SymulacjaZbiornika

0.1

Wygenerowano przez Doxygen 1.8.6

Cz, 4 cze 2015 23:55:51

Spis treści

Rozdział 1

Wizualizacja rozkładu ciśnienia cieczy na podstawie symulacji komputerowej

Autor

Adam Balawender,
Krzysztof Kwieciński,
AiR, ARR, W4

Data

11.05.2015

Wersja

0.1

Aplikacja dotyczy komputerowej symulacji zachowania cieczy oraz wizualizacji jej stanu i rozkładu ciśnienia w zbiorniku z płynem.

1.1 Opis projektu

Symulacja będzie obejmowała ruch cieczy w przekroju 2D wybranego naczynia. Ciecz zostanie przedstawiona na płaszczyźnie jako zbiór oddziaływujących ze sobą cząsteczek. Postaramy się, żeby jej zachowanie było możliwie zbliżone do rzeczywistego. Ruch płynu zostanie zamodelowany metodą numeryczną SPH (particle hydrodynamics - wygładzona hydrodynamika cząstek). Pozwoli to na realistyczne odwzorowanie zachowania cieczy. Możliwe będzie badanie cieczy o różnych parametrach, dlatego też modelowane będą jej właściwości fizyczne: gęstość i lepkość. Dodatkowo mierzone będzie ciśnienie cieczy i zostanie ono zwizualizowane jako odcień koloru płynu. Im będzie on ciemniejszy, tym wyższe ciśnienie będzie odzwierciedlał.

1.2 Funkcjonalnosci aplikacji

Najistotniejszymi funkcjonalnościami aplikacji będą:

- symulacja zachowania cieczy w zależności od zadanych warunków początkowych,
- możliwość przedefiniowania parametrów cieczy (gęstości, lepkości),
- możliwość obserwacji wyniku symulacji (położenia cząsteczek i rozkładu ciśnień).

Rozdział 2

Indeks przestrzeni nazw

2.1 Lista przestrzeni nazw

Tutaj znajdują się wszystkie przestrzenie nazw wraz z ich krótkimi opisami:

Rozdział 3

Indeks hierarchiczny

3.1 Hierarchia klas

Ta lista dziedziczenia posortowana jest z grubsza, choć nie całkowicie, alfabetycznie:

Rozdział 4

Indeks klas

4.1 Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

Rozdział 5

Indeks plików

5.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

Rozdział 6

Dokumentacja przestrzeni nazw

6.1 Dokumentacja przestrzeni nazw Ui

Komponenty

- class DMainWindow

Rozdział 7

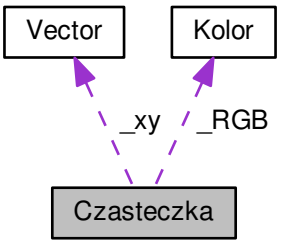
Dokumentacja klas

7.1 Dokumentacja klasy Czasteczka

Klasa modelująca czasteczke.

`#include <czasteczka.hh>`

Diagram współpracy dla Czasteczka:



Metody publiczne

- Czasteczka (Vector xy, int r, const Kolor &rgb)

  Konstruktor.
- void RysujCzasteczke (QPainter &Rysownik, const int Promien, const Kolor RGB, const double x, const double y)

  Metoda rysujaca czasteczke.
- Vector xy () const

  Interfejs pozwalajacy na odczyt prywatnych danych.
- Vector & xy ()

  Interfejs pozwalajacy na zmiane prywatnych danych.
- int Promien () const

  Interfejs pozwalajacy na odczyt prywatnych danych.
- int & Promien ()

  Interfejs pozwalajacy na zmiane prywatnych danych.
- Kolor RGB () const

Interfejs pozwalajacy na odczyt prywatnych danych.

- Kolor & RGB ()

  Interfejs pozwalajacy na zmiane prywatnych danych.

Atrybuty prywatne

- Vector _xy

  Atrybut opisujacy polozenie czasteczki.
- int _Promien

  Atrybut okreslajacy promien czasteczki.
- Kolor _RGB

  Atrybut opisujacy kolor czasteczki.

7.1.1 Opis szczegółowy

Klasa zawierajaca podstawowe atrybuty czasteczki.

Definicja w linii 29 pliku czasteczka.hh.

7.1.2 Dokumentacja konstruktora i destruktora

7.1.2.1 Czasteczka::Czasteczka (Vector *xy*, int *r*, const Kolor & *rgb*) [inline]

Konstruktor parametryczny, inicjalizujacy czasteczke podanymi wlasnosciami.

Parametry

in	*xy*	- wektor polozenia czasteczki
in	*r*	- promien czasteczki
in	*rgb*	- kolor czasteczki

Definicja w linii 40 pliku czasteczka.hh.

7.1.3 Dokumentacja funkcji składowych

7.1.3.1 int Czasteczka::Promien ( ) const [inline]

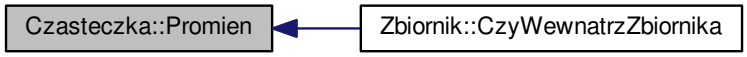
Interfejs pozwalajacy na odczyt prywatnych danych.

Zwraca

_Promien - prywatny atrybut opisujacy promien czasteczki

Definicja w linii 81 pliku czasteczka.hh.

Oto graf wywoływań tej funkcji:



7.1.3.2 int& Czasteczka::Promien ( ) [inline]

Interfejs pozwalajacy na zmiane prywatnych danych.

Zwraca

_Promien - referencja na prywatny atrybut opisujacy promien czasteczki

Definicja w linii 88 pliku czasteczka.hh.

7.1.3.3 Kolor Czasteczka::RGB ( ) const [inline]

Interfejs pozwalajacy na odczyt prywatnych danych.

Zwraca

_RGB - prywatny atrybut opisujacy polozenie kolor czasteczki

Definicja w linii 96 pliku czasteczka.hh.

7.1.3.4 Kolor& Czasteczka::RGB ( ) [inline]

Interfejs pozwalajacy na zmiane prywatnych danych.

Zwraca

_RGB - referencja na prywatny atrybut opisujacy promien czasteczki

Definicja w linii 103 pliku czasteczka.hh.

7.1.3.5 void Czasteczka::RysujCzasteczke (QPainter & *Rysownik*, const int *Promien*, const Kolor *RGB*, const double *x*, const double *y*)

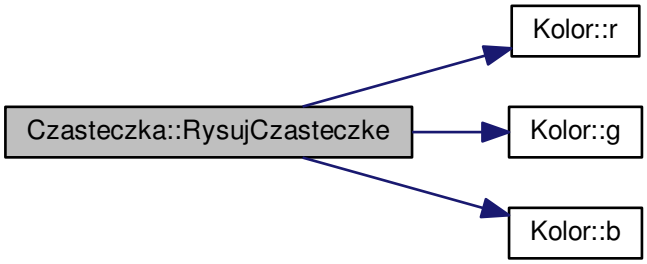
Rysuje czasteczke o zadanych parametrach

Parametry

in,out	*Rysownik*	- referencja na obiekt klasy QPainter
in	*Promien*	- promien czasteczki
in	*RGB*	- kolor czasteczki w formacie RGB
in	*x*	- polozenie czasteczki na osi x
in	*y*	- polozenie czasteczki na osi y

Definicja w linii 16 pliku czasteczka.cpp.

Oto graf wywołań dla tej funkcji:



7.1.3.6 Vector Czasteczka::xy ( ) const [inline]

Interfejs pozwalajacy na odczyt prywatnych danych.

Zwraca

_xy - prywatny atrybut opisujacy polozenie czasteczki

Definicja w linii 66 pliku czasteczka.hh.

Oto graf wywoływań tej funkcji:

7.1.3.7 Vector& Czasteczka::xy ( ) [inline]

Interfejs pozwalajacy na zmiane prywatnych danych.

Zwraca

_xy - referencja na prywatny atrybut opisujacy polozenie czasteczki

Definicja w linii 73 pliku czasteczka.hh.

7.1.4 Dokumentacja atrybutów składowych

7.1.4.1 int Czasteczka::_Promien [private]

Atrybut okreslajacy promien czasteczki.

Definicja w linii 118 pliku czasteczka.hh.

7.1.4.2 Kolor Czasteczka::_RGB [private]

Atrybut opisujacy kolor czasteczki.

Definicja w linii 125 pliku czasteczka.hh.

7.1.4.3 Vector Czasteczka::_xy [private]

Atrybut opisujacy polozenie czasteczki w kartezjanskim ukladzie wspolrzednych.

Definicja w linii 103 pliku czasteczka.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- czasteczka.hh
- czasteczka.cpp

7.2 Dokumentacja klasy DMainWindow

`#include <dmainwindow.h>`

Diagram dziedziczenia dla DMainWindow

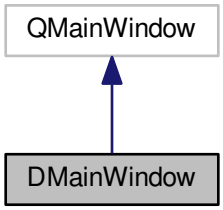
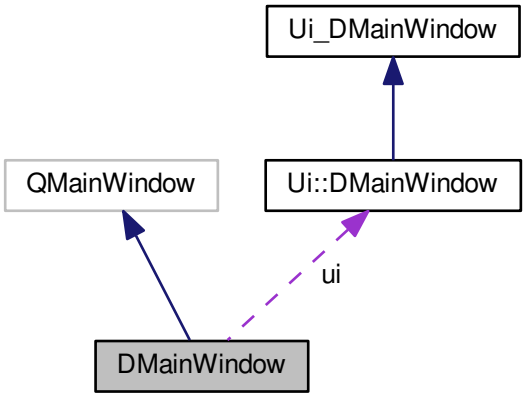


Diagram współpracy dla DMainWindow:



Metody publiczne

- DMainWindow (QWidget ∗parent=0)
- ∼DMainWindow ()

Atrybuty prywatne

- Ui::DMainWindow ∗ ui

7.2.1 Opis szczegółowy

Definicja w linii 10 pliku dmainwindow.h.

7.2.2 Dokumentacja konstruktora i destruktora

7.2.2.1 DMainWindow::DMainWindow (QWidget ∗ *parent* = 0) [explicit]

Definicja w linii 4 pliku dmainwindow.cpp.

Oto graf wywołań dla tej funkcji:

7.2.2.2 DMainWindow::∼DMainWindow ( )

Definicja w linii 11 pliku dmainwindow.cpp.

7.2.3 Dokumentacja atrybutów składowych

7.2.3.1 Ui::DMainWindow* DMainWindow::ui [private]

Definicja w linii 19 pliku dmainwindow.h.

Dokumentacja dla tej klasy została wygenerowana z plików:

- dmainwindow.h
- dmainwindow.cpp

7.3 Dokumentacja klasy Ui::DMainWindow

`#include <ui_dmainwindow.h>`

Diagram dziedziczenia dla Ui::DMainWindow

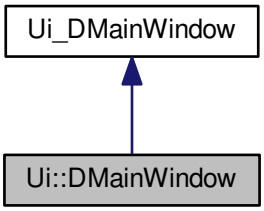
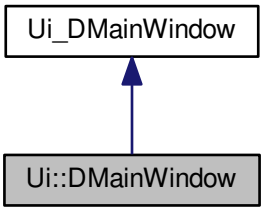


Diagram współpracy dla Ui::DMainWindow:



Dodatkowe Dziedziczone Składowe

7.3.1 Opis szczegółowy

Definicja w linii 200 pliku ui_dmainwindow.h.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- ui_dmainwindow.h

7.4 Dokumentacja klasy Kolor

Klasa modelująca kolor.

```
#include <kolor.hh>
```

Metody publiczne

- Kolor (int r, int g, int b)

  Konstruktor.
- Kolor (const Kolor &rgb)

  Konstruktor kopiujacy.
- int r () const

  Interfejs pozwalajacy na odczyt prywatnych danych.
- int & r ()

  Interfejs pozwalajacy na zmiane prywatnych danych.
- int g () const

  Interfejs pozwalajacy na odczyt prywatnych danych.
- int & g ()

  Interfejs pozwalajacy na zmiane prywatnych danych.
- int b () const

  Interfejs pozwalajacy na odczyt prywatnych danych.
- int & b ()

  Interfejs pozwalajacy na zmiane prywatnych danych.

Atrybuty prywatne

- int _r

  Atrybut opisujacy wartosc odcieniu czerwonego.
- int _g

  Atrybut opisujacy wartosc odcieniu zielonego.
- int _b

  Atrybut opisujacy wartosc odcieniu niebieskiego.

7.4.1 Opis szczegółowy

Klasa opisuje kolor w formacie RGB.

Definicja w linii 24 pliku kolor.hh.

7.4.2 Dokumentacja konstruktora i destruktora

7.4.2.1 Kolor::Kolor (int *r*, int *g*, int *b*) [inline]

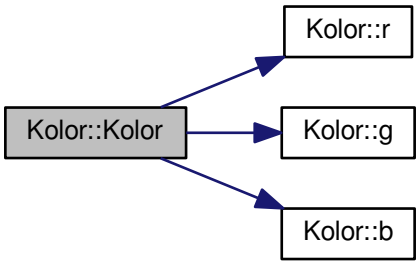
Konstruktor parametryczny, inicjalizujacy kolor podanymi wartosciami.

Parametry

in	*r*	- wartosc odcieniu czerwonego, [0, 255]
in	*g*	- wartosc odcieniu zielonego, [0, 255]
in	*b*	- wartosc odcieniu niebieskiego, [0, 255]

Definicja w linii 35 pliku kolor.hh.

Oto graf wywołań dla tej funkcji:



7.4.2.2 Kolor::Kolor (const Kolor & *rgb*) [inline]

Konstruktor kopiujacy.

Parametry

in	*rgb*	- obiekt do skopiowania

Definicja w linii 43 pliku kolor.hh.

7.4.3 Dokumentacja funkcji składowych

7.4.3.1 int Kolor::b ( ) const [inline]

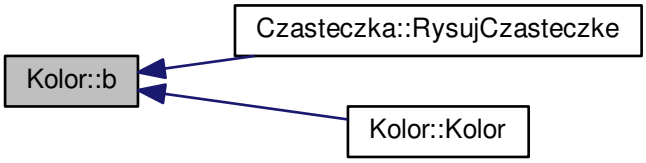
Interfejs pozwalajacy na odczyt prywatnych danych.

Zwraca

_b - prywatny atrybut opisujacy niebieski odcien

Definicja w linii 81 pliku kolor.hh.

Oto graf wywoływań tej funkcji:



7.4.3.2 int& Kolor::b ( ) [inline]

Interfejs pozwalajacy na zmiane prywatnych danych.

Zwraca

_b - referencja na prywatny atrybut opisujacy niebieski odcien

Definicja w linii 88 pliku kolor.hh.

7.4.3.3 int Kolor::g ( ) const [inline]

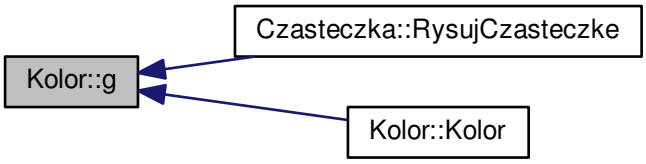
Interfejs pozwalajacy na odczyt prywatnych danych.

Zwraca

_g - prywatny atrybut opisujacy zielony odcien

Definicja w linii 66 pliku kolor.hh.

Oto graf wywoływań tej funkcji:



7.4.3.4 int& Kolor::g ( ) [inline]

Interfejs pozwalajacy na zmiane prywatnych danych.

Zwraca

_g - referencja na prywatny atrybut opisujacy zielony odcien

Definicja w linii 73 pliku kolor.hh.

7.4.3.5 int Kolor::r ( ) const [inline]

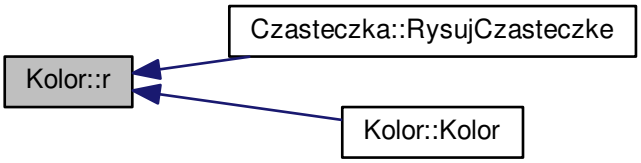
Interfejs pozwalajacy na odczyt prywatnych danych.

Zwraca

_r - prywatny atrybut opisujacy czerwony odcien

Definicja w linii 51 pliku kolor.hh.

Oto graf wywoływań tej funkcji:



7.4.3.6 int& Kolor::r ( ) [inline]

Interfejs pozwalajacy na zmiane prywatnych danych.

Zwraca

_r - referencja na prywatny atrybut opisujacy czerwony odcien

Definicja w linii 58 pliku kolor.hh.

7.4.4 Dokumentacja atrybutów składowych

7.4.4.1 int Kolor::_b [private]

Atrybut opisujacy wartosc odcieniu niebieskiego.

Definicja w linii 110 pliku kolor.hh.

7.4.4.2 int Kolor::_g [private]

Atrybut opisujacy wartosc odcieniu zielonego.

Definicja w linii 103 pliku kolor.hh.

7.4.4.3 int Kolor::_r [private]

Atrybut opisujacy wartosc odcieniu czerwonego.

Definicja w linii 88 pliku kolor.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- kolor.hh

7.5 Dokumentacja klasy OknoGlowne

Klasa modelujaca głowne okno aplikacji.

`#include <okno_glowne.hh>`

Diagram dziedziczenia dla OknoGlowne

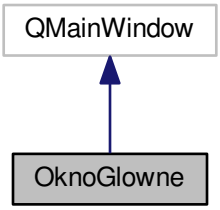
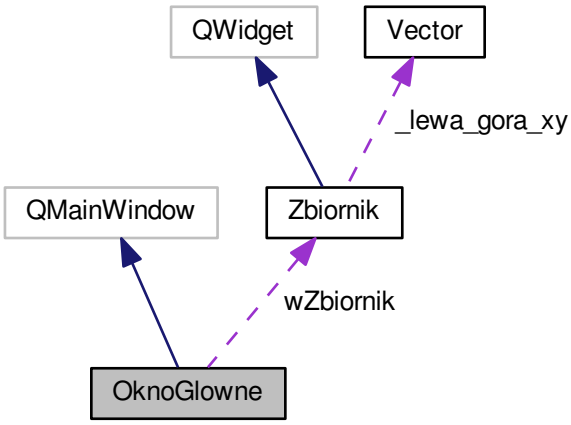


Diagram współpracy dla OknoGlowne:



Sloty publiczne

- void GdyOdpowiedniCzas ()

  Slot odpowiadajacy za aktualizacje danych. .
- void GdyNapis (const QString &)

  Slot odpowiadajacy za wyswietlenie napisu po otrzymaniu sygnalu.
- void on_playButton_clicked ()

  Slot odpowiadajacy za obsluge stanu play.
- void on_pauseButton_clicked ()

  Slot odpowiadajacy za obsluge stanu pauza.
- void on_stopButton_clicked ()

  Slot odpowiadajacy za obsluge stanu stop.
- void on_loadButton_clicked ()

  Slot odpowiadajacy za wczytanie danych z pliku.
- void on_lineEdit_returnPressed ()

  Slot odpowiadajacy za wczytanie danych z pliku.
- void on_sliderSzybkoscSym_valueChanged (int a)

  Slot odpowiadajacy za zmiane wartosci slidera.
- void on_action_Save_triggered ()

  Slot odpowiadajacy za przycisniecie przycisku Save.

Sygnały

- void ZglosNapis (const QString &)

  Sygnal zglaszajacy napis.

Metody publiczne

- OknoGlowne (QWidget ∗wRodzic=NULL)

  Konstruktor.
- virtual void paintEvent (QPaintEvent ∗event)

  Wirtualna metoda paintEvent wyrysowujaca obiekty na ekranie.
- void ZapiszSymulacjeDoPliku ()

  Metoda zapisujaca aktualny stan symulacji do pliku.
- void WczytajSymulacjeZPliku (const std::string nazwa_pliku)

  Metoda wczytujaca z pliku stan symulacji.

Atrybuty publiczne

- QTimer _Stoper

  Miernik czasu.

Atrybuty prywatne

- Zbiornik ∗ wZbiornik

  Wskaznik na zbiornik.
- QMenuBar ∗ menuBar

  Wskaznik na pasek menu.
- QAction ∗ action_Save

  Wskaznik na akcje przycisku menu Save.

- QAction * action_Exit

  Wskaznik na akcje przycisku menu Exit.
- QMenu * menu_File

  Wskaznik na akcje przycisku menu File.
- QMenu * menu_Edit

  Wskaznik na akcje przycisku menu Edit.
- QMenu * menu_Help

  Wskaznik na akcje przycisku menu Help.
- QStatusBar * statusBar

  Wskaznik na pasek statusowy.
- QToolBar * toolBar

  Wskaznik na pasek narzedziowy.
- QHBoxLayout * horizontalLayout

  Wskaznik na obszar do horyzontalnego rozmieszczenia przyciskow.
- QWidget * horizontalLayoutWidget

  Wskaznik na widget odpowiedzialny za horyzontalne wyswietlenie przyciskow.
- QPushButton * playButton

  Wskaznik na przycisk play.
- QPushButton * pauseButton

  Wskaznik na przycisk pause.
- QPushButton * stopButton

  Wskaznik na przycisk stop.
- QPushButton * loadButton

  Wskaznik na przycisk Wczytaj.
- QSlider * sliderSzybkoscSym

  Wskaznik na slider.
- QLCDNumber * lcdSzybkoscSym

  Wskaznik na LCD z szybkoscia symulacji.
- QLabel * labelSzybkoscSym

  Wskaznik na etykiete z szybkoscia symulacji.
- QLabel * labelCzasSym

  Wskaznik na etykiete z czasem symulacji.
- QLCDNumber * lcdCzasSym

  Wskaznik na LCD z czasem trwania symulacji.
- QLabel * labelLiczbaCzasteczek

  Wskaznik na etykiete z liczbe czasteczek.
- QLCDNumber * lcdLiczbaCzasteczek

  Wskaznik na LCD z liczbe czasteczek.
- QLineEdit * lineEdit

  Wskaznik na linijke do wpisywania tekstu.
- double _old_width

  Stara szerokosc okienka.
- double _old_height

  Stara wysokosc okienka.

Statyczne atrybuty prywatne

- static int licznik_plikow

  Sluzy do generowania unikatowych nazw plikow wyjsciowych.

7.5.1 Opis szczegółowy

Dzieki tej klasie wyswietlane jest okno glowne aplikacji.

Definicja w linii 68 pliku okno_glowne.hh.

7.5.2 Dokumentacja konstruktora i destruktora

7.5.2.1 OknoGlowne::OknoGlowne (QWidget * *wRodzic* = NULL)

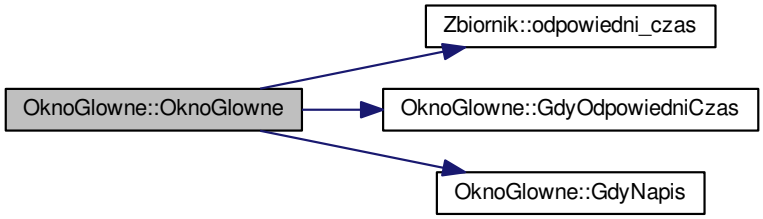
Konstruktor parametryczny.

Parametry

in,out	*wRodzic*	- wskaznik na rodzica

Definicja w linii 16 pliku okno_glowne.cpp.

Oto graf wywołań dla tej funkcji:



7.5.3 Dokumentacja funkcji składowych

7.5.3.1 void OknoGlowne::GdyNapis (const QString & *Napis*) [slot]

Odpowiada za wyswietlenie napisu na belce statusowej.

Parametry

in	*Napis*	- napis do wyswietlenia

Definicja w linii 223 pliku okno_glowne.cpp.

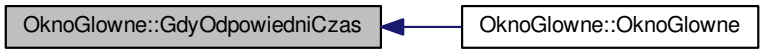
Oto graf wywoływań tej funkcji:

7.5.3.2 void OknoGlowne::GdyOdpowiedniCzas ( ) [slot]

Odpowiada za uaktualnienie okienka w odpowiednich momentach.

Definicja w linii 215 pliku okno_glowne.cpp.

Oto graf wywoływań tej funkcji:

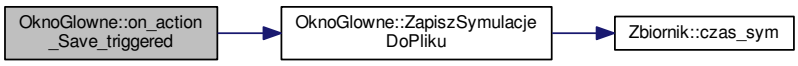


7.5.3.3 void OknoGlowne::on_action_Save_triggered ( ) [slot]

Odpowiada za wykonanie odpowiednich czynnosci po przycisnieciu Save.

Definicja w linii 204 pliku okno_glowne.cpp.

Oto graf wywołań dla tej funkcji:

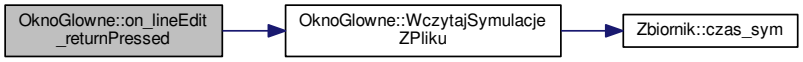


7.5.3.4 void OknoGlowne::on_lineEdit_returnPressed ( ) [slot]

Odpowiada za wczytanie danych z pliku.

Definicja w linii 191 pliku okno_glowne.cpp.

Oto graf wywołań dla tej funkcji:

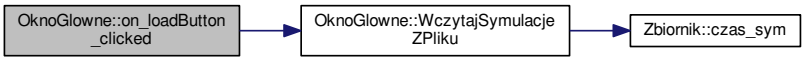


7.5.3.5 void OknoGlowne::on_loadButton_clicked ( ) [slot]

Odpowiada za wczytanie danych z pliku.

Definicja w linii 185 pliku okno_glowne.cpp.

Oto graf wywołań dla tej funkcji:



7.5.3.6 void OknoGlowne::on_pauseButton_clicked ( ) [slot]

Odpowiada za wykonanie odpowiednich czynnosci w trakcie stanu pauza.

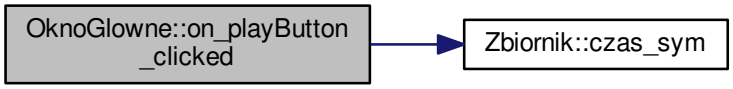
Definicja w linii 177 pliku okno_glowne.cpp.

7.5.3.7 void OknoGlowne::on_playButton_clicked ( ) [slot]

Odpowiada za wykonanie odpowiednich czynnosci w trakcie stanu play.

Definicja w linii 167 pliku okno_glowne.cpp.

Oto graf wywołań dla tej funkcji:

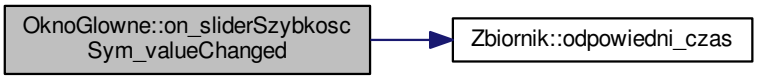


7.5.3.8 void OknoGlowne::on_sliderSzybkoscSym_valueChanged (int *a*) [slot]

Odpowiada za wykonanie odpowiednich czynnosci po zmianie wartosci slidera.

Definicja w linii 197 pliku okno_glowne.cpp.

Oto graf wywołań dla tej funkcji:



7.5.3.9 void OknoGlowne::on_stopButton_clicked ( ) [slot]

Odpowiada za wykonanie odpowiednich czynnosci w trakcie stanu stop.

Definicja w linii 181 pliku okno_glowne.cpp.

7.5.3.10 void OknoGlowne::paintEvent (QPaintEvent * *event*) [virtual]

Odziedziczona wirtualna metoda paintEvent. Rysuje zbiornik i przyciski w nowym miejscu.

Parametry

in,out	*event*	- wskaznik obiekt klasy QPaintEvent

Definicja w linii 227 pliku okno_glowne.cpp.

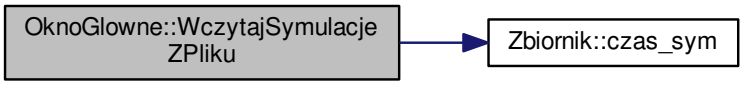
Oto graf wywołań dla tej funkcji:

7.5.3.11 void OknoGlowne::WczytajSymulacjeZPliku (const std::string *nazwa_pliku*)

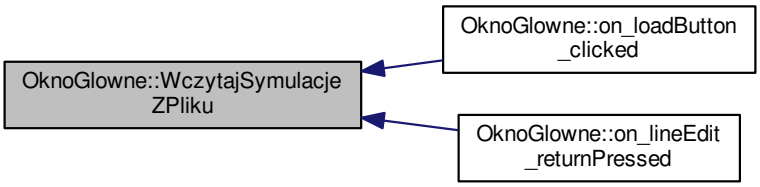
Wczytuje stan symulacji (czas, liczba czasteczek, dane czasteczek).

Definicja w linii 266 pliku okno_glowne.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:

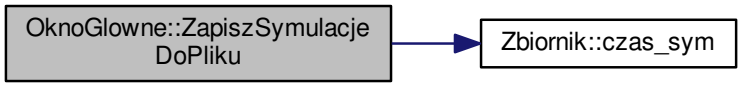


7.5.3.12 void OknoGlowne::ZapiszSymulacjeDoPliku ( )

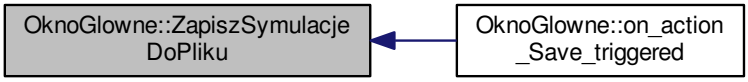
Zapisuje aktualny stan symulacji (czas, liczba czasteczek, dane czasteczek) do pliku.

Definicja w linii 243 pliku okno_glowne.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



7.5.3.13 void OknoGlowne::ZglosNapis (const QString & *_t1*) [signal]

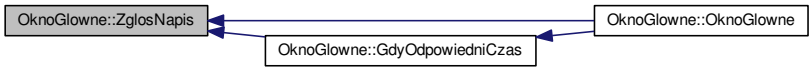
Sygnal zglaszajacy napis do odpowiedniego slotu.

Parametry

in	*_t1*	- napis do zgloszenia

Definicja w linii 121 pliku moc_okno_glowne.cpp.

Oto graf wywoływań tej funkcji:



7.5.4 Dokumentacja atrybutów składowych

7.5.4.1 double OknoGlowne::_old_height [private]

Stara wysokosc okienka.

Definicja w linii 367 pliku okno_glowne.hh.

7.5.4.2 double OknoGlowne::_old_width [private]

Stara szerokosc okienka.

Definicja w linii 361 pliku okno_glowne.hh.

7.5.4.3 QTimer OknoGlowne::_Stoper

Miernik czasu.

Definicja w linii 353 pliku okno_glowne.hh.

7.5.4.4 QAction* OknoGlowne::action_Exit [private]

Wskaznik na akcje przycisku menu Exit.

Definicja w linii 212 pliku okno_glowne.hh.

7.5.4.5 QAction* OknoGlowne::action_Save [private]

Wskaznik na akcje przycisku menu Save.

Definicja w linii 205 pliku okno_glowne.hh.

7.5.4.6 QHBoxLayout* OknoGlowne::horizontalLayout [private]

Wskaznik na obszar do horyzontalnego rozmieszczenia przyciskow.

Definicja w linii 254 pliku okno_glowne.hh.

7.5.4.7 QWidget* OknoGlowne::horizontalLayoutWidget [private]

Wskaznik na widget odpowiedzialny za horyzontalne wyswietlenie przyciskow.

Definicja w linii 261 pliku okno_glowne.hh.

7.5.4.8 QLabel* OknoGlowne::labelCzasSym [private]

Etykieta dla czasu symulacji.

Definicja w linii 317 pliku okno_glowne.hh.

7.5.4.9 QLabel* OknoGlowne::labelLiczbaCzasteczek [private]

Etykieta dla liczby symulowanych czasteczek.

Definicja w linii 331 pliku okno_glowne.hh.

7.5.4.10 QLabel* OknoGlowne::labelSzybkoscSym [private]

Etykieta dla szybkosci symulacji.

Definicja w linii 310 pliku okno_glowne.hh.

7.5.4.11 QLCDNumber* OknoGlowne::lcdCzasSym [private]

Wskaznik na LCD z szybkoscia symulacji. Wyswietla jej czas trwania.

Definicja w linii 324 pliku okno_glowne.hh.

7.5.4.12 QLCDNumber* OknoGlowne::lcdLiczbaCzasteczek [private]

Wyswietla liczbe symulowanych czasteczek.

Definicja w linii 338 pliku okno_glowne.hh.

7.5.4.13 QLCDNumber* OknoGlowne::lcdSzybkoscSym [private]

Wskaznik na LCD z szybkoscia symulacji. Wyswietla jej szybkosc.

Definicja w linii 303 pliku okno_glowne.hh.

7.5.4.14 int OknoGlowne::licznik_plikow [static],[private]

Sluzy do generowania unikatowych nazw plikow wyjsciowych.

Definicja w linii 184 pliku okno_glowne.hh.

7.5.4.15 QLineEdit* OknoGlowne::lineEdit [private]

Wskazuje linijke do wpisywania tekstu.

Definicja w linii 345 pliku okno_glowne.hh.

7.5.4.16 QPushButton* OknoGlowne::loadButton [private]

Wskaznik na przycisk Wczytaj. Wczytuje symulacje.

Definicja w linii 289 pliku okno_glowne.hh.

7.5.4.17 QMenu* OknoGlowne::menu_Edit [private]

Wskaznik na akcje przycisku menu Edit.

Definicja w linii 226 pliku okno_glowne.hh.

7.5.4.18 QMenu* OknoGlowne::menu_File [private]

Wskaznik na akcje przycisku menu File.

Definicja w linii 219 pliku okno_glowne.hh.

7.5.4.19 QMenu* OknoGlowne::menu_Help [private]

Wskaznik na akcje przycisku menu Help.

Definicja w linii 233 pliku okno_glowne.hh.

7.5.4.20 QMenuBar* OknoGlowne::menuBar [private]

Wskaznik na pasek menu.

Definicja w linii 198 pliku okno_glowne.hh.

7.5.4.21 QPushButton* OknoGlowne::pauseButton [private]

Wskaznik na przycisk pause. Wstrzymuje symulacje.

Definicja w linii 275 pliku okno_glowne.hh.

7.5.4.22 QPushButton* OknoGlowne::playButton [private]

Wskaznik na przycisk play. Uruchamia symulacje.

Definicja w linii 268 pliku okno_glowne.hh.

7.5.4.23 QSlider* OknoGlowne::sliderSzybkoscSym [private]

Wskaznik na slider. Steruje szybkoscia symulacji.

Definicja w linii 296 pliku okno_glowne.hh.

7.5.4.24 QStatusBar* OknoGlowne::statusBar [private]

Wskaznik na pasek statusowy.

Definicja w linii 240 pliku okno_glowne.hh.

7.5.4.25 QPushButton* OknoGlowne::stopButton [private]

Wskaznik na przycisk stop. Zatrzymuje symulacje.

Definicja w linii 282 pliku okno_glowne.hh.

7.5.4.26 QToolBar* OknoGlowne::toolBar [private]

Wskaznik na pasek narzedziowy.

Definicja w linii 247 pliku okno_glowne.hh.

7.5.4.27 Zbiornik* OknoGlowne::wZbiornik [private]

Wskaznik na zbiornik.

Definicja w linii 191 pliku okno_glowne.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- okno_glowne.hh
- moc_okno_glowne.cpp
- okno_glowne.cpp

7.6 Dokumentacja struktury params_t

Atrybuty publiczne

- unsigned nframes
- unsigned npframe
- float h
- float dt
- float rho0
- float k
- float mu
- float gx
- float gy
- float mass

7.6.1 Opis szczegółowy

Definicja w linii 137 pliku simulation.cpp.

7.6.2 Dokumentacja atrybutów składowych

7.6.2.1 float params_t::dt

Definicja w linii 141 pliku simulation.cpp.

7.6.2.2 float params_t::gx

Definicja w linii 145 pliku simulation.cpp.

7.6.2.3 float params_t::gy

Definicja w linii 146 pliku simulation.cpp.

7.6.2.4 float params_t::h

Definicja w linii 140 pliku simulation.cpp.

7.6.2.5 float params_t::k

Definicja w linii 143 pliku simulation.cpp.

7.6.2.6 float params_t::mass

Definicja w linii 147 pliku simulation.cpp.

7.6.2.7 float params_t::mu

Definicja w linii 144 pliku simulation.cpp.

7.6.2.8 unsigned params_t::nframes

Definicja w linii 138 pliku simulation.cpp.

7.6.2.9 unsigned params_t::npframe

Definicja w linii 139 pliku simulation.cpp.

7.6.2.10 float params_t::rho0

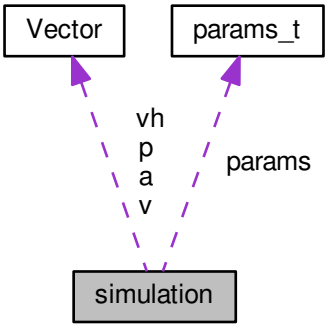
Definicja w linii 142 pliku simulation.cpp.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- simulation.cpp

7.7 Dokumentacja klasy simulation

Diagram współpracy dla simulation:



Typy publiczne

- typedef bool(simulation::∗ indicate_fun_t)(float, float)

Metody publiczne

- simulation (const unsigned &_n, params_t ∗_param)

  konstruktor klasy
- void compute_density (params_t ∗_p)

  funkcja
- void compute_accel (params_t ∗params)
- void damp_reflect_x (float _barrier, unsigned i)
- void damp_reflect_y (float _barrier, unsigned i)
- void reflect_bc ()

- void leapfrog_start (float dt)
- void leapfrog_step (float dt)
- void check_state ()
- bool box_indicator (float x, float y)
- simulation & place_particles (params_t *params, indicate_fun_t indicate_fun)
- void init ()
- void go ()
- const unsigned & getN () const

Atrybuty prywatne

- unsigned n
- float * rho

  Liczba cząstek.
- Vector * p

  Gęstości.
- Vector * vh

  Pozycje.
- Vector * v

  Prędkości (half step)
- Vector * a

  Prędkości (full step)
- params_t * params

  Przyspieszenia.

Przyjaciele

- std::ostream & operator<< (std::ostream &_os, const simulation &_s)

7.7.1 Opis szczegółowy

Definicja w linii 162 pliku simulation.cpp.

7.7.2 Dokumentacja składowych definicji typu

7.7.2.1 typedef bool(simulation::* simulation::indicate_fun_t)(float, float)

Definicja w linii 314 pliku simulation.cpp.

7.7.3 Dokumentacja konstruktora i destruktora

7.7.3.1 simulation::simulation (const unsigned & *_n*, params_t * *_param*) [inline]

Alokuje pamięć na potrzeby przechowania stanu symulacji

Parametry

in	*rozmiar*	

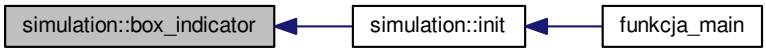
Definicja w linii 181 pliku simulation.cpp.

7.7.4 Dokumentacja funkcji składowych

7.7.4.1 bool simulation::box_indicator (float *x*, float *y*) [inline]

Definicja w linii 316 pliku simulation.cpp.

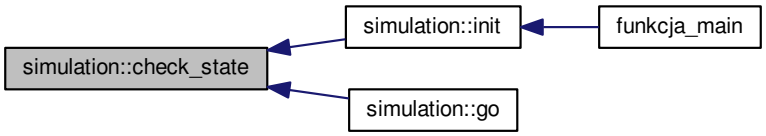
Oto graf wywoływań tej funkcji:



7.7.4.2 void simulation::check_state ( ) [inline]

Definicja w linii 306 pliku simulation.cpp.

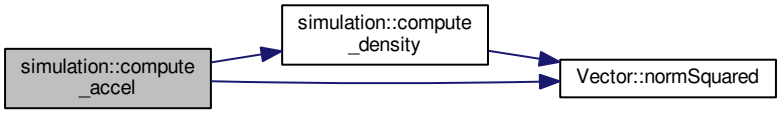
Oto graf wywoływań tej funkcji:



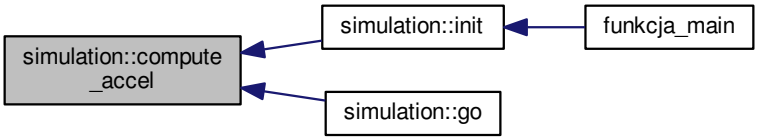
7.7.4.3 void simulation::compute_accel (params_t * *params*) [inline]

Definicja w linii 213 pliku simulation.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:

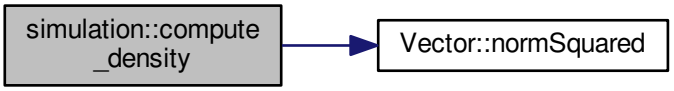


7.7.4.4 void simulation::compute_density (params_t * _p) [inline]

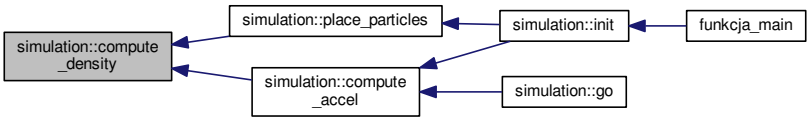
$$|I| = |\alpha|$$

Definicja w linii 191 pliku simulation.cpp.

Oto graf wywołań dla tej funkcji:



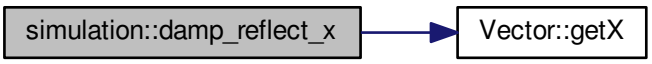
Oto graf wywoływań tej funkcji:



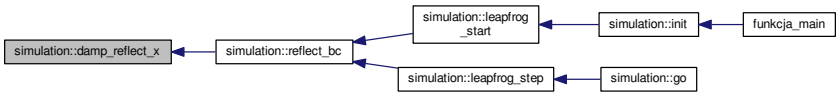
7.7.4.5 void simulation::damp_reflect_x (float _barrier, unsigned i) [inline]

Definicja w linii 252 pliku simulation.cpp.

Oto graf wywołań dla tej funkcji:



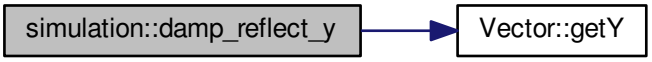
Oto graf wywoływań tej funkcji:



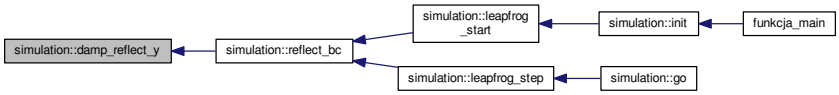
7.7.4.6 void simulation::damp_reflect_y (float *_barrier,* unsigned *i*) [inline]

Definicja w linii 265 pliku simulation.cpp.

Oto graf wywołań dla tej funkcji:



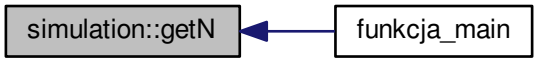
Oto graf wywoływań tej funkcji:



7.7.4.7 const unsigned& simulation::getN ( ) const [inline]

Definicja w linii 356 pliku simulation.cpp.

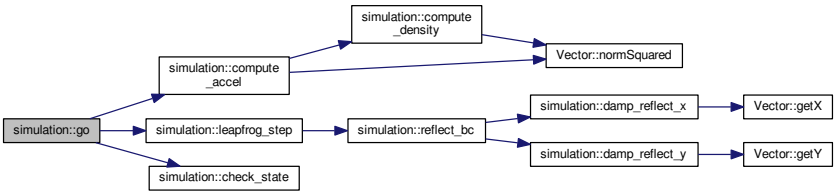
Oto graf wywoływań tej funkcji:



7.7.4.8 void simulation::go ( ) [inline]

Definicja w linii 349 pliku simulation.cpp.

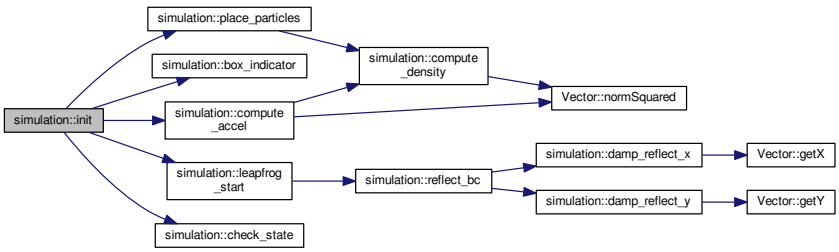
Oto graf wywołań dla tej funkcji:



7.7.4.9 void simulation::init ( ) [inline]

Definicja w linii 343 pliku simulation.cpp.

Oto graf wywołań dla tej funkcji:

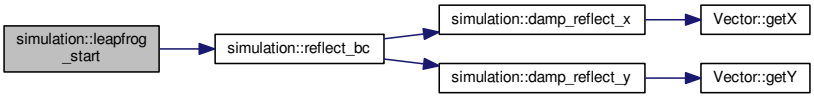


Oto graf wywoływań tej funkcji:

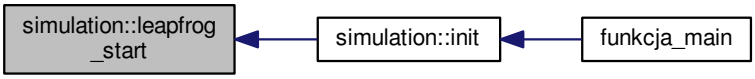
7.7.4.10 void simulation::leapfrog_start (float *dt*) [inline]

Definicja w linii 289 pliku simulation.cpp.

Oto graf wywołań dla tej funkcji:



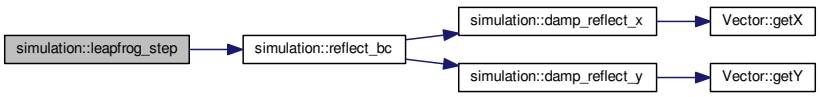
Oto graf wywoływań tej funkcji:



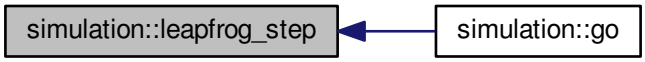
7.7.4.11 void simulation::leapfrog_step (float *dt*) [inline]

Definicja w linii 297 pliku simulation.cpp.

Oto graf wywołań dla tej funkcji:



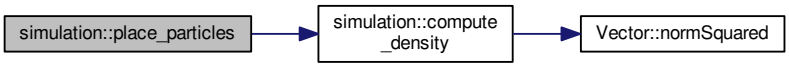
Oto graf wywoływań tej funkcji:



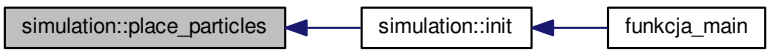
7.7.4.12 simulation& simulation::place_particles (params_t * *params*, indicate_fun_t *indicate_fun*) [inline]

Definicja w linii 320 pliku simulation.cpp.

Oto graf wywołań dla tej funkcji:



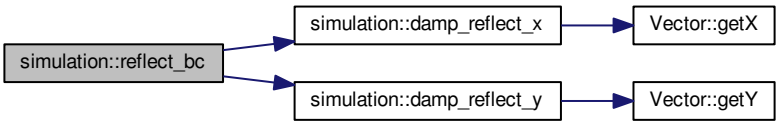
Oto graf wywoływań tej funkcji:



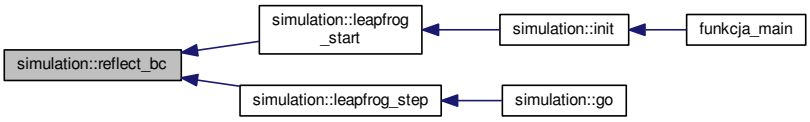
7.7.4.13 void simulation::reflect_bc ( ) [inline]

Definicja w linii 279 pliku simulation.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



7.7.5 Dokumentacja przyjaciół i funkcji związanych

7.7.5.1 std::ostream& operator<< (std::ostream & *_os*, const simulation & *_s*) [friend]

Definicja w linii 360 pliku simulation.cpp.

7.7.6 Dokumentacja atrybutów składowych

7.7.6.1 Vector* simulation::a [private]

Definicja w linii 170 pliku simulation.cpp.

7.7.6.2 unsigned simulation::n [private]

Definicja w linii 165 pliku simulation.cpp.

7.7.6.3 Vector* simulation::p [private]

Definicja w linii 167 pliku simulation.cpp.

7.7.6.4 params_t* simulation::params [private]

Definicja w linii 172 pliku simulation.cpp.

7.7.6.5 float* simulation::rho [private]

Definicja w linii 166 pliku simulation.cpp.

7.7.6.6 Vector* simulation::v [private]

Definicja w linii 169 pliku simulation.cpp.

7.7.6.7 Vector* simulation::vh [private]

Definicja w linii 168 pliku simulation.cpp.

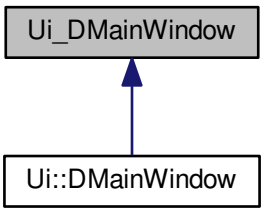
Dokumentacja dla tej klasy została wygenerowana z pliku:

- simulation.cpp

7.8 Dokumentacja klasy Ui_DMainWindow

```
#include <ui_dmainwindow.h>
```

Diagram dziedziczenia dla Ui_DMainWindow



Metody publiczne

- void setupUi (QMainWindow *DMainWindow)
- void retranslateUi (QMainWindow *DMainWindow)

Atrybuty publiczne

- QAction * action_Save
- QAction * action_Exit
- QAction * actionExit
- QAction * actionPlay
- QWidget * centralWidget
- QWidget * horizontalLayoutWidget
- QHBoxLayout * horizontalLayout
- QPushButton * playButton
- QSpacerItem * verticalSpacer
- QPushButton * pauseButton
- QPushButton * stopButton
- QSlider * sliderSzybkoscSym
- QLCDNumber * lcdSzybkoscSym
- QLineEdit * lineSzybkoscSym
- QLineEdit * lineCzasSym
- QLCDNumber * lcdCzasSym
- QLineEdit * lineLiczbaCzasteczek
- QLCDNumber * lcdLiczbaCzasteczek
- QLabel * label
- QMenuBar * menuBar
- QMenu * menu_File
- QMenu * menu_Edit
- QMenu * menu_Help
- QToolBar * mainToolBar
- QStatusBar * statusBar
- QToolBar * toolBar

7.8.1 Opis szczegółowy

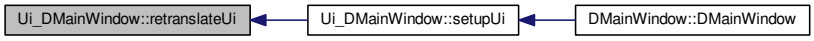
Definicja w linii 33 pliku ui_dmainwindow.h.

7.8.2 Dokumentacja funkcji składowych

7.8.2.1 void Ui_DMainWindow::retranslateUi (QMainWindow * *DMainWindow*) [inline]

Definicja w linii 177 pliku ui_dmainwindow.h.

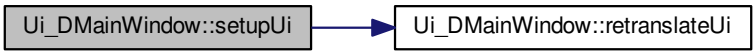
Oto graf wywoływań tej funkcji:



7.8.2.2 void Ui_DMainWindow::setupUi (QMainWindow * *DMainWindow*) [inline]

Definicja w linii 63 pliku ui_dmainwindow.h.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:

7.8.3 Dokumentacja atrybutów składowych

7.8.3.1 QAction* Ui_DMainWindow::action_Exit

Definicja w linii 37 pliku ui_dmainwindow.h.

7.8.3.2 QAction* Ui_DMainWindow::action_Save

Definicja w linii 36 pliku ui_dmainwindow.h.

7.8.3.3 QAction* Ui_DMainWindow::actionExit

Definicja w linii 38 pliku ui_dmainwindow.h.

7.8.3.4 QAction* Ui_DMainWindow::actionPlay

Definicja w linii 39 pliku ui_dmainwindow.h.

7.8.3.5 QWidget* Ui_DMainWindow::centralWidget

Definicja w linii 40 pliku ui_dmainwindow.h.

7.8.3.6 QHBoxLayout* Ui_DMainWindow::horizontalLayout

Definicja w linii 42 pliku ui_dmainwindow.h.

7.8.3.7 QWidget* Ui_DMainWindow::horizontalLayoutWidget

Definicja w linii 41 pliku ui_dmainwindow.h.

7.8.3.8 QLabel* Ui_DMainWindow::label

Definicja w linii 54 pliku ui_dmainwindow.h.

7.8.3.9 QLCDNumber* Ui_DMainWindow::lcdCzasSym

Definicja w linii 51 pliku ui_dmainwindow.h.

7.8.3.10 QLCDNumber* Ui_DMainWindow::lcdLiczbaCzasteczek

Definicja w linii 53 pliku ui_dmainwindow.h.

7.8.3.11 QLCDNumber* Ui_DMainWindow::lcdSzybkoscSym

Definicja w linii 48 pliku ui_dmainwindow.h.

7.8.3.12 QLineEdit* Ui_DMainWindow::lineCzasSym

Definicja w linii 50 pliku ui_dmainwindow.h.

7.8.3.13 QLineEdit* Ui_DMainWindow::lineLiczbaCzasteczek

Definicja w linii 52 pliku ui_dmainwindow.h.

7.8.3.14 QLineEdit* Ui_DMainWindow::lineSzybkoscSym

Definicja w linii 49 pliku ui_dmainwindow.h.

7.8.3.15 QToolBar* Ui_DMainWindow::mainToolBar

Definicja w linii 59 pliku ui_dmainwindow.h.

7.8.3.16 QMenu* Ui_DMainWindow::menu_Edit

Definicja w linii 57 pliku ui_dmainwindow.h.

7.8.3.17 QMenu* Ui_DMainWindow::menu_File

Definicja w linii 56 pliku ui_dmainwindow.h.

7.8.3.18 QMenu* Ui_DMainWindow::menu_Help

Definicja w linii 58 pliku ui_dmainwindow.h.

7.8.3.19 QMenuBar* Ui_DMainWindow::menuBar

Definicja w linii 55 pliku ui_dmainwindow.h.

7.8.3.20 QPushButton* Ui_DMainWindow::pauseButton

Definicja w linii 45 pliku ui_dmainwindow.h.

7.8.3.21 QPushButton* Ui_DMainWindow::playButton

Definicja w linii 43 pliku ui_dmainwindow.h.

7.8.3.22 QSlider* Ui_DMainWindow::sliderSzybkoscSym

Definicja w linii 47 pliku ui_dmainwindow.h.

7.8.3.23 QStatusBar* Ui_DMainWindow::statusBar

Definicja w linii 60 pliku ui_dmainwindow.h.

7.8.3.24 QPushButton* Ui_DMainWindow::stopButton

Definicja w linii 46 pliku ui_dmainwindow.h.

7.8.3.25 QToolBar* Ui_DMainWindow::toolBar

Definicja w linii 61 pliku ui_dmainwindow.h.

7.8.3.26 QSpacerItem* Ui_DMainWindow::verticalSpacer

Definicja w linii 44 pliku ui_dmainwindow.h.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- ui_dmainwindow.h

7.9 Dokumentacja klasy Vector

klasa Vector

```
#include <vector.hh>
```

Metody publiczne

- Vector (float _x=0, float _y=0)

  Współrzędne wektora.
- const float getX () const

  zwraga pierwszą współrzędną wektora
- const float getY () const

  pobiera drugą współrzędną wektora
- float & getX ()

  pobiera pierwszą współrzędną wektora
- float & getY ()

  pobiera drugą współrzędną wektora
- Vector & operator+= (const Vector &_v)

  operator dodawania
- Vector & operator-= (const Vector &_v)

  operator odejmowania
- Vector & operator∗= (const float &_c)

  operator skalowania
- Vector operator+ (const Vector &_v) const

  operator dodawania
- Vector operator- (const Vector &_v) const

  operator odejmowania
- Vector operator∗ (const float &_c) const

  operator skalowania
- Vector & operator= (const Vector &_v)

  operator przypisania
- float normSquared ()

  kwadrat normy
- Vector (float _x=0, float _y=0)

  Współrzędne wektora.
- const float getX () const

  zwraga pierwszą współrzędną wektora
- const float getY () const

  pobiera drugą współrzędną wektora
- float & getX ()

  pobiera pierwszą współrzędną wektora
- float & getY ()

  pobiera drugą współrzędną wektora
- Vector & operator+= (const Vector &_v)

  operator dodawania
- Vector & operator-= (const Vector &_v)

  operator odejmowania
- Vector & operator∗= (const float &_c)

  operator skalowania
- Vector operator+ (const Vector &_v) const

operator dodawania

- Vector operator- (const Vector &_v) const

  operator odejmowania

- Vector operator∗ (const float &_c) const

  operator skalowania

- Vector & operator= (const Vector &_v)

  operator przypisania

- float normSquared ()

  kwadrat normy

Atrybuty prywatne

- float x
- float y

Przyjaciele

- std::ostream & operator<< (std::ostream &_os, const Vector &_s)
- std::ostream & operator<< (std::ostream &_os, const Vector &_s)

7.9.1 Opis szczegółowy

Posiada metody do obsługi dwuelementowych wektorów o współrzędnych typu float

Definicja w linii 9 pliku vector.hh.

7.9.2 Dokumentacja konstruktora i destruktora

7.9.2.1 Vector::Vector (float _x = 0, float _y = 0) [inline]

konstruktor wektora

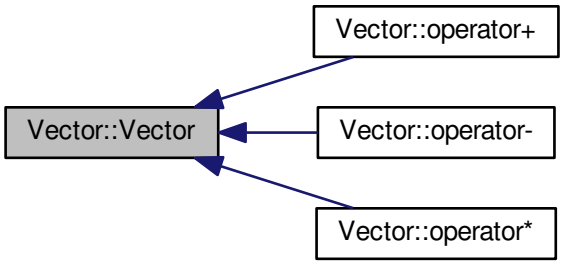
Ustawia współrzędne wektora

Parametry

in	_x	- pierwsza współrzędna (domyślnie: 0)
in	_y	- druga współrzędna (domyślnie: 0)

Definicja w linii 21 pliku vector.hh.

Oto graf wywoływań tej funkcji:



7.9.2.2 Vector::Vector (float _x = 0, float _y = 0) [inline]

konstruktor wektora

Ustawia współrzędne wektora

Parametry

in	_x	- pierwsza współrzędna (domyślnie: 0)
in	_y	- druga współrzędna (domyślnie: 0)

Definicja w linii 47 pliku simulation.cpp.

7.9.3 Dokumentacja funkcji składowych

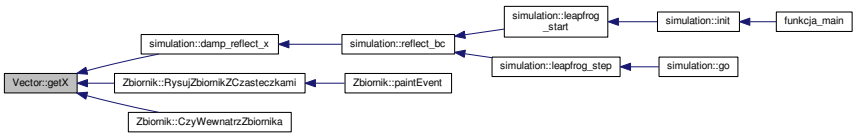
7.9.3.1 const float Vector::getX ( ) const [inline]

Zwraca

pierwsza współrzędna wektora jako stała

Definicja w linii 26 pliku vector.hh.

Oto graf wywoływań tej funkcji:



7.9.3.2 float& Vector::getX ( ) [inline]

Zwraca

pierwsza współrzędna wektora jako stała

Definicja w linii 36 pliku vector.hh.

7.9.3.3 const float Vector::getX ( ) const [inline]

Zwraca

pierwsza współrzędna wektora jako stała

Definicja w linii 52 pliku simulation.cpp.

7.9.3.4 float& Vector::getX ( ) [inline]

Zwraca

pierwsza współrzędna wektora jako stała

Definicja w linii 62 pliku simulation.cpp.

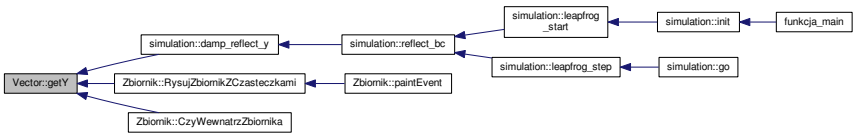
7.9.3.5 const float Vector::getY ( ) const [inline]

Zwraca

druga współrzędna wektora jako stała

Definicja w linii 31 pliku vector.hh.

Oto graf wywoływań tej funkcji:



7.9.3.6 float& Vector::getY ( ) [inline]

Zwraca

druga współrzędna wektora jako stała

Definicja w linii 41 pliku vector.hh.

7.9.3.7 const float Vector::getY ( ) const [inline]

Zwraca

druga współrzędna wektora jako stała

Definicja w linii 57 pliku simulation.cpp.

7.9.3.8 float& Vector::getY ( ) [inline]

Zwraca

druga współrzędna wektora jako stała

Definicja w linii 67 pliku simulation.cpp.

7.9.3.9 float Vector::normSquared ( ) [inline]

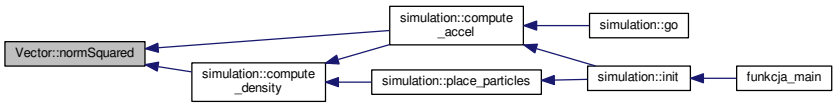
Zwraca sumę kwadratów współrzędnych wektora

Zwraca

wartość kwadratu normy

Definicja w linii 104 pliku vector.hh.

Oto graf wywoływań tej funkcji:



7.9.3.10 float Vector::normSquared ( ) [inline]

Zwraca sumę kwadratów współrzędnych wektora

Zwraca

wartość kwadratu normy

Definicja w linii 130 pliku simulation.cpp.

7.9.3.11 Vector Vector::operator∗ (const float & _c) const [inline]

Skaluje wektor

Parametry

in	_c	- współczynnik skalowania

Zwraca

przeskalowany wektor

Definicja w linii 89 pliku vector.hh.

Oto graf wywołań dla tej funkcji:

7.9.3.12 Vector Vector::operator∗ (const float & _c) const [inline]

Skaluje wektor

Parametry

in	_c	- współczynnik skalowania

Zwraca

przeskalowany wektor

Definicja w linii 115 pliku simulation.cpp.

Oto graf wywołań dla tej funkcji:

7.9.3.13 Vector& Vector::operator∗= (const float & _c) [inline]

Skaluje wektor

Parametry

in	_c	- współczynnik

Zwraca

referencja na pierwotny obiekt

Definicja w linii 65 pliku vector.hh.

7.9.3.14 Vector& Vector::operator*= (const float & _c) [inline]

Skaluje wektor

Parametry

in	_c	- współczynnik

Zwraca

referencja na pierwotny obiekt

Definicja w linii 91 pliku simulation.cpp.

7.9.3.15 Vector Vector::operator+ (const Vector & _v) const [inline]

Dodaje wartość dwóch wektorów

Parametry

in	_v	- inny wektor

Zwraca

wektor będący sumą

Definicja w linii 73 pliku vector.hh.

Oto graf wywołań dla tej funkcji:

7.9.3.16 Vector Vector::operator+ (const Vector & _v) const [inline]

Dodaje wartość dwóch wektorów

Parametry

in	_v	- inny wektor

Zwraca

wektor będący sumą

Definicja w linii 99 pliku simulation.cpp.

Oto graf wywołań dla tej funkcji:

7.9.3.17 Vector& Vector::operator+= (const Vector & _v) [inline]

Dodaje wartość innego wektora do klasy

Parametry

in	_v	- inny wektor

Zwraca

referencja na pierwotny obiekt

Definicja w linii 49 pliku vector.hh.

7.9.3.18 Vector& Vector::operator+= (const Vector & _v) [inline]

Dodaje wartość innego wektora do klasy

Parametry

in	_v	- inny wektor

Zwraca

referencja na pierwotny obiekt

Definicja w linii 75 pliku simulation.cpp.

7.9.3.19 Vector Vector::operator- (const Vector & _v) const [inline]

Odejmuje wartość innego wektora od klasy

Parametry

in	_v	- inny wektor

Zwraca

wektor będący różnicą

Definicja w linii 81 pliku vector.hh.

Oto graf wywołań dla tej funkcji:

7.9.3.20 Vector Vector::operator- (const Vector & _v) const [inline]

Odejmuje wartość innego wektora od klasy

Parametry

in	_v	- inny wektor

Zwraca

wektor będący różnicą

Definicja w linii 107 pliku simulation.cpp.

Oto graf wywołań dla tej funkcji:

7.9.3.21 Vector& Vector::operator-= (const Vector & _v) [inline]

Odejmuje wartość innego wektora od klasy

Parametry

in	_v	- inny wektor

Zwraca

referencja na pierwotny obiekt

Definicja w linii 57 pliku vector.hh.

7.9.3.22 Vector& Vector::operator-= (const Vector & _v) [inline]

Odejmuje wartość innego wektora od klasy

Parametry

in	_v	- inny wektor

Zwraca

referencja na pierwotny obiekt

Definicja w linii 83 pliku simulation.cpp.

7.9.3.23 Vector& Vector::operator= (const Vector & _v) [inline]

Przypisuje współrzędne innego wektora do klasy

Parametry

in	_v	- inny wektor

Zwraca

referencja na pierwotny obiekt

Definicja w linii 97 pliku vector.hh.

7.9.3.24 Vector& Vector::operator= (const Vector & _v) [inline]

Przypisuje współrzędne innego wektora do klasy

Parametry

in	_v	- inny wektor

Zwraca

referencja na pierwotny obiekt

Definicja w linii 123 pliku simulation.cpp.

7.9.4 Dokumentacja przyjaciół i funkcji związanych

7.9.4.1 std::ostream& operator<< (std::ostream & _os, const Vector & _s) [friend]

Definicja w linii 133 pliku simulation.cpp.

7.9.4.2 std::ostream& operator<< (std::ostream & _os, const Vector & _s) [friend]

Definicja w linii 133 pliku simulation.cpp.

7.9.5 Dokumentacja atrybutów składowych

7.9.5.1 float Vector::x [private]

Definicja w linii 11 pliku vector.hh.

7.9.5.2 float Vector::y [private]

Definicja w linii 11 pliku vector.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- vector.hh
- simulation.cpp

7.10 Dokumentacja klasy Zbiornik

Klasa modelująca zbiornik.

`#include <zbiornik.hh>`

Diagram dziedziczenia dla Zbiornik

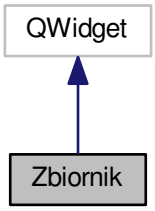
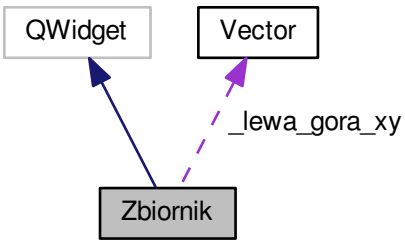


Diagram współpracy dla Zbiornik:



Sloty publiczne

- void GdyOdpowiedniCzas ()

  Slot odpowiadajacy za aktualizacje danych. .

Sygnały

- void ZglosNapis (const QString &)

  Sygnal zglaszajacy napis.
- void ZglosLiczbeCzasteczek (const int)

  Sygnal zglaszajacy liczbe czasteczek.
- void ZglosCzasSymulacji (const double)

  Sygnal zglaszajacy czas trwania symulacji.

Metody publiczne

- Zbiornik (QWidget *wRodzic, const Vector &lewa_gora_xy, const double podstawa, const double wysokosc, const double grubosc)

  Konstruktor parametryczny.
- virtual void paintEvent (QPaintEvent *event)

  Wirtualna metoda paintEvent wyrysowujaca obiekt na ekranie.
- void RysujZbiornik (QPainter &Rysownik, const int Podstawa, const int Wysokosc, const int Grubosc, const int x, const int y)

  Metoda rysujaca zbiornik.
- void RysujZbiornikZCzasteczkami (QPainter &Rysownik)

  Metoda rysujaca zbiornik wraz z czasteczkami.
- bool CzyWewnatrzZbiornika (const double x, const double y) const

  Metoda sprawdzająca czy punkt znajduje sie wewnatrz zbiornika.
- bool CzyWewnatrzZbiornika (const Vector &xy) const

  Metoda sprawdzająca czy punkt znajduje sie wewnatrz zbiornika.
- bool CzyWewnatrzZbiornika (const Czasteczka &cz) const

  Metoda sprawdzająca czy czasteczka znajduje sie wewnatrz zbiornika.
- Vector lewa_gora_xy () const

  Interfejs pozwalajacy na odczyt prywatnych danych.
- Vector & lewa_gora_xy ()

  Interfejs pozwalajacy na zmiane prywatnych danych.
- double podstawa () const

  Interfejs pozwalajacy na odczyt prywatnych danych.
- double & podstawa ()

  Interfejs pozwalajacy na zmiane prywatnych danych.
- double wysokosc () const

  Interfejs pozwalajacy na odczyt prywatnych danych.
- double & wysokosc ()

  Interfejs pozwalajacy na zmiane prywatnych danych.
- double grubosc () const

  Interfejs pozwalajacy na odczyt prywatnych danych.
- double & grubosc ()

  Interfejs pozwalajacy na zmiane prywatnych danych.
- double czas_sym () const

  Interfejs pozwalajacy na odczyt prywatnych danych.
- double & czas_sym ()

  Interfejs pozwalajacy na zmiane prywatnych danych.
- int odpowiedni_czas () const

  Interfejs pozwalajacy na odczyt prywatnych danych.
- int & odpowiedni_czas ()

  Interfejs pozwalajacy na zmiane prywatnych danych.

Atrybuty publiczne

- std::list< Czasteczka > Czasteczki

  Lista wszystkich czasteczek.
- QTimer _Stoper

  Miernik czasu dla zbiornika.

Atrybuty prywatne

- Vector _lewa_gora_xy

  Wektor polozenia lewego gornego punktu zbiornika.
- double _podstawa

  Dlugosc podstawy zbiornika.
- double _wysokosc

  Wysokosc zbiornika.
- double _grubosc

  Grubosc sciany zbiornika.
- double _czas_sym

  Miernik czasu dla zbiornika.
- int _odpowiedni_czas

  Interwal dla timeout'ow z timera [ms].

7.10.1 Opis szczegółowy

Dzieki tej klasie mozliwe jest wyrysowywanie na ekranie zbiornika.

Definicja w linii 45 pliku zbiornik.hh.

7.10.2 Dokumentacja konstruktora i destruktora

7.10.2.1 Zbiornik::Zbiornik (QWidget * *wRodzic*, const Vector & *lewa_gora_xy*, const double *podstawa*, const double *wysokosc*, const double *grubosc*)

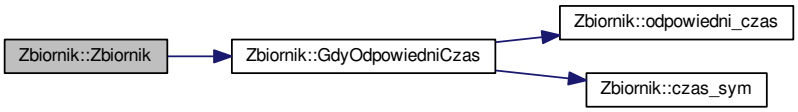
Konstruktor parametryczny.

Parametry

in,out	*wRodzic*	- wskaznik na rodzica
in	*lewa_gora_xy*	- polozenie lewego gornego rogu zbiornika
in	*podstawa*	- dlugosc podstawy zbiornika
in	*wysokosc*	- wysokosc zbiornika
in	*grubosc*	- grubosc sciany zbiornika

Definicja w linii 23 pliku zbiornik.cpp.

Oto graf wywołań dla tej funkcji:



7.10.3 Dokumentacja funkcji składowych

7.10.3.1 double Zbiornik::czas_sym ( ) const [inline]

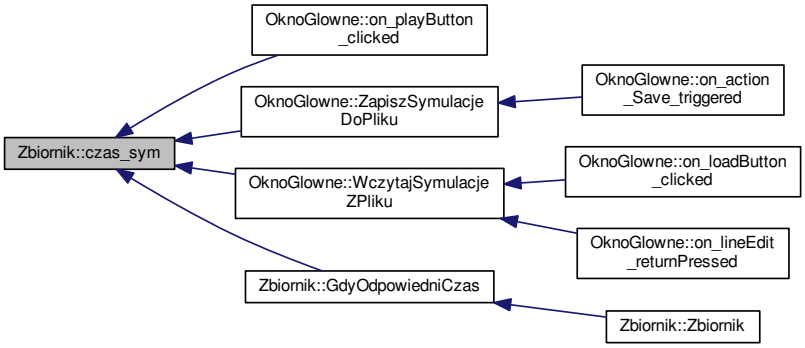
Interfejs pozwalajacy na odczyt prywatnych danych.

Zwraca

_czas_sym - prywatny atrybut opisujacy pomiar czasu

Definicja w linii 191 pliku zbiornik.hh.

Oto graf wywoływań tej funkcji:



7.10.3.2 double& Zbiornik::czas_sym ( ) [inline]

Interfejs pozwalajacy na zmiane prywatnych danych.

Zwraca

_czas_sym - referencja na prywatny atrybut opisujacy pomiar czasu

Definicja w linii 198 pliku zbiornik.hh.

7.10.3.3 bool Zbiornik::CzyWewnatrzZbiornika (const double *x*, const double *y*) const

Sprawdza, czy punkt znajduje sie wewnatrz zbiornika.

Parametry

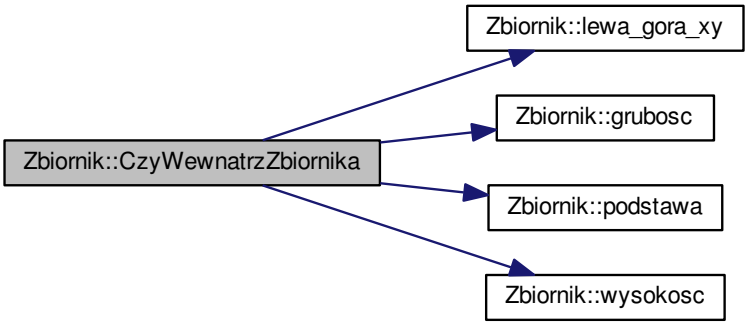
in	*x*	- polozenie punktu na osi X,
in	*y*	- polozenie punktu na osi Y,

Zwraca

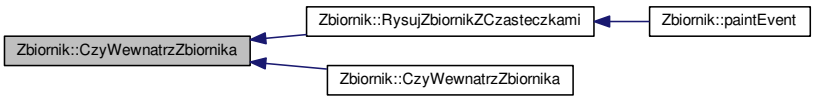
true - jesli znajduje sie wewnatrz zbiornika,
false - jesli nie znajduje sie wewnatrz zbiornika.

Definicja w linii 72 pliku zbiornik.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



7.10.3.4 bool Zbiornik::CzyWewnatrzZbiornika (const Vector & *xy*) const

Sprawdza, czy punkt znajduje sie wewnatrz zbiornika.

Parametry

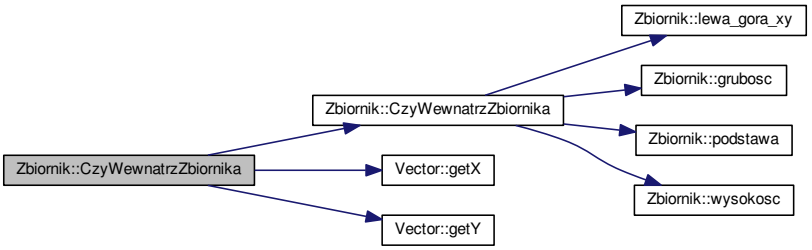
in	*xy*	- polozenie punktu

Zwraca

true - jesli znajduje sie wewnatrz zbiornika,
false - jesli nie znajduje sie wewnatrz zbiornika.

Definicja w linii 81 pliku zbiornik.cpp.

Oto graf wywołań dla tej funkcji:



7.10.3.5 bool Zbiornik::CzyWewnatrzZbiornika (const Czasteczka & *cz*) const

Sprawdza, czy czasteczka znajduje sie wewnatrz zbiornika.

Parametry

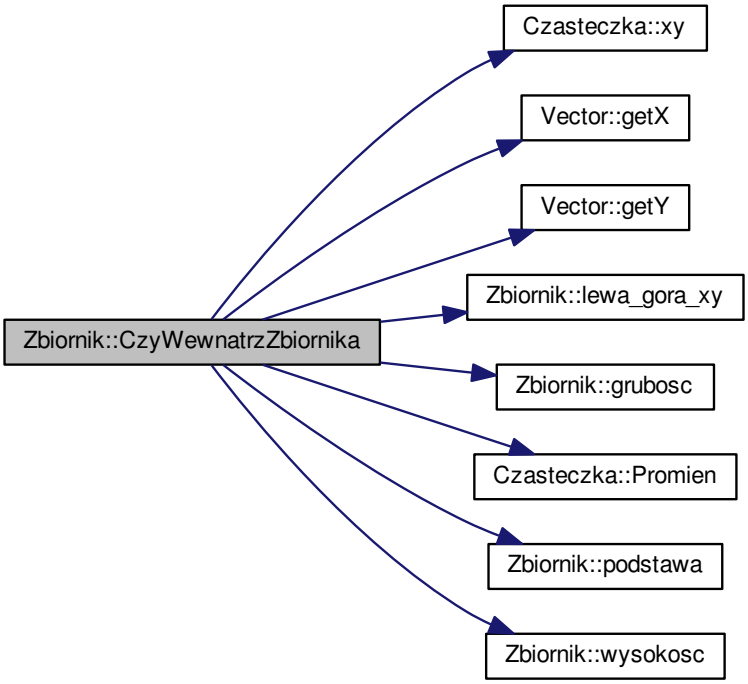
in	*cz*	- czasteczka do sprawdzenia

Zwraca

true - jesli znajduje sie wewnatrz zbiornika,
false - jesli nie znajduje sie wewnatrz zbiornika.

Definicja w linii 89 pliku zbiornik.cpp.

Oto graf wywołań dla tej funkcji:

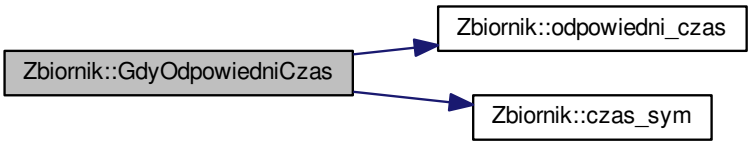


7.10.3.6 void Zbiornik::GdyOdpowiedniCzas ( ) [slot]

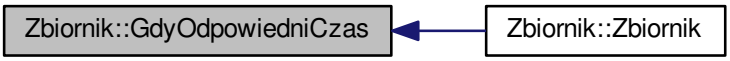
Odpowiada za uaktualnienie zbiornika w odpowiednich momentach.

Definicja w linii 110 pliku zbiornik.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:



7.10.3.7 double Zbiornik::grubosc ( ) const [inline]

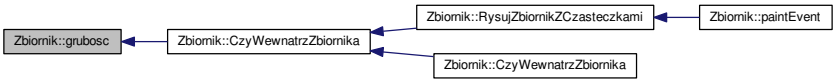
Interfejs pozwalajacy na odczyt prywatnych danych.

Zwraca

_grubosc - prywatny atrybut opisujacy grubosc sciany

Definicja w linii 176 pliku zbiornik.hh.

Oto graf wywoływań tej funkcji:



7.10.3.8 double& Zbiornik::grubosc ( ) [inline]

Interfejs pozwalajacy na zmiane prywatnych danych.

Zwraca

_grubosc - referencja na prywatny atrybut opisujacy grubosc sciany

Definicja w linii 183 pliku zbiornik.hh.

7.10.3.9 Vector Zbiornik::lewa_gora_xy ( ) const [inline]

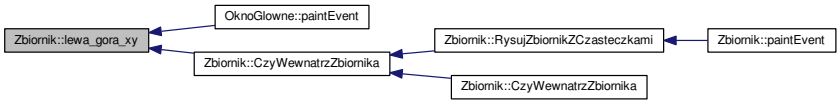
Interfejs pozwalajacy na odczyt prywatnych danych.

Zwraca

_lewa_gora_xy - prywatny atrybut opisujacy wektor polozenia lewego gornego punktu zbiornika

Definicja w linii 134 pliku zbiornik.hh.

Oto graf wywoływań tej funkcji:



7.10.3.10 Vector& Zbiornik::lewa_gora_xy ( ) [inline]

Interfejs pozwalajacy na zmiane prywatnych danych.

Zwraca

_podstawa - referencja na prywatny atrybut opisujacy wektor polozenia lewego gornego punktu zbiornika

Definicja w linii 141 pliku zbiornik.hh.

7.10.3.11 int Zbiornik::odpowiedni_czas ( ) const [inline]

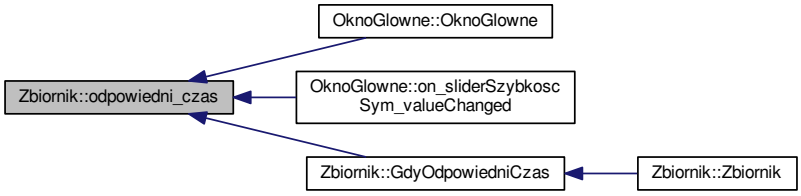
Interfejs pozwalajacy na odczyt prywatnych danych.

Zwraca

odpowiedni_czas - prywatny atrybut opisujacy czas trwania symulacji

Definicja w linii 206 pliku zbiornik.hh.

Oto graf wywoływań tej funkcji:



7.10.3.12 int& Zbiornik::odpowiedni_czas ( ) [inline]

Interfejs pozwalajacy na zmiane prywatnych danych.

Zwraca

odpowiedni_czas - referencja na prywatny atrybut opisujacy czas trwania symulacji

Definicja w linii 213 pliku zbiornik.hh.

7.10.3.13 void Zbiornik::paintEvent (QPaintEvent * *event*) [virtual]

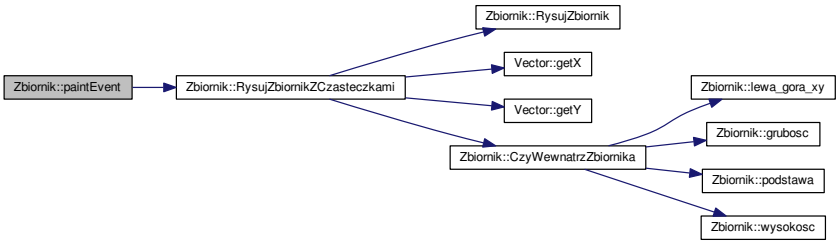
Odziedziczona wirtualna metoda paintEvent.

Parametry

in,out	*event*	- wskaznik obiekt klasy QPaintEvent

Definicja w linii 104 pliku zbiornik.cpp.

Oto graf wywołań dla tej funkcji:



7.10.3.14 double Zbiornik::podstawa ( ) const [inline]

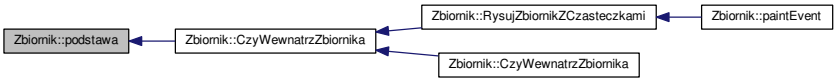
Interfejs pozwalajacy na odczyt prywatnych danych.

Zwraca

_podstawa - prywatny atrybut opisujacy dlugosc podstaw

Definicja w linii 148 pliku zbiornik.hh.

Oto graf wywoływań tej funkcji:



7.10.3.15 double& Zbiornik::podstawa ( ) [inline]

Interfejs pozwalajacy na zmiane prywatnych danych.

Zwraca

_podstawa - referencja na prywatny atrybut opisujacy dlugosc podstawy

Definicja w linii 155 pliku zbiornik.hh.

7.10.3.16 void Zbiornik::RysujZbiornik (QPainter & *Rysownik,* const int *Podstawa,* const int *Wysokosc,* const int *Grubosc,* const int *x,* const int *y*)

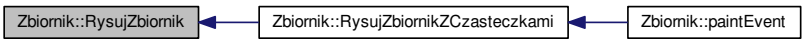
Rysuje zbiornik o zadanych parametrach.

Parametry

in,out	*Rysownik*	- referencja na obiekt klasy QPainter
in	*Podstawa*	- dlugosc podstawy zbiornika
in	*Wysokosc*	- wysokosc zbiornika
in	*Grubosc*	- grubosc wyrysowywanego odcinka
in	*x*	- polozenie lewego gornego punktu zbiornika na osi x
in	*y*	- polozenie lewego gornego punktu zbiornika na osi y

Definicja w linii 40 pliku zbiornik.cpp.

Oto graf wywoływań tej funkcji:



7.10.3.17 void Zbiornik::RysujZbiornikZCzasteczkami (QPainter & *Rysownik*)

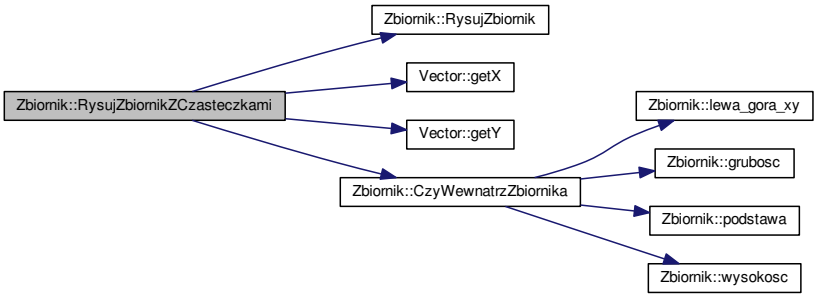
Rysuje zbiornik oraz czasteczki.

Parametry

in,out	*Rysownik*	- referencja na obiekt klasy QPainter

Definicja w linii 56 pliku zbiornik.cpp.

Oto graf wywołań dla tej funkcji:



Oto graf wywoływań tej funkcji:

7.10.3.18 double Zbiornik::wysokosc ( ) const [inline]

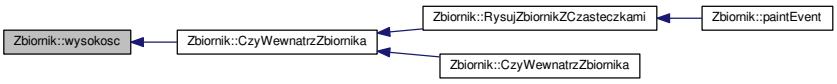
Interfejs pozwalajacy na odczyt prywatnych danych.

Zwraca

_podstawa - prywatny atrybut opisujacy wysokosc

Definicja w linii 162 pliku zbiornik.hh.

Oto graf wywoływań tej funkcji:



7.10.3.19 double& Zbiornik::wysokosc ( ) [inline]

Interfejs pozwalajacy na zmiane prywatnych danych.

Zwraca

_podstawa - referencja na prywatny atrybut opisujacy wysokosc

Definicja w linii 169 pliku zbiornik.hh.

7.10.3.20 void Zbiornik::ZglosCzasSymulacji (const double *_t1*) [signal]

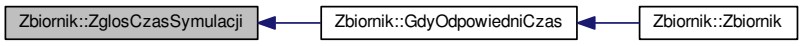
Sygnal zglaszajacy czas trwania symulacji do odpowiedniego slotu.

Parametry

in	*_t1*	- czas do zgloszenia

Definicja w linii 119 pliku moc_zbiornik.cpp.

Oto graf wywoływań tej funkcji:



7.10.3.21 void Zbiornik::ZglosLiczbeCzasteczek (const int *_t1*) [signal]

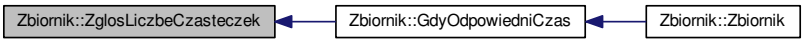
Sygnal zglaszajacy liczbe czasteczek do odpowiedniego slotu.

Parametry

in	*_t1*	- liczba czasteczek do zgloszenia

Definicja w linii 112 pliku moc_zbiornik.cpp.

Oto graf wywoływań tej funkcji:



7.10.3.22 void Zbiornik::ZglosNapis (const QString & *_t1*) [signal]

Sygnal zglaszajacy napis do odpowiedniego slotu.

Parametry

in	*_t1*	- napis do zgloszenia

Definicja w linii 105 pliku moc_zbiornik.cpp.

7.10.4 Dokumentacja atrybutów składowych

7.10.4.1 double Zbiornik::_czas_sym [private]

Miernik czasu zbiornika. Sluzy do mierzenia i wyswietlania czasu.

Definicja w linii 294 pliku zbiornik.hh.

7.10.4.2 double Zbiornik::_grubosc [private]

Grubosc sciany zbiornika.

Definicja w linii 287 pliku zbiornik.hh.

7.10.4.3 Vector Zbiornik::_lewa_gora_xy [private]

Wektor polozenia lewego gornego punktu zbiornika.

Definicja w linii 269 pliku zbiornik.hh.

7.10.4.4 int Zbiornik::_odpowiedni_czas [private]

Interwal dla timeout'ow z timera [ms].

Definicja w linii 300 pliku zbiornik.hh.

7.10.4.5 double Zbiornik::_podstawa [private]

Dlugosc podstawy zbiornika.

Definicja w linii 275 pliku zbiornik.hh.

7.10.4.6 QTimer Zbiornik::_Stoper

Miernik czasu zbiornika. Sluzy do odswiezania ekranu.

Definicja w linii 261 pliku zbiornik.hh.

7.10.4.7 double Zbiornik::_wysokosc [private]

Wysokosc zbiornika.

Definicja w linii 281 pliku zbiornik.hh.

7.10.4.8 std::list<Czasteczka> Zbiornik::Czasteczki

Lista wszystkich czasteczek.

Definicja w linii 254 pliku zbiornik.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- zbiornik.hh
- moc_zbiornik.cpp
- zbiornik.cpp

Rozdział 8

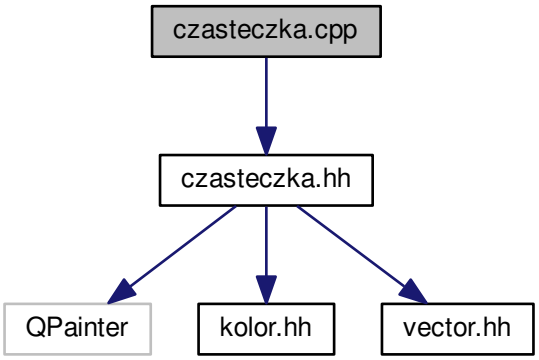
Dokumentacja plików

8.1 Dokumentacja pliku czasteczka.cpp

Zawiera definicje metod klasy Czasteczka.

```
#include "czasteczka.hh"
```

Wykres zależności załączania dla czasteczka.cpp:



8.1.1 Opis szczegółowy

W pliku znajduja sie:

- definicje konstruktorow, metod i przeciazen klasy Czasteczka.

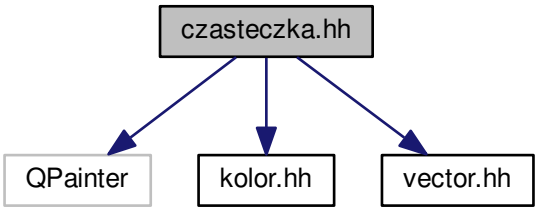
Definicja w pliku czasteczka.cpp.

8.2 Dokumentacja pliku czasteczka.hh

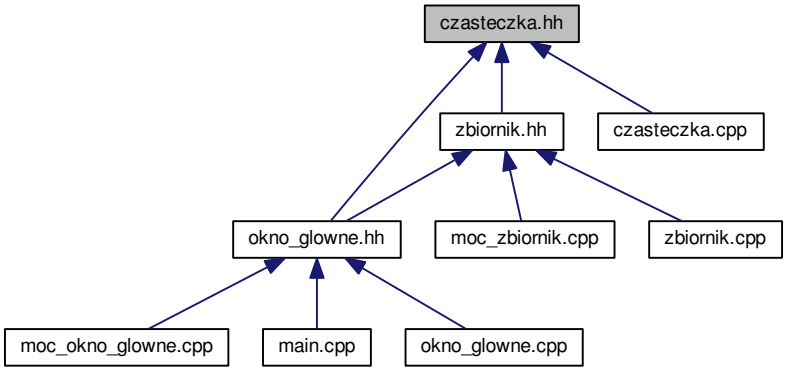
Zawiera definicje klasy Czasteczka oraz deklaracje jej metod.

```
#include <QPainter>
#include "kolor.hh"
#include "vector.hh"
```

Wykres zależności załączania dla czasteczka.hh:



Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Komponenty

- class Czasteczka

  Klasa modelująca czasteczke.

8.2.1 Opis szczegółowy

W pliku znajduja sie:

- definicja klasy Czasteczka (modeluje pojecie czasteczki),
- definicje konstruktorow,
- deklaracje atrybutow.

Definicja w pliku czasteczka.hh.

8.3 Dokumentacja pliku dmainwindow.cpp

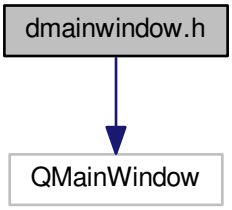
```
#include "dmainwindow.h"
#include "ui_dmainwindow.h"
```

Wykres zależności załączania dla dmainwindow.cpp:

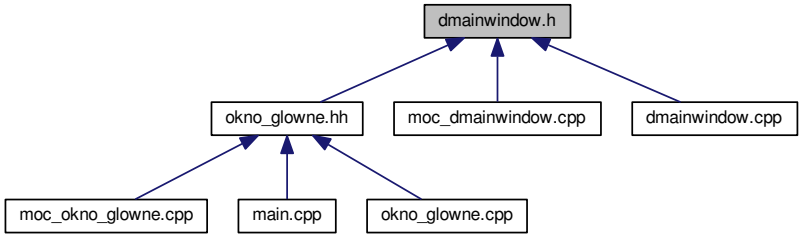
8.4 Dokumentacja pliku dmainwindow.h

```
#include <QMainWindow>
```

Wykres zależności załączania dla dmainwindow.h:



Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Komponenty

- class DMainWindow

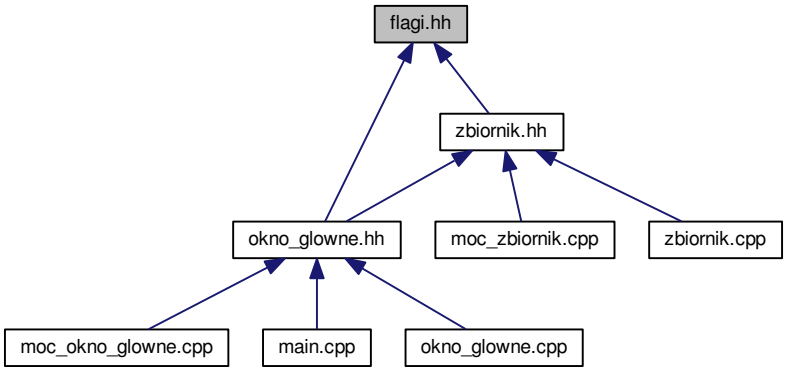
Przestrzenie nazw

- Ui

8.5 Dokumentacja pliku flagi.hh

Zawiera globalne zmienne opisujace symulacje.

Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Wyliczenia

- enum { eSTOP, ePAUSE, ePLAY }

  Typ wyliczeniowy okreslajacy stany symulacji.

Zmienne

- int STAN

  Stan symulacji.
- const int PROMIEN = 5

  Promien czasteczki.
- const int PODSTAWA = 200

  Dlugosc podstawy zbiornika.
- const int WYSOKOSC = 200

  Wysokosc zbiornika.
- const int GRUBOSC = 10

  Grubosc krawedzi zbiornika.

8.5.1 Opis szczegółowy

W pliku znajduja sie:

- zmienne opisujaca stan symulacji,
- poczatkowy promien czasteczek,
- zalozone wymiary zbiornika,

Definicja w pliku flagi.hh.

8.5.2 Dokumentacja typów wyliczanych

8.5.2.1 anonymous enum

Stany symulacji to Play/Pause/Stop.

Wartości wyliczeń

eSTOP

ePAUSE

ePLAY

Definicja w linii 27 pliku flagi.hh.

8.5.3 Dokumentacja zmiennych

8.5.3.1 const int GRUBOSC = 10

Grubosc krawedzi zbiornika. Grubosc jest symetryczna wzgledem osi.

Definicja w linii 59 pliku flagi.hh.

8.5.3.2 const int PODSTAWA = 200

Dlugosc podstawy zbiornika.

Definicja w linii 44 pliku flagi.hh.

8.5.3.3 const int PROMIEN = 5

Promien czasteczki.

Definicja w linii 37 pliku flagi.hh.

8.5.3.4 int STAN

Stan symulacji to Play/Pause/Stop. Inicjalizacja w pliku zbiornik.cpp. Wybor poczatkowego stanu w konstruktorze okienka.

Stan symulacji.

Poczatkowy stan symulacji to Play/Pauza/Stop.

Definicja w linii 21 pliku zbiornik.cpp.

8.5.3.5 const int WYSOKOSC = 200

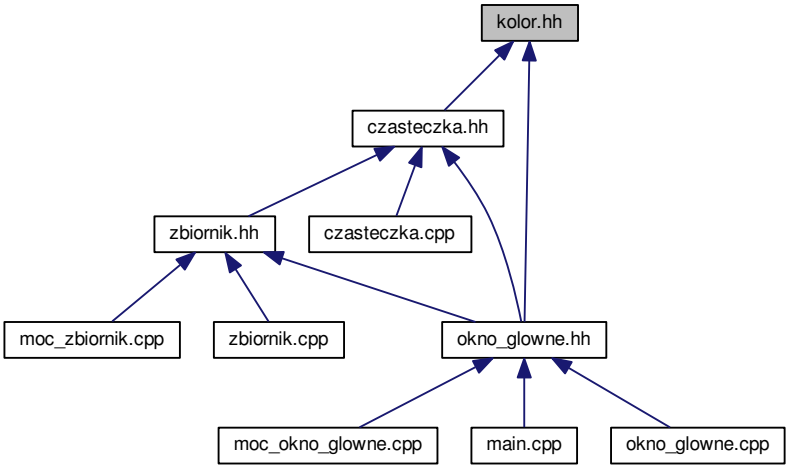
Wysokosc zbiornika.

Definicja w linii 51 pliku flagi.hh.

8.6 Dokumentacja pliku kolor.hh

Zawiera definicje klasy Kolor oraz deklaracje jej metod.

Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Komponenty

- class Kolor

  Klasa modelująca kolor.

8.6.1 Opis szczegółowy

W pliku znajduja sie:

- definicja klasy Kolor (modeluje pojecie Koloru),
- definicje konstruktorow,
- deklaracje atrybutow.

Definicja w pliku kolor.hh.

8.7 Dokumentacja pliku main.cpp

Zawiera ogolna strukture funkcji main.

```
#include <QApplication>
#include <stdlib.h>
#include <time.h>
#include "okno_glowne.hh"
```

Wykres zależności załączania dla main.cpp:

Funkcje

- int main (int argc, char ∗argv[ ])

8.7.1 Opis szczegółowy

W funkcji main wyroznia sie:

- utworznie obiektu klasy QApplication i wstepne przetworzenie argumentow z linii wywolania,
- utworzenie okna aplikacji,
- wymuszenie ukazania sie okna,
- uruchomienie obslugi petli zdarzen dla calej aplikacji.

Definicja w pliku main.cpp.

8.7.2 Dokumentacja funkcji

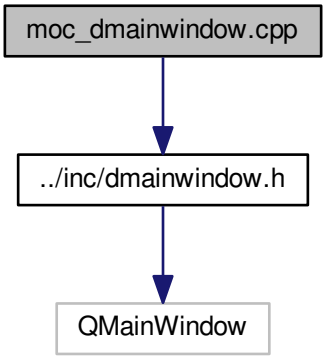
8.7.2.1 int main (int *argc*, char ∗ *argv[ ]*)

Definicja w linii 19 pliku main.cpp.

8.8 Dokumentacja pliku moc_dmainwindow.cpp

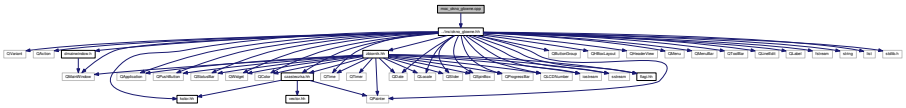
```
#include "../inc/dmainwindow.h"
```

Wykres zależności załączania dla moc_dmainwindow.cpp:



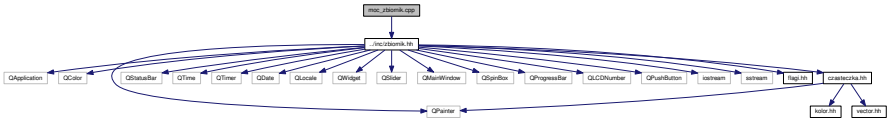
8.9 Dokumentacja pliku moc_okno_glowne.cpp

`#include "../inc/okno_glowne.hh"`
Wykres zależności załączania dla moc_okno_glowne.cpp:



8.10 Dokumentacja pliku moc_zbiornik.cpp

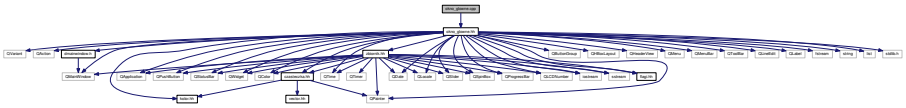
`#include "../inc/zbiornik.hh"`
Wykres zależności załączania dla moc_zbiornik.cpp:



8.11 Dokumentacja pliku okno_glowne.cpp

Zawiera definicje metod klasy OknoGlowne.

`#include "okno_glowne.hh"`
Wykres zależności załączania dla okno_glowne.cpp:



8.11.1 Opis szczegółowy

W pliku znajduja sie:

- definicje konstruktorow, metod i przeciazen klasy OknoGlowne.

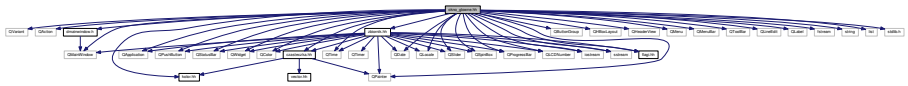
Definicja w pliku okno_glowne.cpp.

8.12 Dokumentacja pliku okno_glowne.hh

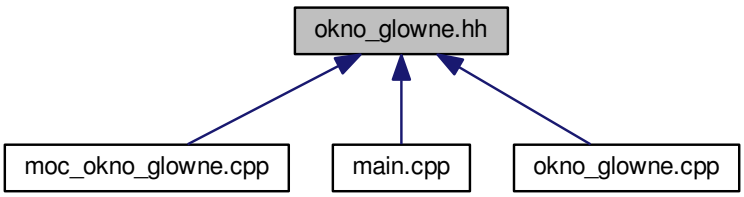
Zawiera definicje klasy OknoGlowne i deklaracje jej metod.

```
#include <QVariant>
#include <QAction>
#include <QApplication>
#include <QButtonGroup>
#include <QHBoxLayout>
#include <QHeaderView>
#include <QMainWindow>
#include <QMenu>
#include <QMenuBar>
#include <QPushButton>
#include <QStatusBar>
#include <QToolBar>
#include <QWidget>
#include <QColor>
#include <QPainter>
#include <QTime>
#include <QDate>
#include <QLocale>
#include <QSlider>
#include <QSpinBox>
#include <QProgressBar>
#include <QLCDNumber>
#include <QLineEdit>
#include <QLabel>
#include <iostream>
#include <sstream>
#include <fstream>
#include <string>
#include <list>
#include <stdlib.h>
#include "flagi.hh"
#include "kolor.hh"
#include "czasteczka.hh"
#include "zbiornik.hh"
#include "dmainwindow.h"
```

Wykres zależności załączania dla okno_glowne.hh:



Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Komponenty

- class OknoGlowne

  Klasa modelujaca głowne okno aplikacji.

8.12.1 Opis szczegółowy

W pliku znajduja sie:

- definicja klasy OknoGlowne (modeluje glowne okno aplikacji),
- deklaracje konstruktorow, metod i przeciazen ww. klasy.

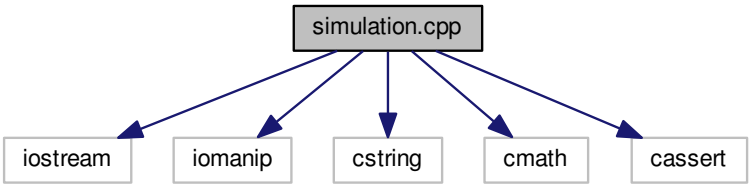
Definicja w pliku okno_glowne.hh.

8.13 Dokumentacja pliku simulation.cpp

Plik z kodem symulatora cieczy.

```
#include <iostream>
#include <iomanip>
#include <cstring>
#include <cmath>
#include <cassert>
```

Wykres zależności załączania dla simulation.cpp:



Komponenty

- class Vector

  klasa Vector
- struct params_t
- class simulation

Definicje

- #define DAMP 0.75
- #define XMAX 2.0
- #define YMAX 2.0
- #define LOG(MSG)

  makro do logowania

Funkcje

- std::ostream & operator<< (std::ostream &_os, const Vector &_s)
- void setup (params_t *params)
- std::ostream & operator<< (std::ostream &_os, const simulation &_s)
- int funkcja_main ()

8.13.1 Opis szczegółowy

Zawiera definicje klas i funkcji pozwalających na przeprowadzenie symulacji zachowania sparametryzowanego modelu cieczy w zadanym środowisku od określonego warunku początkowego.

Definicja w pliku simulation.cpp.

8.13.2 Dokumentacja definicji

8.13.2.1 #define DAMP 0.75

Definicja w linii 6 pliku simulation.cpp.

8.13.2.2 #define LOG(*MSG*)

Wartość:

```
do { std::cerr << "\e[32m" << __FILE__ << "\e[31m:\e[33m" << __LINE__ \
    << "\e[35m:\e[34m" << std::setw(16) << __FUNCTION__ << "\e[36m : \e[37m" << MSG << "\e[0m\n"; } while(
      false )
```

Łatwe w użyciu makro poprawiające czytelność komunikatów. Pętla do...while zastosowana aby wymusić poprawne użycie średnika po poleceniu.

Parametry

in	*MSG*	- wiadomość do wyświetlenia

Definicja w linii 18 pliku simulation.cpp.

8.13.2.3 #define XMAX 2.0

Definicja w linii 7 pliku simulation.cpp.

8.13.2.4 #define YMAX 2.0

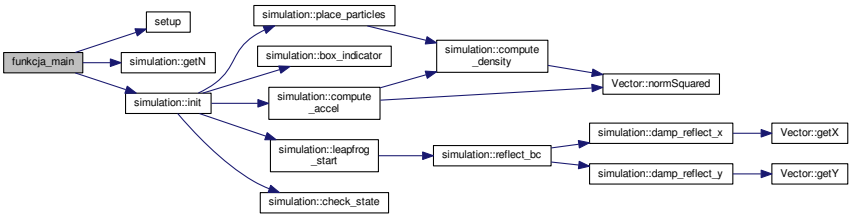
Definicja w linii 8 pliku simulation.cpp.

8.13.3 Dokumentacja funkcji

8.13.3.1 int funkcja_main ( )

Definicja w linii 369 pliku simulation.cpp.

Oto graf wywołań dla tej funkcji:



8.13.3.2 std::ostream& operator<< (std::ostream & *_os*, const Vector & *_s*)

Definicja w linii 133 pliku simulation.cpp.

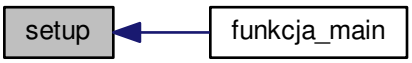
8.13.3.3 std::ostream& operator<< (std::ostream & *_os*, const simulation & *_s*)

Definicja w linii 360 pliku simulation.cpp.

8.13.3.4 void setup (params_t * *params*)

Definicja w linii 150 pliku simulation.cpp.

Oto graf wywoływań tej funkcji:



8.14 Dokumentacja pliku strona.dox

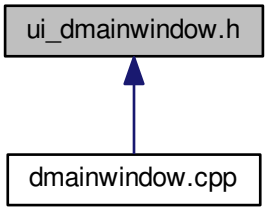
8.15 Dokumentacja pliku ui_dmainwindow.h

```
#include <QtCore/QVariant>
```

```
#include <QtGui/QAction>
#include <QtGui/QApplication>
#include <QtGui/QButtonGroup>
#include <QtGui/QHBoxLayout>
#include <QtGui/QHeaderView>
#include <QtGui/QLCDNumber>
#include <QtGui/QLabel>
#include <QtGui/QLineEdit>
#include <QtGui/QMainWindow>
#include <QtGui/QMenu>
#include <QtGui/QMenuBar>
#include <QtGui/QPushButton>
#include <QtGui/QSlider>
#include <QtGui/QSpacerItem>
#include <QtGui/QStatusBar>
#include <QtGui/QToolBar>
#include <QtGui/QWidget>
```

Wykres zależności załączania dla ui_dmainwindow.h:

Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Komponenty

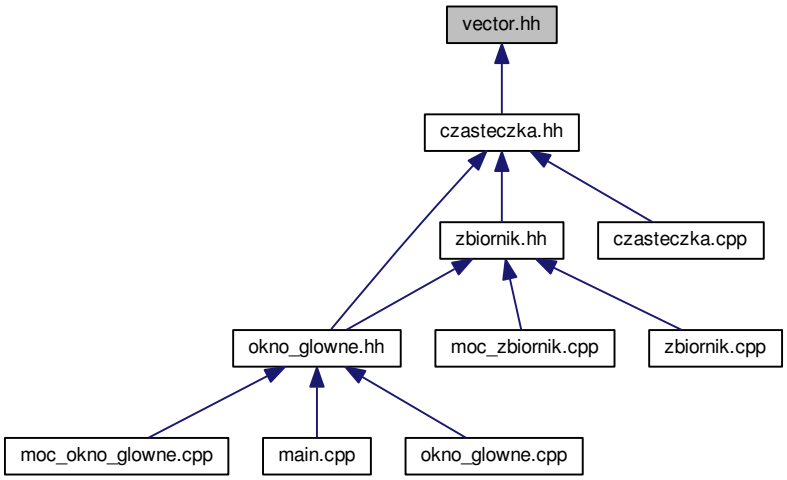
- class Ui_DMainWindow
- class Ui::DMainWindow

Przestrzenie nazw

- Ui

8.16 Dokumentacja pliku vector.hh

Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Komponenty

- class Vector

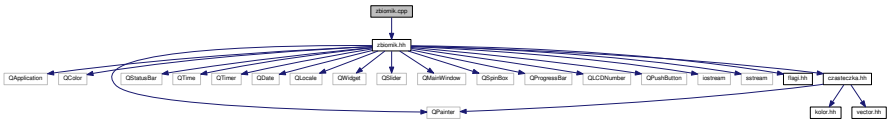
  klasa Vector

8.17 Dokumentacja pliku zbiornik.cpp

Zawiera definicje metod klasy Zbiornik.

`#include "zbiornik.hh"`

Wykres zależności załączania dla zbiornik.cpp:



Zmienne

- int STAN = ePAUSE

  Poczatkowy stan symulacji.

8.17.1 Opis szczegółowy

W pliku znajduja sie:

- definicje konstruktorow, metod i przeciazen klasy Zbiornik.

Definicja w pliku zbiornik.cpp.

8.17.2 Dokumentacja zmiennych

8.17.2.1 int STAN = ePAUSE

Stan symulacji.

Poczatkowy stan symulacji to Play/Pauza/Stop.

Definicja w linii 21 pliku zbiornik.cpp.

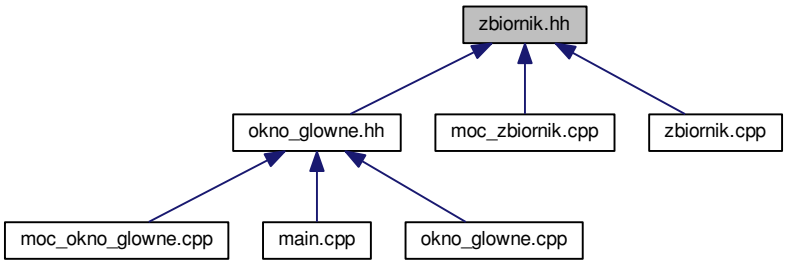
8.18 Dokumentacja pliku zbiornik.hh

Zawiera definicje klasy Zbiornik i deklaracje jej metod.

```
#include <QApplication>
#include <QColor>
#include <QPainter>
#include <QStatusBar>
#include <QTime>
#include <QTimer>
#include <QDate>
#include <QLocale>
#include <QWidget>
#include <QSlider>
#include <QMainWindow>
#include <QSpinBox>
#include <QProgressBar>
#include <QLCDNumber>
#include <QPushButton>
#include <iostream>
#include <sstream>
#include "flagi.hh"
#include "czasteczka.hh"
```

Wykres zależności załączania dla zbiornik.hh:

Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Komponenty

- class Zbiornik

  Klasa modelująca zbiornik.

8.18.1 Opis szczegółowy

W pliku znajduja sie:

- definicja klasy Zbiornik (modeluje pojecie Zbiornika),
- deklaracje konstruktorow, metod i przeciazen ww. klasy.

Definicja w pliku zbiornik.hh.

Skorowidz