

Własna Kontrolka

Generated by Doxygen 1.8.11



# Contents



# Chapter 1

## Własna kontrolka

### Author

Damian Świerk

### Date

15.11.2017

### 1.1 Założenia

Założenia projektowe Cel projektu:

1. przeglądanie zawartości systemu plików,
2. wizualizacja hierarchii plików w postaci drzewa,
3. wyszukiwanie i "katalogowanie" plików konkretnego rodzaju (np. robienie spisu filmów, książek czy muzyki),
4. obliczanie zajętego przez katalog i pliki miejsca,
5. podgląd wybranych rodzajów plików (np. tekstowych),
6. program powinien prezentować zawartość katalogu w postaci graficznej,
7. w przypadku katalogu z podkatalogami, rozmiar podkatalogów powinien być obliczony rekurencyjnie,
8. wybranie konkretnego obszaru przy pomocy myszki powinno dostarczać dodatkowych informacji o pliku,
9. pliki o podobnych formatach (np. pliki tekstowe, muzyczne czy filmy) powinny być podobnie kolorowane.

Wnioski:

1. Nie udało się poprawnie wyświetlić danych zawartych w katalogach, we własnej kontrolce, w sposób wymieniony w założeniach,
2. Wprowadzono samo-skalującą się kontrolkę przedstawiającą w sposób graficzny udział wielkości, zdefiniowanego w konstruktorze klasy DirVisualizer, wektora liczb całkowitych wraz z automatyczną kontrolą odcieni wycinków koła.



## Chapter 2

# Namespace Index

### 2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

[Ui](#) . . . . . ??





## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

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

QMainWindow	
MainWindow . . . . .	??
QObject	
DirAnalyzer . . . . .	??
QWidget	
DirVisualiser . . . . .	??



## Chapter 4

# Class Index

### 4.1 Class List

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

#### DirAnalyzer

Klasa [DirAnalyzer](#) Klasa służy do przetwarzania informacji o danych zamieszczonych w podanej lokalizacji. Pozwala na rekursywne pozyskanie listy plików znajdujących się w katalogu. Posiada funkcjonalność otwierania plików tekstowych . . . . . ??

#### DirVisualiser

Klasa [DirVisualiser](#) Klasa służy do przechowywania właściwości widgetu służącego za graficzne przedstawienie danych w wybranym katalogu . . . . . ??

#### MainWindow

Klasa [MainWindow](#) Klasa Main Window - interfejs graficzny steracza dyskowego. Jest to pomost między operacjami na katalogach i plikach, które wykonywane są w obiekcie m\_analyzer, a graficzną wizualizacją danych w głównym oknie programu . . . . . ??



## Chapter 5

# File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

<a href="#">diranalyzer.cpp</a>	.....	??
<a href="#">diranalyzer.h</a>	.....	??
<a href="#">dirvisualiser.cpp</a>	.....	??
<a href="#">dirvisualiser.h</a>	.....	??
<a href="#">main.cpp</a>	.....	??
<a href="#">mainwindow.cpp</a>	.....	??
<a href="#">mainwindow.h</a>	.....	??



## Chapter 6

# Namespace Documentation

### 6.1 Ui Namespace Reference

#### 6.1.1 Detailed Description

Date

8.11.2017r.

Author

Damian Świerk





## Chapter 7

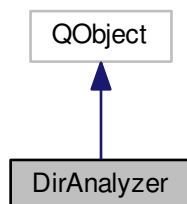
# Class Documentation

### 7.1 DirAnalyzer Class Reference

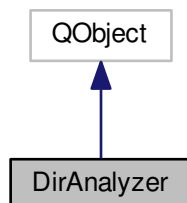
Klasa [DirAnalyzer](#) Klasa służy do przetwarzania informacji o danych zamieszczonych w podanej lokalizacji. Pozwala na rekursywne pozyskanie listy plików znajdujących się w katalogu. Posiada funkcjonalność otwierania plików tekstowych.

```
#include <diranalyzer.h>
```

Inheritance diagram for DirAnalyzer:



Collaboration diagram for DirAnalyzer:



## Public Member Functions

- [DirAnalyzer](#) (QObject \*parent=0)  
*Konstruktor.*
- void [passNewDir](#) (const QModelIndex &index)  
*metoda przygotowująca nowe informacje o danych w podanej lokalizacji*
- bool [isDir](#) ()  
*metoda sprawdzająca czy podana lokalizacja jest folderem*
- bool [isTextFile](#) ()  
*metoda sprawdzająca czy podana lokalizacja jest plikiem tekstowym*
- QString [getTextFileContent](#) ()  
*metoda zwracająca zawartość pliku tekstowego*
- QString [getDirName](#) ()  
*metoda zwracająca nazwę katalogu*
- QStringList [getFileList](#) ()  
*metoda zwracająca zawartość katalogu*
- QString [getFileName](#) ()  
*metoda zwracająca nazwę pliku*
- QString [getPath](#) ()  
*metoda zwracająca podaną ścieżkę*
- QMap< QString, QString > [getDirMap](#) ()  
*metoda zwracająca mapę podkatalogów*
- void [setFilter](#) (QString suffix)  
*metoda ustawiająca filtr plików*

### 7.1.1 Detailed Description

Klasa [DirAnalyzer](#) Klasa służy do przetwarzania informacji o danych zamieszczonych w podanej lokalizacji. Pozwala na rekursywne pozyskanie listy plików znajdujących się w katalogu. Posiada funkcjonalność otwierania plików tekstowych.

### 7.1.2 Constructor & Destructor Documentation

#### 7.1.2.1 DirAnalyzer::DirAnalyzer ( QObject \* parent = 0 ) [explicit]

Konstruktor.

[DirAnalyzer::DirAnalyzer](#) Konstruktor klasy.

#### Parameters

<i>parent</i>	- wskaźnik na widget nadrzędny
---------------	--------------------------------

### 7.1.3 Member Function Documentation

#### 7.1.3.1 QMap< QString, QString > DirAnalyzer::getDirMap ( )

metoda zwracająca mapę podkatalogów

[DirAnalyzer::getDirMap](#) Metoda zwracająca gotową mapę podkatalogów wraz z ich rozmiarami.

#### 7.1.3.2 QString DirAnalyzer::getDirName ( )

metoda zwracająca nazwę katalogu

[DirAnalyzer::getDirName](#) Metoda zwracająca nazwę katalogu.

#### 7.1.3.3 QStringList DirAnalyzer::getFileList ( )

metoda zwracająca zawartość katalogu

[DirAnalyzer::getFileList](#) Metoda zwracająca listę plików w katalogu wraz z ich rozmiarami jako obiekt typu `QStringList`.

#### 7.1.3.4 QString DirAnalyzer::getFileName ( )

metoda zwracająca nazwę pliku

[DirAnalyzer::getFileName](#) Metoda zwracająca nazwę wybranego pliku.

#### 7.1.3.5 QString DirAnalyzer::getPath ( )

metoda zwracająca podaną ścieżkę

[DirAnalyzer::getPath](#) Metoda zwracająca bezwzględny adres do pliku.

#### 7.1.3.6 QString DirAnalyzer::getTextFileContent ( )

metoda zwracająca zawartość pliku tekstowego

[DirAnalyzer::getTextFileContent](#) Metoda zwracająca zawartość pliku tekstowego.

#### 7.1.3.7 bool DirAnalyzer::isDir ( )

metoda sprawdzająca czy podana lokalizacja jest folderem

[DirAnalyzer::isDir](#) Metoda sprawdzająca czy przechowywana ścieżka jest adresem folderu.

#### 7.1.3.8 bool DirAnalyzer::isTextFile ( )

metoda sprawdzająca czy podana lokalizacja jest plikiem tekstowym

[DirAnalyzer::isTextFile](#) Metoda sprawdzająca czy przechowywana ścieżka jest adresem pliku tekstowego.

#### 7.1.3.9 void DirAnalyzer::passNewDir ( const QModelIndex & index )

metoda przygotowująca nowe informacje o danych w podanej lokalizacji

[DirAnalyzer::passNewDir](#) Metoda pobierająca index modelu przedstawiającego wybraną przez użytkownika lokalizację, a następnie rekursywnie przetwarza informacje o danych zawierających się w tej lokalizacji. Algorytm sprawdza czy podana ścieżka przechowuje plik, jeśli tak to sprawdza czy jest on również plikiem tekstowym. Jeżeli ścieżka przechowuje plik tekstowy to jego zawartość zostaje pobrana i zapisana w atrybucie `m_txtContent`. Jeżeli ścieżka wskazuje na folder, to zostaje wczytana zawartość folderu wraz z rozmiarami plików i uwzględnieniem filtru rozszerzenia.

## Parameters

<i>index</i>	- indeks modelu przechowującego adres lokalizacji
--------------	---

7.1.3.10 void DirAnalyzer::setFilter ( QString *suffix* )

metoda ustawiająca filtr plików

[DirAnalyzer::setFilter](#) Metoda ustawiająca nowy filtr rozszerzenia plików.

## Parameters

<i>index</i>	- indeks modelu przechowującego adres lokalizacji
--------------	---

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

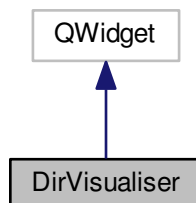
- [diranalyzer.h](#)
- [diranalyzer.cpp](#)

## 7.2 DirVisualiser Class Reference

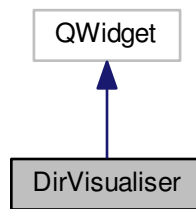
Klasa [DirVisualiser](#) Klasa służy do przechowywania właściwości widgetu służącego za graficzne przedstawienie danych w wybranym katalogu.

```
#include <dirvisualiser.h>
```

Inheritance diagram for DirVisualiser:



Collaboration diagram for DirVisualiser:



### Public Member Functions

- [DirVisualiser](#) (QWidget \*parent=0)  
*Konstruktor.*
- void [setDirMap](#) (const QMap< QString, QString > map)  
*Metoda przypisująca wartość obiektowi m\_map.*

### Protected Member Functions

- void [paintEvent](#) (QPaintEvent \*)  
*Metoda wywoływana w chwili przerysowywania widgetu.*
- void [mouseDoubleClickEvent](#) (QMouseEvent \*e)  
*Metoda wywoływana w chwili podwójnego kliknięcia na widget.*

#### 7.2.1 Detailed Description

Klasa [DirVisualiser](#) Klasa służy do przechowywania właściwości widgetu służącego za graficzne przedstawienie danych w wybranym katalogu.

#### 7.2.2 Constructor & Destructor Documentation

##### 7.2.2.1 DirVisualiser::DirVisualiser ( QWidget \* parent = 0 ) [explicit]

Konstruktor.

[DirVisualiser::DirVisualiser](#) Konstruktor klasy.

Parameters

<i>parent</i>	- wskaźnik na widget nadrzędny
---------------	--------------------------------

### 7.2.3 Member Function Documentation

#### 7.2.3.1 void DirVisualiser::mouseDoubleClickEvent ( QMouseEvent \* *e* ) [protected]

Metoda wywoływana w chwili podwójnego kliknięcia na widget.

[DirVisualiser::mouseDoubleClickEvent](#) Metoda odpowiadająca za wykonanie działań po podwójnym kliknięciu myszy na kontrolce.

##### Parameters

<i>*e</i>	- wskaźnik na wydarzenie QMouseEvent
-----------	--------------------------------------

#### 7.2.3.2 void DirVisualiser::paintEvent ( QPaintEvent \* ) [protected]

Metoda wywoływana w chwili przerysowywania widgetu.

[DirVisualiser::paintEvent](#) Metoda odpowiadająca za przerysowywanie kontrolki.

##### Parameters

*	- wskaźnik na wydarzenie QPaintEvent
---	--------------------------------------

#### 7.2.3.3 void DirVisualiser::setDirMap ( const QMap< QString, QString > *map* )

Metoda przypisująca wartość obiektowi m\_map.

[DirVisualiser::setDirMap](#) Metoda ustawia właściwości atrybutu m\_map.

##### Parameters

<i>map</i>	- mapa wybranego katalogu
------------	---------------------------

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

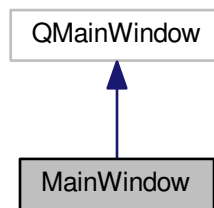
- [dirvisualiser.h](#)
- [dirvisualiser.cpp](#)

## 7.3 MainWindow Class Reference

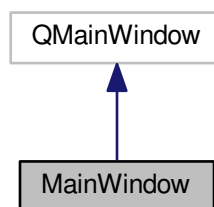
Klasa [MainWindow](#) Klasa Main Window - interfejs graficzny szperacza dyskowego. Jest to pomost między operacjami na katalogach i plikach, które wykonywane są w obiekcie m\_analyzer, a graficzną wizualizacją danych w głównym oknie programu.

```
#include <mainwindow.h>
```

Inheritance diagram for MainWindow:



Collaboration diagram for MainWindow:



## Public Member Functions

- [MainWindow](#) (QWidget \*parent=0)  
*Konstruktor.*
- [~MainWindow](#) ()  
*Destruktor.*

### 7.3.1 Detailed Description

Klasa [MainWindow](#) Klasa Main Window - interfejs graficzny szperacza dyskowego. Jest to pomost między operacjami na katalogach i plikach, które wykonywane są w obiekcie `m_analyzer`, a graficzną wizualizacją danych w głównym oknie programu.

### 7.3.2 Constructor & Destructor Documentation

#### 7.3.2.1 MainWindow::MainWindow ( QWidget \* parent = 0 ) [explicit]

Konstruktor.

[MainWindow::MainWindow](#) Konstruktor klasy. W Konstruktorze tym jest zawarta obsługa slotów i sygnałów programu, a także wprowadzone są ustawienia programu, wraz z deklaracją koniecznych obiektów.

## Parameters

<i>parent</i>	- wskaźnik na widget nadrzędny
---------------	--------------------------------

### 7.3.2.2 `MainWindow::~~MainWindow ( )`

Destruktor.

[MainWindow::~~MainWindow](#) Destruktor klasy.

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

- [mainwindow.h](#)
- [mainwindow.cpp](#)



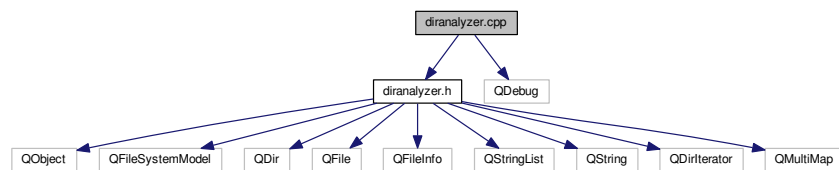
## Chapter 8

# File Documentation

### 8.1 diranalyzer.cpp File Reference

```
#include "diranalyzer.h"  
#include <QDebug>
```

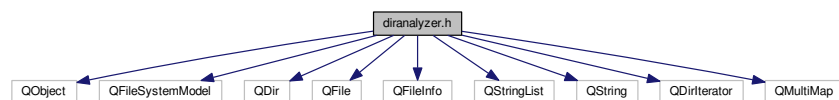
Include dependency graph for diranalyzer.cpp:



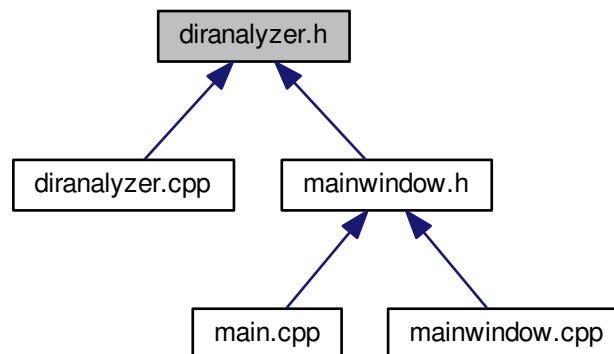
### 8.2 diranalyzer.h File Reference

```
#include <QObject>  
#include <QFileSystemModel>  
#include <QDir>  
#include <QFile>  
#include <QFileInfo>  
#include <QStringList>  
#include <QString>  
#include <QDirIterator>  
#include <QMultiMap>
```

Include dependency graph for diranalyzer.h:



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



## Classes

- class [DirAnalyzer](#)

*Klasa [DirAnalyzer](#) służy do przetwarzania informacji o danych zamieszczonych w podanej lokalizacji. Pozwala na rekursywne pozyskanie listy plików znajdujących się w katalogu. Posiada funkcjonalność otwierania plików tekstowych.*

## Macros

- `#define DIR\_SIZE\_PREFIX "$$$"`

### 8.2.1 Macro Definition Documentation

#### 8.2.1.1 `#define DIR\_SIZE\_PREFIX "$$$"`

#### Date

15.11.2017r.

#### Author

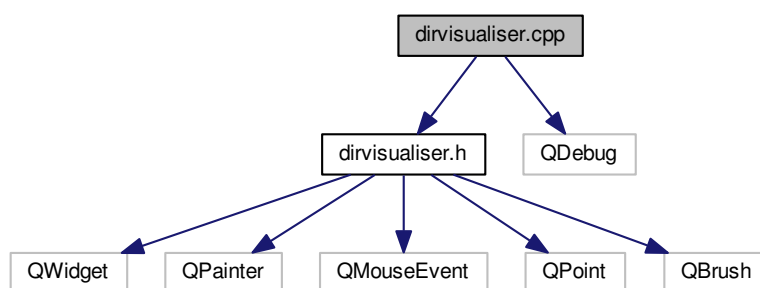
Damian Świerk

## 8.3 dirvisualiser.cpp File Reference

```
#include "dirvisualiser.h"
```

```
#include <QDebug>
```

Include dependency graph for dirvisualiser.cpp:



## 8.4 dirvisualiser.h File Reference

```
#include <QWidget>
```

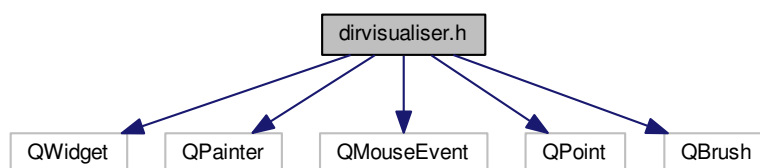
```
#include <QPainter>
```

```
#include <QMouseEvent>
```

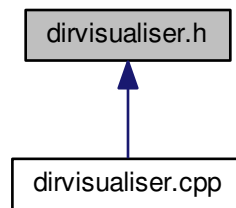
```
#include <QPoint>
```

```
#include <QBrush>
```

Include dependency graph for dirvisualiser.h:



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



## Classes

- class [DirVisualiser](#)

Klasa [DirVisualiser](#) Klasa służy do przechowywania właściwości widgetu służącego za graficzne przedstawienie danych w wybranym katalogu.

## Macros

- #define [DIR\\_SIZE\\_PREFIX](#) "\$\$\$"

## 8.4.1 Macro Definition Documentation

### 8.4.1.1 #define DIR\_SIZE\_PREFIX "\$\$\$"

#### Date

15.11.2017r.

#### Author

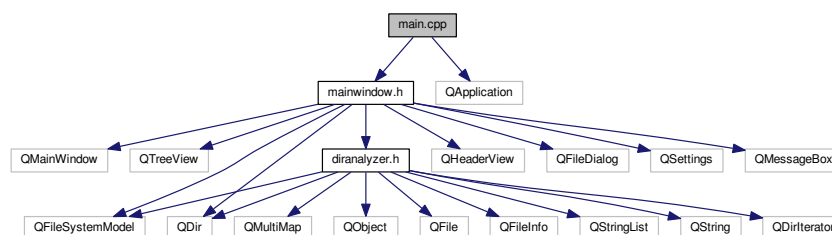
Damian Świerk

## 8.5 main.cpp File Reference

```
#include "mainwindow.h"
```

```
#include <QApplication>
```

Include dependency graph for main.cpp:



## Functions

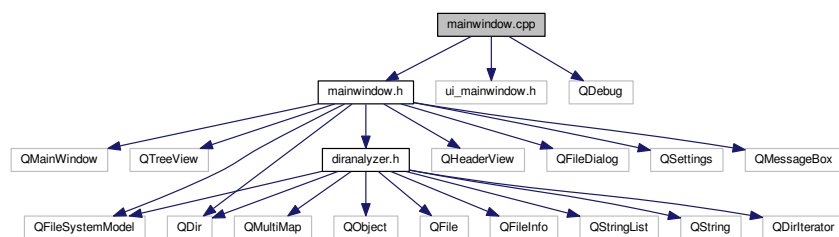
- int [main](#) (int argc, char \*argv[])

### 8.5.1 Function Documentation

#### 8.5.1.1 int main ( int argc, char \* argv[] )

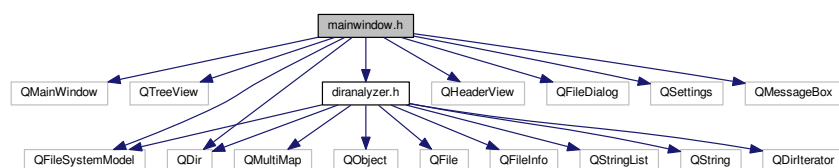
## 8.6/mainwindow.cpp File Reference

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QDebug>
Include dependency graph for mainwindow.cpp:
```

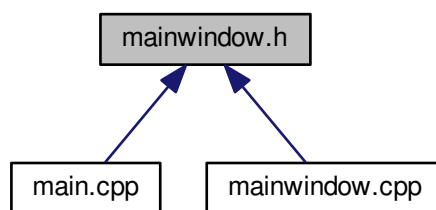


## 8.7/mainwindow.h File Reference

```
#include <QMainWindow>
#include <QTreeView>
#include <QFileSystemModel>
#include <QDir>
#include <QHeaderView>
#include <QFileDialog>
#include <QSettings>
#include <QMessageBox>
#include "diranalyzer.h"
Include dependency graph for mainwindow.h:
```



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



## Classes

- class [MainWindow](#)

*Klasa [MainWindow](#) Klasa Main Window - interfejs graficzny szperacza dyskowego. Jest to pomost między operacjami na katalogach i plikach, które wykonywane są w obiekcie `m_analyzer`, a graficzną wizualizacją danych w głównym oknie programu.*

## Namespaces

- [Ui](#)