  *function to be called when play is pressed.*
- void Backk ()

  *function to be called when back is pressed.*

## Public Member Functions

- game2menu (QWidget ∗parent=nullptr)

  *game 1 menu constructor.*

## Public Attributes

- QLabel ∗ title

  *Othello title image.*
- QPushButton ∗ play

  *Play button.*
- QPushButton ∗ back

  *Back button.*
- QVBoxLayout ∗ Vbox

  *Vertical box used for layout.*
- account ∗ user

  *passed down user from sign in widget*

### 4.6.1 Detailed Description

contains game 2 menu class definition

This class is responsible for creating the main menu of game 2

### 4.6.2 Constructor & Destructor Documentation

#### 4.6.2.1 game2menu()

```
game2menu::game2menu (
            QWidget * parent = nullptr )  [explicit]
```

game 1 menu constructor.

constructs the class and sets up the layout. also links signals to slots.

### 4.6.3 Member Function Documentation

#### 4.6.3.1 Backk

```
void game2menu::Backk ( )  [slot]
```

function to be called when back is pressed.

exits the game menu and goes back to sign in widget

#### 4.6.3.2 playb

```
void game2menu::playb ( )  [slot]
```

function to be called when play is pressed.

checks goes to game 2 scene.

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

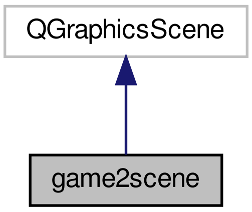
- game2menu.h
- game2menu.cpp

## 4.7 game2scene Class Reference

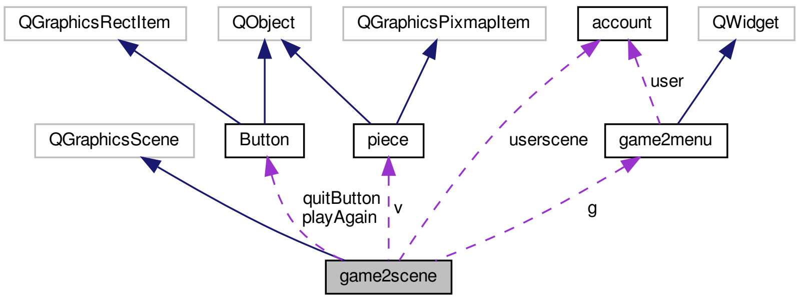
contains game 2 scene class definition

```
#include <game2scene.h>
```

Inheritance diagram for game2scene:



Collaboration diagram for game2scene:



### Public Slots

- void checkFresh ()

  *Function that is called when a signal is emitted from any piece.*
- void restartGame ()

  *Function that restarts the game.*
- void quitGame ()

  *Function that quits the game.*

### Public Member Functions

- game2scene ()

  *game 2 scene constructor.*
- void drawPanel (int x, int y, int width, int height, QColor color, double opacity)

  *function that draws a panel.*
- void outflank (int i, int j)

  *Outflank function.*
- bool checklegal ()

  *function that checks which pieces should be valid.*
- void end ()

  *end game function.*

### Public Attributes

- account ∗ userscene

  *user account passed down from game 1 menu*
- QGraphicsView ∗ view

  *view used to create the scene*
- piece ∗ v [8][8]

  *Double array containing all 64 pieces of the game.*
- QMediaPlayer ∗ au

  *adio player used to play different sounds*
- QGraphicsRectItem ∗ panel

  *panel used to darken the screen for endgame*
- QGraphicsTextItem ∗ io

  *text item to display who won and the score*
- game2menu ∗ g

  *pointer to game 2 menu to be able to exit the game*
- Button ∗ playAgain

  *button used for Play again when the game is over*
- Button ∗ quitButton

  *button used to quit the game when the game is over*

## 4.7.1 Detailed Description

contains game 2 scene class definition

This class is responsible for creating and playing game 2 (othello or reversi)

### 4.7.2 Constructor & Destructor Documentation

#### 4.7.2.1 game2scene()

```
game2scene::game2scene ( )
```

game 2 scene constructor.

constructs game 2 scene and sets background image and initialized all elements inlucding all 64 pieces links their signals to the correct slots.

### 4.7.3 Member Function Documentation

#### 4.7.3.1 checkFresh

```
void game2scene::checkFresh ( )  [slot]
```

Function that is called when a signal is emitted from any piece.

This function is called whenever a signal is emitted from any piece, It searches for the fresh or newest piece placed on the board and calls the function outflank.

#### 4.7.3.2 checklegal()

```
bool game2scene::checklegal ( )
```

function that checks which pieces should be valid.

this boolean function checks all pieces on the board from all directions to see if they are outflankable. if a piece is outflankable this piece is set to valid. this function returns false if no valid piece is found and thus does not switch turns else it returns true.

#### 4.7.3.3 drawPanel()

```
void game2scene::drawPanel (
             int x,
             int y,
             int width,
             int height,
             QColor color,
             double opacity )
```

function that draws a panel.

function that draws pannels used to create a dark panel for the end game.

#### 4.7.3.4 end()

```
void game2scene::end ( )
```

end game function.

when the game is over, this function checks who won, display the end game scene on the view and plays a victory sound.

#### 4.7.3.5 outflank()

```
void game2scene::outflank (
             int i,
             int j )
```

Outflank function.

this function takes the location of a piece and checks all 8 directions around it if it can outflank the enemy pieces and outflanks them. At the end, the function checklegal is called and the turn is changed.

#### 4.7.3.6 quitGame

```
void game2scene::quitGame ( )  [slot]
```

Function that quits the game.

Function that quits the game when the Quit game button is pressed. goes back to game 2 menu

#### 4.7.3.7 restartGame

```
void game2scene::restartGame ( )  [slot]
```

Function that restarts the game.

Function that restart the game when the Restart button is pressed.

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

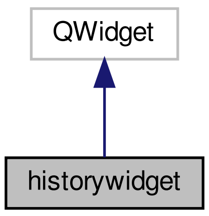
- game2scene.h
- game2scene.cpp

## 4.8 historywidget Class Reference

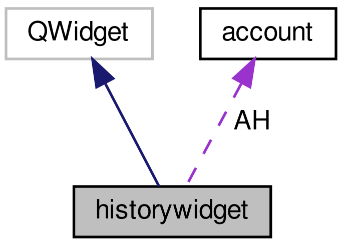
contains history widget class defintion

```
#include <historywidget.h>
```

Inheritance diagram for historywidget:



Collaboration diagram for historywidget:



### Public Slots

- void Back1 ()

    *back slot*

### Public Member Functions

- historywidget (QWidget ∗parent=nullptr)

    *history widget constructor*

- void getHistory ()

    *Get history function.*

### Public Attributes

- account ∗ AH

  *account*
- QPushButton ∗ back

  *back button*
- QVBoxLayout ∗ VBox

  *vertical box layout*

## 4.8.1 Detailed Description

contains history widget class defintion

This class is responsible for constructing the history widget

## 4.8.2 Constructor & Destructor Documentation

#### 4.8.2.1 historywidget()

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

history widget constructor

responsible for setting up the gui for history widget and linking the button to its slot

## 4.8.3 Member Function Documentation

#### 4.8.3.1 Back1

```
void historywidget::Back1 ( )   [slot]
```

back slot

if user is guest goes back to sign in widget as guest but if user has a username goes back to sign in widget

#### 4.8.3.2 getHistory()

`void historywidget::getHistory ( )`

Get history function.

gets the history of the user by checking the database text file of that user under /userhistory path and displays that information

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

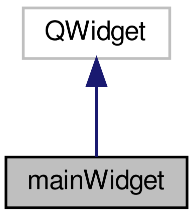
- historywidget.h
- historywidget.cpp

## 4.9 mainWidget Class Reference

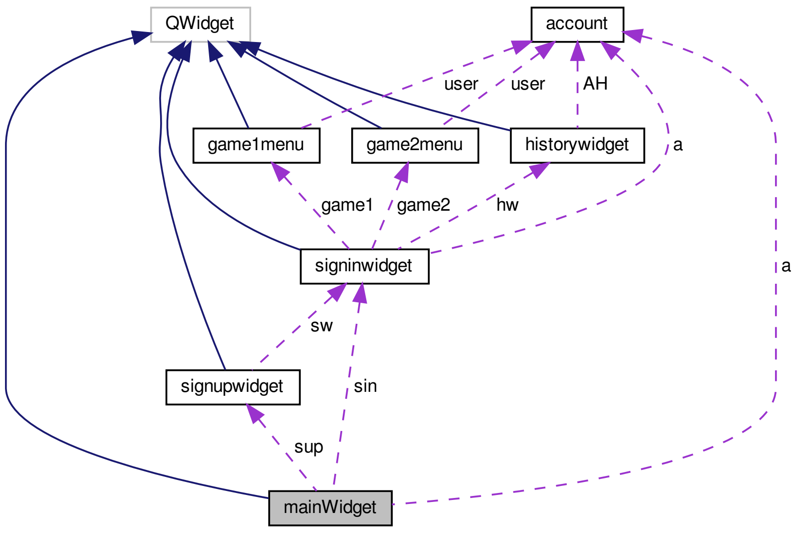
contains main widget class defintion

`#include <mainwidget.h>`

Inheritance diagram for mainWidget:



Collaboration diagram for mainWidget:



### Public Slots

- void signin ()

  *signin slot*
- void signup ()

  *sign up slot*
- void playAsGuest ()

  *Play as guest slot.*

### Public Member Functions

- mainWidget (QWidget ∗parent=nullptr)

  *main widget constructor*

### Public Attributes

- QLabel ∗ label0

  *user name label*
- QLabel ∗ label1

  *password label*
- QLineEdit ∗ line0

  *user name line*
- QLineEdit ∗ line1

  *password line*
- QPushButton ∗ PB0

  *sign in button*
- QPushButton ∗ PB1

  *sign up button*
- QPushButton ∗ PB2

  *play as guest button*
- QVBoxLayout ∗ VBox

  *vertical box layout*
- signupwidget ∗ sup

  *sign up widget*
- account ∗ a

  *account*
- signinwidget ∗ sin

  *sign in widget*
- QMessageBox ∗ messageBox

  *messagebox to display errors*

### 4.9.1 Detailed Description

contains main widget class defintion

This class is responsible for constructing the main widget

## 4.9.2 Constructor & Destructor Documentation

### 4.9.2.1 mainWidget()

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

main widget constructor

responsible for building the gui of main widget and linking the buttons to their respective slots

## 4.9.3 Member Function Documentation

### 4.9.3.1 playAsGuest

```
void mainWidget::playAsGuest ( )  [slot]
```

Play as guest slot.

opens sign in widget as a guest and closes main widget and adds guest to the list of users if it is not there.

### 4.9.3.2 signin

```
void mainWidget::signin ( )  [slot]
```

signin slot

responsible for checking if the user is in the database and if he typed the password correctly if so opens sign in widget

### 4.9.3.3 signup

```
void mainWidget::signup ( )  [slot]
```

sign up slot

open sign up widget and closes main widget

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

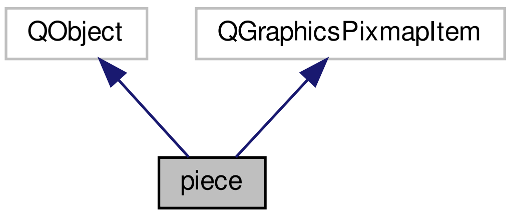
- mainwidget.h
- mainwidget.cpp

## 4.10 piece Class Reference

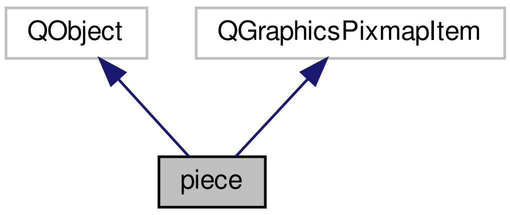
contains piece class definition

`#include <piece.h>`

Inheritance diagram for piece:



Collaboration diagram for piece:



### Public Types

- enum **state** { **invalid**, **valid**, **white**, **black** }

### Public Slots

- void place ()

  *places a piece.*

### Signals

- void placed ()

  *signal to be emitted.*

## Public Member Functions

- piece (QObject ∗parent=nullptr)

  *piece constructor.*
- void setState (state s)

  *sets the state of a piece.*

## Public Attributes

- state s

  *indicates the state of a piece*
- bool fresh

  *bool value to store if the piece is the newest piece on the board*
- QMediaPlayer ∗ placep

  *audio to be played when a piece is placed*

### 4.10.1 Detailed Description

contains piece class definition

This class is responsible for placing a piece in game 2 and displaying it on the screen

### 4.10.2 Constructor & Destructor Documentation

#### 4.10.2.1 piece()

```
piece::piece (
              QObject ∗ parent = nullptr )   [explicit]
```

piece constructor.

constructor was only used to set the media player and timers.

### 4.10.3 Member Function Documentation

#### 4.10.3.1 place

```
void piece::place ( )  [slot]
```

places a piece.

when a valid piece is selected it's state is replaced by eithe white or black depending on the turn. In addition to that, the place audio is played and a signal is emitted to game2scene.

#### 4.10.3.2 placed

```
void piece::placed ( )  [signal]
```

signal to be emitted.

signal to be emitted to game2scene to indicate that a piece is placed.

#### 4.10.3.3 setState()

```
void piece::setState (
             state s )
```

sets the state of a piece.

sets the state of a piece and displays the correct image (black circle, white circle, green box, yellow box).

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

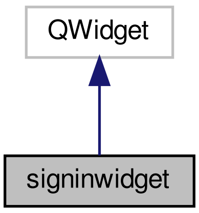
- piece.h
- piece.cpp

## 4.11 signinwidget Class Reference

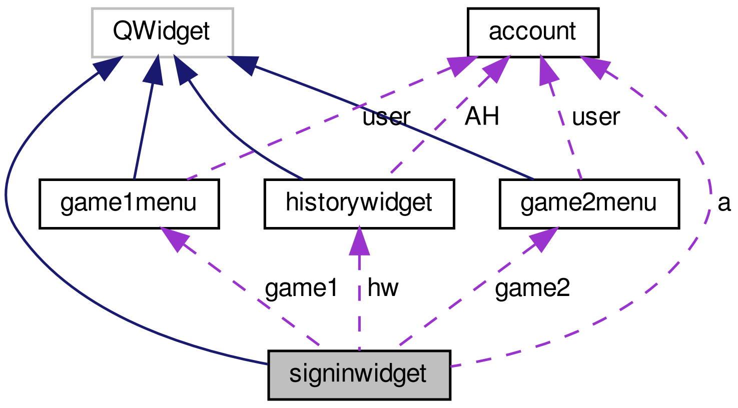
contains signin widget class defintion

```
#include <signinwidget.h>
```

Inheritance diagram for signinwidget:



Collaboration diagram for signinwidget:



## Public Slots

- void History ()

  *History button slot.*
- void playGame1 ()

  *Play game1 button.*
- void playGame2 ()

  *Play game 2 button.*
- void quit ()

  *quit button*

## Public Member Functions

- signinwidget (QWidget ∗parent=nullptr)

  *signin widget constructor*
- void getName ()

  *get name function*
- void checkBirthday ()

  *Checking birthday function.*

## Public Attributes

- account ∗ a

  *account of user who signed in*
- game1menu ∗ game1

  *game1 menu*
- game2menu ∗ game2

  *game2 menu*
- historywidget ∗ hw

*the history widget*

- QLabel ∗ pic

  *picture label for profile picture*
- QLabel ∗ name

  *name label for the user's name*
- QMessageBox ∗ messageBox

  *message box to display birthday congratulations*
- QPushButton ∗ history

  *history button*
- QPushButton ∗ play1

  *game 1 button*
- QPushButton ∗ play2

  *game 2 button*
- QPushButton ∗ logout

  *logout button*
- QVBoxLayout ∗ VBox

  *vertical box layout*
- QImage ∗ image

  *image for profile pic*

### 4.11.1 Detailed Description

contains signin widget class defintion

This class is responsible for constructing the signin widget

### 4.11.2 Constructor & Destructor Documentation

#### 4.11.2.1 signinwidget()

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

signin widget constructor

responsible for building the gui of signin widget and linking the buttons to their respective slots

### 4.11.3 Member Function Documentation

#### 4.11.3.1 checkBirthday()

```
void signinwidget::checkBirthday ( )
```

Checking birthday function.

responsible to check if it is the user's birthday and if so display congratulations message box

#### 4.11.3.2 getName()

```
void signinwidget::getName ( )
```

get name function

responsible for getting accounts information and displaying what is relevant on the widget

#### 4.11.3.3 History

```
void signinwidget::History ( ) [slot]
```

History button slot.

responsible for checking the users history and displaying the history widget

#### 4.11.3.4 playGame1

```
void signinwidget::playGame1 ( ) [slot]
```

Play game1 button.

opens game 1 menu for the user to play game 1

#### 4.11.3.5 playGame2

```
void signinwidget::playGame2 ( ) [slot]
```

Play game 2 button.

opens game 2 menu for the user to play game 2

#### 4.11.3.6 quit

```
void signinwidget::quit ( ) [slot]
```

quit button

goes back to main widget and closes sign in widget

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

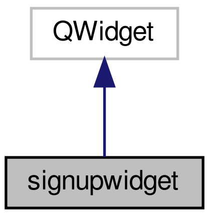
- signinwidget.h
- signinwidget.cpp

## 4.12 signupwidget Class Reference

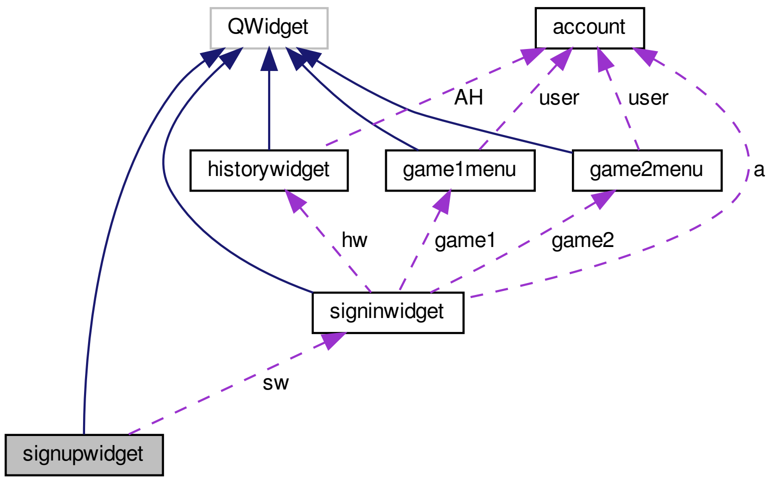
contains signup widget class defintion

`#include <signupwidget.h>`

Inheritance diagram for signupwidget:



Collaboration diagram for signupwidget:



### Public Slots

- void signup ()

  *sign up button slot*
- void image ()

  *image function*
- void quit ()

  *quit signup*

## Public Member Functions

- signupwidget (QWidget ∗parent=nullptr)

    *class constructor*

## Public Attributes

- QLabel ∗ firstname_label

    *first name*
- QLabel ∗ lastname_label

    *last name*
- QLabel ∗ username_label

    *username*
- QLabel ∗ password_label

    *password*
- QLabel ∗ confirmpassword_label

    *confirm password*
- QLabel ∗ picture_label

    *picture*
- QLabel ∗ gender_label

    *gender*
- QLabel ∗ DateOfBirth_label

    *date of birth*
- QLineEdit ∗ firstname_line

    *first name*
- QLineEdit ∗ lastname_line

    *last name*
- QLineEdit ∗ username_line

    *username*
- QLineEdit ∗ password_line

    *password*
- QLineEdit ∗ comfirmpassword_line

    *confirm password*
- QRadioButton ∗ RB0

    *male RB*
- QRadioButton ∗ RB1

    *female RB*
- QCalendarWidget ∗ C

    *calender widget*
- QPushButton ∗ PB0

    *sign up*
- QPushButton ∗ PB1

    *upload pic button*
- QPushButton ∗ back

    *back button*
- QVBoxLayout ∗ VBox

    *vertical box layout*
- QHBoxLayout ∗ HBox

    *horizontal box layout*
- QGroupBox ∗ G

*group box*
- QMessageBox ∗ messageBox

  *message box displaying errors*
- QRegExp ∗ password_RegEx

  *regular expression used to define the password limits*
- QString imageName =""

  *filepath as a string for image of user initialized to null*
- signinwidget ∗ sw

  *sign in widget to take the user to sign in directly after sign up*

### 4.12.1 Detailed Description

contains signup widget class defintion

This class is responsible for constructing the sign up widget

### 4.12.2 Constructor & Destructor Documentation

#### 4.12.2.1 signupwidget()

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

class constructor

responsible for constructing the gui and linking the buttons to their respective slot functions

### 4.12.3 Member Function Documentation

#### 4.12.3.1 image

```
void signupwidget::image ( )  [slot]
```

image function

saves the image to our build directory to have it on next launch

#### 4.12.3.2 quit

```
void signupwidget::quit ( )  [slot]
```

quit signup

takes the user back from sign up widget to main widget

#### 4.12.3.3 signup

```
void signupwidget::signup ( )  [slot]
```

sign up button slot

responsible for checking if all information is valid and saving the user in the database and signing them in

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

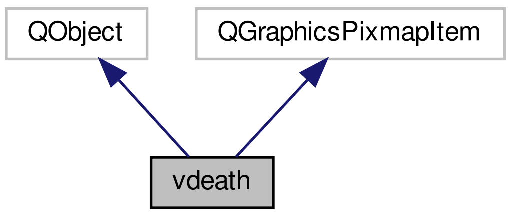
- signupwidget.h
- signupwidget.cpp

## 4.13 vdeath Class Reference

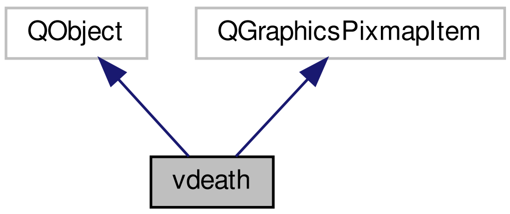
contains virus death class definition

```
#include <vdeath.h>
```

Inheritance diagram for vdeath:



Collaboration diagram for vdeath:



### Public Slots

- void die ()
  *Remove death image from scene.*

### Public Member Functions

- vdeath (QObject ∗parent=nullptr)
  *sets the death image and calls slot die to remove it*

### Public Attributes

- QTimer ∗ timer
  *timer to remove the death image*

## 4.13.1 Detailed Description

contains virus death class definition

This class is responsible for setting the death animation and removing it from the scene

## 4.13.2 Constructor & Destructor Documentation

### 4.13.2.1 vdeath()

```
vdeath::vdeath (
             QObject ∗ parent = nullptr )  [explicit]
```

sets the death image and calls slot die to remove it

constructor to set the death image and call slot die to remove it from the scene

## 4.13.3 Member Function Documentation

### 4.13.3.1 die

```
void vdeath::die ( )  [slot]
```

Remove death image from scene.

clearing up the scene from the dead viruses

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

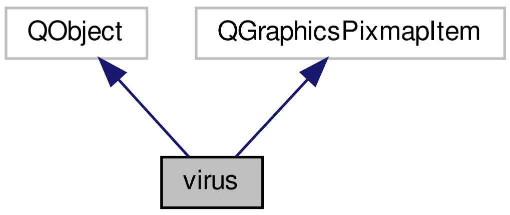
- vdeath.h
- vdeath.cpp

## 4.14 virus Class Reference

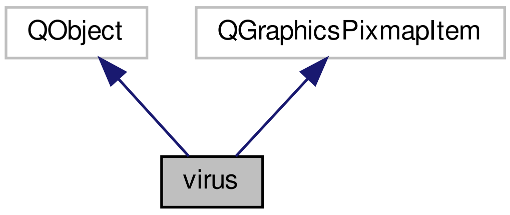
contains virus class definition

```
#include <virus.h>
```

Inheritance diagram for virus:



Collaboration diagram for virus:



### Public Types

- enum **sizz** { **small**, **medium**, **big** }

### Public Slots

- void update ()

  *Updates position of the virus.*
- void kill ()

  *Check if virus is selected then kill it.*

## Public Member Functions

- virus (QObject ∗parent=nullptr)

    *sets the death sound mediaplayer*
- void setSize (sizz s)

    *sets the size and image to viruses*

## Public Attributes

- sizz size

    *instance of enum type*
- bool alive

    *boolean to verify if virus is alive*
- bool isChecked

    *used to keep track for score*
- QMediaPlayer ∗ deathsound

    *mediaplayer to play death sound*
- int speed

    *speed of viruses falling*
- QTimer ∗ timer

    *first timer responsible for calling update position of virus*
- QTimer ∗ timer2

    *second timer responsible for calling kill to check for killed viruses*

### 4.14.1 Detailed Description

contains virus class definition

This class is responsible for creating the viruses for the game along with destroying them on click

### 4.14.2 Constructor & Destructor Documentation

#### 4.14.2.1 virus()

```
virus::virus (
             QObject * parent = nullptr )  [explicit]
```

sets the death sound mediaplayer

constructor was only used to set the media player other functionalities were delegated to setSize()

### 4.14.3 Member Function Documentation

#### 4.14.3.1 kill

```
void virus::kill ( )  [slot]
```

Check if virus is selected then kill it.

checks if a virus was selected then it kills it and plays the deathsound and displays death animation

#### 4.14.3.2 setSize()

```
void virus::setSize (
             sizz s )
```

sets the size and image to viruses

sets the size of the virus along with a respective image and connects the timers to their respective slots. to be called everytime you need to create a virus.

#### 4.14.3.3 update

```
void virus::update ( )  [slot]
```

Updates position of the virus.

make the virus fall down with varrying speeds.

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

- virus.h
- virus.cpp

# Chapter 5

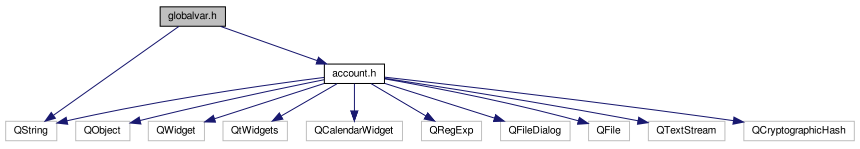
# File Documentation

## 5.1 globalvar.h File Reference

contains global variable defintions

```
#include <QString>
#include "account.h"
```

Include dependency graph for globalvar.h:



### Variables

- QString textf

  *string to store textfile path we are using for level choice*
- QString aud

  *string to store audio file path we are using for level choice*
- int speedowagon

  *speed for varrying speed for different levels*
- account ∗ guest

  *guest account across the whole project*
- bool turnmasta

  *boolean to determine turn in game 2*

### 5.1.1 Detailed Description

contains global variable defintions

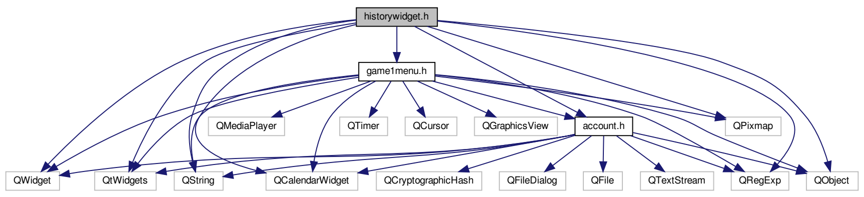
This file defines global variables used in the project

## 5.2 historywidget.h File Reference

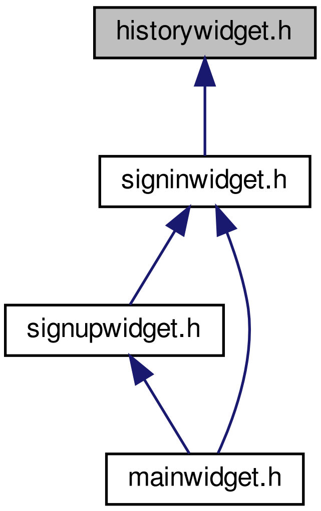
contains history widget class defintion

```
#include <QObject>
#include <QWidget>
#include <QtWidgets>
#include <QString>
#include <QCalendarWidget>
#include <QRegExp>
#include "account.h"
#include "game1menu.h"
#include <QPixmap>
```

Include dependency graph for historywidget.h:



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



### Classes

- class historywidget

  *contains history widget class defintion*

### 5.2.1 Detailed Description

contains history widget class defintion

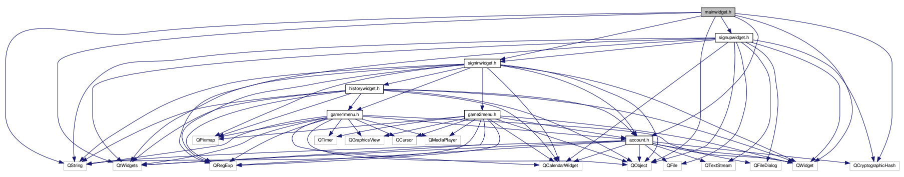
This class is responsible for constructing the history widget

## 5.3 mainwidget.h File Reference

contains main widget class defintion

```
#include <QObject>
#include <QWidget>
#include <QtWidgets>
#include <QString>
#include "signupwidget.h"
#include "signinwidget.h"
#include "account.h"
#include <QCryptographicHash>
```

Include dependency graph for mainwidget.h:



### Classes

- class mainWidget

    *contains main widget class defintion*

### 5.3.1 Detailed Description

contains main widget class defintion

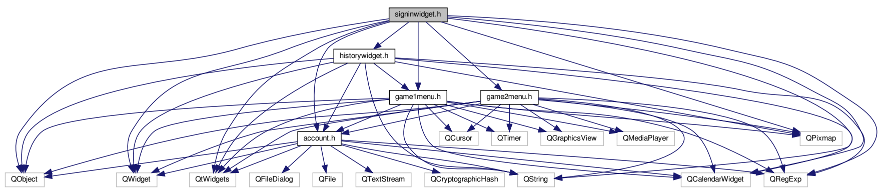
This class is responsible for constructing the main widget

## 5.4 signinwidget.h File Reference

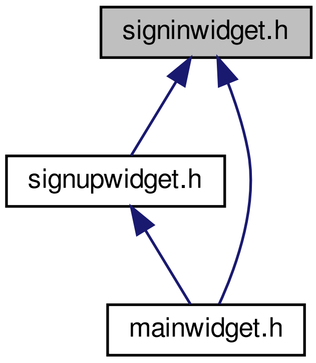
contains signin widget class defintion

```
#include <QObject>
#include <QWidget>
#include <QtWidgets>
#include <QString>
#include <QCalendarWidget>
#include <QRegExp>
#include "account.h"
#include "game1menu.h"
#include "game2menu.h"
#include "historywidget.h"
#include <QPixmap>
```

Include dependency graph for signinwidget.h:



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



### Classes

- class signinwidget

  *contains signin widget class defintion*

### 5.4.1 Detailed Description

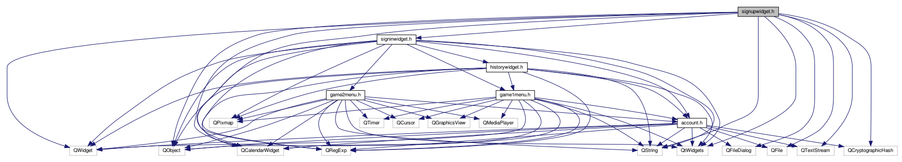
contains signin widget class defintion

This class is responsible for constructing the signin widget

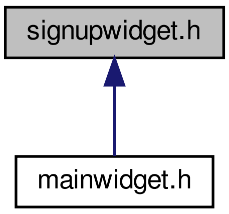
## 5.5 signupwidget.h File Reference

contains signup widget class defintion

```
#include <QObject>
#include <QWidget>
#include <QtWidgets>
#include <QString>
#include <QCalendarWidget>
#include <QRegExp>
#include <QFileDialog>
#include <QFile>
#include <QTextStream>
#include <QCryptographicHash>
#include "signinwidget.h"
```
Include dependency graph for signupwidget.h:



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



### Classes

- class signupwidget

    *contains signup widget class defintion*

### 5.5.1 Detailed Description

contains signup widget class defintion

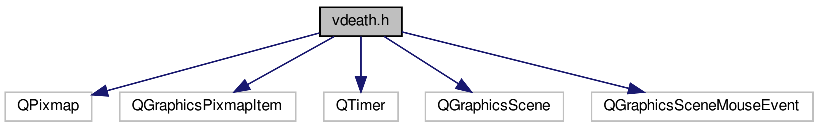
This class is responsible for constructing the sign up widget

## 5.6 vdeath.h File Reference

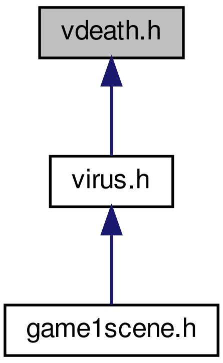
contains virus death class definition

```
#include <QPixmap>
#include <QGraphicsPixmapItem>
#include <QTimer>
#include <QGraphicsScene>
#include <QGraphicsSceneMouseEvent>
```

Include dependency graph for vdeath.h:



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



### Classes

- class vdeath
  *contains virus death class definition*

### 5.6.1 Detailed Description

contains virus death class definition

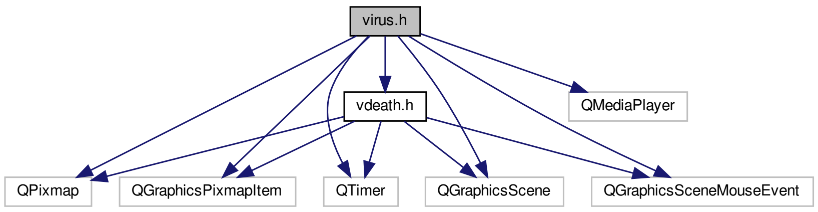
This class is responsible for setting the death animation and removing it from the scene

## 5.7 virus.h File Reference

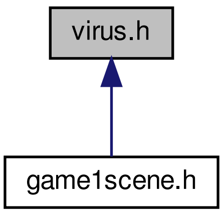
contains virus class definition definition

```
#include <QPixmap>
#include <QGraphicsPixmapItem>
#include <QTimer>
#include <QGraphicsScene>
#include <QGraphicsSceneMouseEvent>
#include <QMediaPlayer>
#include "vdeath.h"
```

Include dependency graph for virus.h:



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



### Classes

- class virus

  *contains virus class definition*

### 5.7.1 Detailed Description

contains virus class definition definition

This class is responsible for creating the viruses for the game along with destroying them on click