Projekt 45- Prosty edytor grafiki 3D

1.0

Wygenerowano przez Doxygen 1.9.1

Rozdział 1

Indeks hierarchiczny

1.1 Hierarchia klas

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

Rozdział 2

Indeks klas

2.1 Lista klas

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

Rozdział 3

Indeks plików

3.1 Lista plików

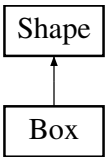
Tutaj znajduje się lista wszystkich udokumentowanych plików z ich krótkimi opisami:

Rozdział 4

Dokumentacja klas

4.1 Dokumentacja klasy Box

Diagram dziedziczenia dla Box



Metody publiczne

- **Box** (unsigned id, double x1, double y1, double z1, double x2, double y2, double z2)

  Tworzy sześcian.
- std::string **toString** () override

  zwraca informacje o obiekcie w formie std::string.

Dodatkowe Dziedziczone Składowe

4.1.1 Dokumentacja konstruktora i destruktora

4.1.1.1 Box()

```
Box::Box (
            unsigned id,
            double x1,
            double y1,
            double z1,
            double x2,
            double y2,
            double z2 )
```

Tworzy sześcian.

Parametry

id	Identyfikator.
x1	Współrzędna x pierwszego narożnika.
y1	Współrzędna y pierwszego narożnika.
z1	Współrzędna z pierwszego narożnika.
x2	Współrzędna x drugiego narożnika.
y2	Współrzędna y drugiego narożnika.
z2	Współrzędna z drugiego narożnika.

4.1.2 Dokumentacja funkcji składowych

4.1.2.1 toString()

```
std::string Box::toString ( )  [override], [virtual]
```

zwraca informacje o obiekcie w formie std::string.

Zwraca

String z informacjami.

Reimplementowana z **Shape** (str. **??**).

Dokumentacja dla tej klasy została wygenerowana z pliku:

- Include/Shape.h

4.2 Dokumentacja struktury Command

Struktura reprezentująca komendę.

```
#include <Command.h>
```

Metody publiczne

- **Command** (std::string n, std::function< void(std::vector< std::string >)> c)
  Konstruktor komendy.

Atrybuty publiczne

- std::string **name**
- std::function< void(std::vector< std::string >)> **command**

4.2.1 Opis szczegółowy

Struktura reprezentująca komendę.

4.2.2 Dokumentacja konstruktora i destruktora

4.2.2.1 Command()

```
Command::Command (
            std::string n,
            std::function< void(std::vector< std::string >)> c )  [inline]
```

Konstruktor komendy.

Parametry

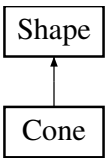
n	Nazwa komendy, używana do wywoływania przez użytkownika.
c	Funkcja wywoływana po użyciu komendy.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- Include/Command.h

4.3 Dokumentacja klasy Cone

Diagram dziedziczenia dla Cone



Metody publiczne

- **Cone** (unsigned id, double x1, double y1, double z1, double r1, double x2, double y2, double z2, double r2, double n)

  Tworzy (ścięty) stożek.
- void **move** (double x, double y, double z) override

  Przesuwa bryłę o wektor [x, y, z].
- void **rotate** (double x, double y, double z, double a, double b, double g) override

  Obraca bryłę wokół punktu (x, y, z) o kąt a wzdłuż osi X, b wzdłuż osi Y i kąt g wzdłuż osi Z.
- std::string **toString** () override

  Zwraca podstawowe informacje o bryle w formacie odpowiednim do wyświetlania na liście brył.

Atrybuty chronione

- **Point m_center1**
- double **m_radius1**
- **Point m_center2**
- double **m_radius2**
- double **m_quads**

Dodatkowe Dziedziczone Składowe

4.3.1 Dokumentacja konstruktora i destruktora

4.3.1.1 Cone()

```
Cone::Cone (
            unsigned id,
            double x1,
            double y1,
            double z1,
            double r1,
            double x2,
            double y2,
            double z2,
            double r2,
            double n )
```

Tworzy (ścięty) stożek.

Parametry

id	Identyfikator.
x1	Współrzędna x pierwszej podstawy.
y1	Współrzędna y pierwszej podstawy.
z1	Współrzędna z pierwszej podstawy.
r1	Promień dolnej podstawy.
x2	Współrzędna x drugiej podstawy.
y2	Współrzędna y drugiej podstawy.
z2	Współrzędna z drugiej podstawy.
r2	Promień górnej podstawy.
n	"Promień" do użycia w algorytmie generowania okręgu w podstawie.

4.3.2 Dokumentacja funkcji składowych

4.3.2.1 move()

```
void Cone::move (
            double x,
            double y,
            double z )  [override], [virtual]
```

Przesuwa bryłę o wektor [x, y, z].

Parametry

x	Wartość x wektora przesunięcia.
y	Wartość y wektora przesunięcia.
z	Wartość z wektora przesunięcia.

Reimplementowana z **Shape** (str. **??**).

4.3.2.2 rotate()

```
void Cone::rotate (
            double x,
            double y,
            double z,
            double a,
            double b,
            double g )  [override], [virtual]
```

Obraca bryłę wokół punktu (x, y, z) o kąt a wzdłuż osi X, b wzdłuż osi Y i kąt g wzdłuż osi Z.

Parametry

x	Współrzędna x środka obrotu.
y	Współrzędna y środka obrotu.
z	Współrzędna z środka obrotu.
a	Kąt obrotu wokół osi X.
b	Kąt obrotu wokół osi Y.
g	Kąt obrotu wokół osi Z.

Reimplementowana z **Shape** (str. **??**).

4.3.2.3 toString()

```
std::string Cone::toString ( )  [override], [virtual]
```

Zwraca podstawowe informacje o bryle w formacie odpowiednim do wyświetlania na liście brył.

Zwraca

Łańcuch znaków z informacjami o bryle.

Reimplementowana z **Shape** (str. **??**).

Dokumentacja dla tej klasy została wygenerowana z pliku:

- Include/Shape.h

4.4 Dokumentacja klasy Console

Klasa reprezentująca konsolę.

`#include <Console.h>`

Metody publiczne

- **Console** (wxTextCtrl ∗textCtrl)
  Konstruktor konsoli.
- void **registerCommand** (**Command** command)
  Dodaje komendę do obsługi przez konsolę.
- void **sendCommand** (std::string commandText)
  Wysyła komendę do konsoli.
- std::vector< std::any > **validateArguments** (std::vector< std::string > arguments, std::initializer_list< std↩ ::string > argumentTypes)
  Sprawdza poprawność wpisanej komendy.
- void **print** (std::string text, const wxColour ∗color=wxWHITE)
  Drukuje tekst w kontrolce tekstu.

Atrybuty chronione

- wxTextCtrl ∗ **_textCtrl**
- std::ostream ∗ **_stream**
- std::vector< **Command** > **_commands**

4.4.1 Opis szczegółowy

Klasa reprezentująca konsolę.

4.4.2 Dokumentacja konstruktora i destruktora

4.4.2.1 Console()

```
Console::Console (
            wxTextCtrl * textCtrl )
```

Konstruktor konsoli.

Parametry

textCtrl	Kontrolka w której będzie wypisywany tekst wysyłany przez konsolę. Powinna posiadać styl wxTE_MULTILINE, wxTE_READONLY i wxTE_RICH lub wxTE_RICH2 (do obsługi kolorowego tekstu). Kolor jej tła będzie automatycznie zmieniony na czarny.

4.4.3 Dokumentacja funkcji składowych

4.4.3.1 print()

```
void Console::print (
            std::string text,
            const wxColour * color = wxWHITE )
```

Drukuje tekst w kontrolce tekstu.

Parametry

text	Tekst do wydrukowania.
color	Kolor drukowanego tekstu, domyślnie biały.

4.4.3.2 registerCommand()

```
void Console::registerCommand (
             Command command )
```

Dodaje komendę do obsługi przez konsolę.

Parametry

command	Dodawana komenda. Zobacz **Command::Command** (str. **??**).

4.4.3.3 sendCommand()

```
void Console::sendCommand (
            std::string commandText )
```

Wysyła komendę do konsoli.

Metoda wysyłająca komendę do konsoli. Komenda powinna zostać pobrana samodzielnie przez aplikację i wysłana w formie tekstowej tą metodą.

Parametry

commandText	Tekst wysyłanej komendy. Powinien być całą wysyłaną komendą w formie czystego tekstu zawierającego zarówno nazwę jak i argumenty komendy.

4.4.3.4 validateArguments()

```
std::vector<std::any> Console::validateArguments (
            std::vector< std::string > arguments,
            std::initializer_list< std::string > argumentTypes )
```

Sprawdza poprawność wpisanej komendy.

Metoda sprawdzająca poprawność argumentów. Powinna być wywoływana na początku wywoływanej funkcji.

Parametry

arguments	Lista argumentów do weryfikacji. Powinien być tutaj bezpośrednio przekazany argument wywoływanej funkcji (porównaj **Command::Command** (str. **??**)).
argumentTypes	Rodzaje argumentów przyjmowanych przez funkcję. Każdy element listy reprezentuje jeden argument i przyjmuje wartość "number" (**Number** (str. **??**)), "pair" (**Pair** (str. **??**)), "triplet" (Tripler) lub "name" (**Name** (str. **??**)).

Zwraca

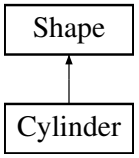
Zwraca wektor łańcuch znaków taki, że każdy element jest argumentem wyłuskanym z tekstu, w formie **Number** (str. **??**), **Pair** (str. **??**), **Triplet** (str. **??**) lub **Name** (str. **??**) (analogicznie do parametru argumentTypes).

Dokumentacja dla tej klasy została wygenerowana z pliku:

- Include/Console.h

4.5 Dokumentacja klasy Cylinder

Diagram dziedziczenia dla Cylinder



Metody publiczne

- **Cylinder** (unsigned id, double x1, double y1, double z1, double x2, double y2, double z2, double r, double n)

  Tworzy cylinder.
- void **move** (double x, double y, double z) override

  Przesuwa bryłę o wektor [x, y, z].
- void **rotate** (double x, double y, double z, double a, double b, double g) override

  Obraca bryłę wokół punktu (x, y, z) o kąt a wzdłuż osi X, b wzdłuż osi Y i kąt g wzdłuż osi Z.
- std::string **toString** () override

  Zwraca podstawowe informacje o bryle w formacie odpowiednim do wyświetlania na liście brył.

Atrybuty chronione

- **Point m_center1**
- **Point m_center2**
- double **m_radius**
- double **m_quads**

Dodatkowe Dziedziczone Składowe

4.5.1 Dokumentacja konstruktora i destruktora

4.5.1.1 Cylinder()

```
Cylinder::Cylinder (
            unsigned id,
            double x1,
            double y1,
            double z1,
            double x2,
            double y2,
            double z2,
            double r,
            double n )
```

Tworzy cylinder.

Parametry

id	Identyfikator.
x1	Współrzędna x pierwszej podstawy.
y1	Współrzędna y pierwszej podstawy.
z1	Współrzędna z pierwszej podstawy.
x2	Współrzędna x drugiej podstawy.
y2	Współrzędna y drugiej podstawy.
z2	Współrzędna z drugiej podstawy.
r	Promień podstaw.
n	"Promień" do użycia w algorytmie generowania okręgu w podstawie.

4.5.2 Dokumentacja funkcji składowych

4.5.2.1 move()

```
void Cylinder::move (
            double x,
            double y,
            double z )  [override], [virtual]
```

Przesuwa bryłę o wektor [x, y, z].

Parametry

x	Wartość x wektora przesunięcia.
y	Wartość y wektora przesunięcia.
z	Wartość z wektora przesunięcia.

Reimplementowana z **Shape** (str. **??**).

4.5.2.2 rotate()

```
void Cylinder::rotate (
            double x,
            double y,
            double z,
            double a,
            double b,
            double g )  [override], [virtual]
```

Obraca bryłę wokół punktu (x, y, z) o kąt a wzdłuż osi X, b wzdłuż osi Y i kąt g wzdłuż osi Z.

Parametry

x	Współrzędna x środka obrotu.
y	Współrzędna y środka obrotu.
z	Współrzędna z środka obrotu.
a	Kąt obrotu wokół osi X.
b	Kąt obrotu wokół osi Y.
g	Kąt obrotu wokół osi Z.

Reimplementowana z **Shape** (str. **??**).

4.5.2.3 toString()

```
std::string Cylinder::toString ( )  [override], [virtual]
```

Zwraca podstawowe informacje o bryle w formacie odpowiednim do wyświetlania na liście brył.

Zwraca

Łańcuch znaków z informacjami o bryle.

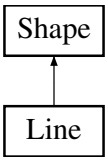
Reimplementowana z **Shape** (str. **??**).

Dokumentacja dla tej klasy została wygenerowana z pliku:

- Include/Shape.h

4.6 Dokumentacja klasy Line

Diagram dziedziczenia dla Line



Metody publiczne

- **Line** (unsigned id, double x1, double y1, double z1, double x2, double y2, double z2)
  Tworzy odcinek.
- std::string **toString** () override
  zwraca informacje o obiekcie w formie std::string.

Dodatkowe Dziedziczone Składowe

4.6.1 Dokumentacja konstruktora i destruktora

4.6.1.1 Line()

```
Line::Line (
            unsigned id,
            double x1,
            double y1,
            double z1,
            double x2,
            double y2,
            double z2 )
```

Tworzy odcinek.

Parametry

id	Identyfikator.
x1	Współrzędna x pierwszego końca.
y1	Współrzędna y pierwszego końca.
z1	Współrzędna z pierwszego końca.
x2	Współrzędna x drugiego końca.
y2	Współrzędna y drugiego końca.
z2	Współrzędna z drugiego końca.

4.6.2 Dokumentacja funkcji składowych

4.6.2.1 toString()

```
std::string Line::toString ( )  [override], [virtual]
```

zwraca informacje o obiekcie w formie std::string.

Zwraca

String z informacjami.

Reimplementowana z **Shape** (str. **??**).

Dokumentacja dla tej klasy została wygenerowana z pliku:

- Include/Shape.h

4.7 Dokumentacja struktury ListObject

Struktura reprezentująca element listy brył do wyświetlenia.

```
#include <ListObject.h>
```

Metody publiczne

- **ListObject** (unsigned i, wxStaticText ∗st)

Atrybuty publiczne

- unsigned **id**
- wxStaticText ∗ **staticText**

4.7.1 Opis szczegółowy

Struktura reprezentująca element listy brył do wyświetlenia.

Dokumentacja dla tej struktury została wygenerowana z pliku:

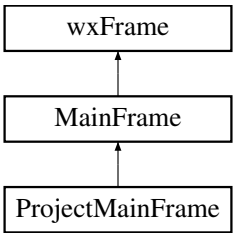
- Include/ListObject.h

4.8 Dokumentacja klasy MainFrame

Class **MainFrame** (str. **??**).

```
#include <Project.h>
```

Diagram dziedziczenia dla MainFrame



Metody publiczne

- **MainFrame** (wxWindow ∗parent, wxWindowID id=wxID_ANY, const wxString &title=wxT("Prosty edytor grafiki 3D"), const wxPoint &pos=wxDefaultPosition, const wxSize &size=wxSize(640, 480), long style=wx↩ DEFAULT_FRAME_STYLE|wxTAB_TRAVERSAL)

Metody chronione

- virtual void **MainFrameOnClose** (wxCloseEvent &event)
- virtual void **MainFrameOnPaint** (wxPaintEvent &event)
- virtual void **_promptFieldOnTextEnter** (wxCommandEvent &event)

Atrybuty chronione

- wxPanel ∗ **_viewTop**
- wxPanel ∗ **_viewFront**
- wxPanel ∗ **_viewPerspective**
- wxPanel ∗ **_viewRight**
- wxTextCtrl ∗ **_shapeList**
- wxTextCtrl ∗ **_consoleOutput**
- wxStaticText ∗ **_promptChar**
- wxTextCtrl ∗ **_promptField**

4.8.1 Opis szczegółowy

Class **MainFrame** (str. **??**).

Dokumentacja dla tej klasy została wygenerowana z pliku:

- Include/Project.h

4.9 Dokumentacja struktury Name

Argument konsoli zawierający łańcuch znaków.

`#include <ArgumentTypes.h>`

Metody publiczne

- **Name** (std::string v)

Atrybuty publiczne

- std::string **value**

4.9.1 Opis szczegółowy

Argument konsoli zawierający łańcuch znaków.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- Include/ArgumentTypes.h

4.10 Dokumentacja struktury Number

Argument konsoli zawierający pojedynczą liczbę.

`#include <ArgumentTypes.h>`

Metody publiczne

- **Number** (double v)

Atrybuty publiczne

- double **value**

4.10.1 Opis szczegółowy

Argument konsoli zawierający pojedynczą liczbę.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- Include/ArgumentTypes.h

4.11 Dokumentacja struktury Pair

Argument konsoli zawierający parę liczb.

`#include <ArgumentTypes.h>`

Metody publiczne

- **Pair** (double v1, double v2)

Atrybuty publiczne

- double **value1**
- double **value2**

4.11.1 Opis szczegółowy

Argument konsoli zawierający parę liczb.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- Include/ArgumentTypes.h

4.12 Dokumentacja struktury Point

Struktura reprezentująca punkt w przestrzeni trójwymiarowej.

`#include <Segment.h>`

Metody publiczne

- **Point** (double vx, double vy, double vz)

Atrybuty publiczne

- double **x**
- double **y**
- double **z**

4.12.1 Opis szczegółowy

Struktura reprezentująca punkt w przestrzeni trójwymiarowej.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- Include/Segment.h

4.13 Dokumentacja klasy Presentation

Klasa wirtualna reprezentująca kształt, jej potomkami są konkretne bryły.

`#include <Presentation.h>`

Statyczne metody publiczne

- static void **drawTop** (wxClientDC &DC, std::vector< **Shape** * > shapes)
  Rysuje widok z góry, widoczne osie to X i Z, kamera przesuwa się wzdłuż osi Y.
- static void **drawFront** (wxClientDC &DC, std::vector< **Shape** * > shapes)
  Rysuje widok od przodu, widoczne osie to X i Y, kamera przesuwa się wzdłuż osi Z.
- static void **drawRight** (wxClientDC &DC, std::vector< **Shape** * > shapes)
  Rysuje widok z prawej strony, widoczne osie to Y i Z, kamera przesuwa się wzdłuż osi X.
- static void **drawPerspective** (wxClientDC &DC, std::vector< **Shape** * > shapes)
  Rysuje widok perspektywiczny.
- static void **setLineColor** (wxColor color)
  Zmienia kolor linii.
- static void **setFillingColor** (wxColor color)
  Zmienia kolor wypełnienia (niezaimplementowana).
- static void **setTopRange** (double range)
  Ustawia punkt z którego patrzy się górna kamera na (0, `range`, 0).
- static void **setFrontRange** (double range)
  Ustawia punkt z którego patrzy się przednia kamera na (0,0,`range`).
- static void **setRightRange** (double range)
  Ustawia punkt z którego patrzy się prawa kamera na (`range`,0,0).
- static void **setCamera** (double x, double y, double z)
  Ustawia położenie kamery perspektywicznej.
- static void **setFov** (double fov)
  Ustawia pole widzenia kamery perspektywicznej.
- static wxColor **getLineColor** ()
  Zwraca kolor linii.
- static wxColor **getFillingColor** ()
  Zwraca kolor wypełnienia (niezaimplementowana).

- static double **getTopRange** ()

  Zwraca punkt patrzenia górnej kamery.
- static double **getFrontRange** ()

  Zwraca punkt patrzenia przedniej kamery.
- static double **getRightRange** ()

  Zwraca punkt patrzenia prawej kamery.
- static double **getCameraX** ()

  Zwraca współrzędną x kamery perspektywicznej.
- static double **getCameraY** ()

  Zwraca współrzędną y kamery perspektywicznej.
- static double **getCameraZ** ()

  Zwraca współrzędną z kamery perspektywicznej.
- static double **getFov** ()

  Zwraca pole widzenia kamery perspektywicznej.

Statyczne atrybuty publiczne

- static double **debug_scale**
- static bool **debug_enableCulling**

Statyczne atrybuty chronione

- static wxColor **m_lineColor**
- static wxColor **m_fillingColor**
- static double **m_topRange**
- static double **m_frontRange**
- static double **m_rightRange**
- static double **m_topScale**
- static double **m_frontScale**
- static double **m_rightScale**
- static double **m_cameraX**
- static double **m_cameraY**
- static double **m_cameraZ**
- static double **m_fov**

4.13.1 Opis szczegółowy

Klasa wirtualna reprezentująca kształt, jej potomkami są konkretne bryły.

4.13.2 Dokumentacja funkcji składowych

4.13.2.1 drawFront()

```
static void Presentation::drawFront (
            wxClientDC & DC,
            std::vector< Shape * > shapes )  [static]
```

Rysuje widok od przodu, widoczne osie to X i Y, kamera przesuwa się wzdłuż osi Z.

Parametry

DC	Wskaźnik na powierzchnię po której będą rysowane bryły.
shapes	Kontener brył do narysowania.

4.13.2.2 drawPerspective()

```
static void Presentation::drawPerspective (
            wxClientDC & DC,
            std::vector< Shape * > shapes )  [static]
```

Rysuje widok perspektywiczny.

Parametry

DC	Wskaźnik na powierzchnię po której będą rysowane bryły.
shapes	Kontener brył do narysowania.

4.13.2.3 drawRight()

```
static void Presentation::drawRight (
            wxClientDC & DC,
            std::vector< Shape * > shapes )  [static]
```

Rysuje widok z prawej strony, widoczne osie to Y i Z, kamera przesuwa się wzdłuż osi X.

Parametry

DC	Wskaźnik na powierzchnię po której będą rysowane bryły.
shapes	Kontener brył do narysowania.

4.13.2.4 drawTop()

```
static void Presentation::drawTop (
            wxClientDC & DC,
            std::vector< Shape * > shapes )  [static]
```

Rysuje widok z góry, widoczne osie to X i Z, kamera przesuwa się wzdłuż osi Y.

Parametry

DC	Wskaźnik na powierzchnię po której będą rysowane bryły.
shapes	Kontener brył do narysowania.

4.13.2.5 getCameraX()

```
static double Presentation::getCameraX ( )  [inline], [static]
```

Zwraca współrzędną x kamery perspektywicznej.

Zwraca

Współrzędna x.

4.13.2.6 getCameraY()

```
static double Presentation::getCameraY ( )  [inline], [static]
```

Zwraca współrzędną y kamery perspektywicznej.

Zwraca

Współrzędna y.

4.13.2.7 getCameraZ()

```
static double Presentation::getCameraZ ( )  [inline], [static]
```

Zwraca współrzędną z kamery perspektywicznej.

Zwraca

Współrzędna z.

4.13.2.8 getFillingColor()

```
static wxColor Presentation::getFillingColor ( )  [inline], [static]
```

Zwraca kolor wypełnienia (niezaimplementowana).

Zwraca

Kolor wypełnienia.

4.13.2.9 getFov()

```
static double Presentation::getFov ( )  [inline], [static]
```

Zwraca pole widzenia kamery perspektywicznej.

Zwraca

Pole widzenia.

4.13.2.10 getFrontRange()

```
static double Presentation::getFrontRange ( )  [inline], [static]
```

Zwraca punkt patrzenia przedniej kamery.

Zwraca

Punkt patrzenia przedniej kamery.

4.13.2.11 getLineColor()

```
static wxColor Presentation::getLineColor ( )  [inline], [static]
```

Zwraca kolor linii.

Zwraca

Kolor linii.

4.13.2.12 getRightRange()

```
static double Presentation::getRightRange ( )  [inline], [static]
```

Zwraca punkt patrzenia prawej kamery.

Zwraca

Punkt patrzenia prawej kamery.

4.13.2.13 getTopRange()

```
static double Presentation::getTopRange ( )  [inline], [static]
```

Zwraca punkt patrzenia górnej kamery.

Zwraca

Punkt patrzenia górnej kamery.

4.13.2.14 setCamera()

```
static void Presentation::setCamera (
            double x,
            double y,
            double z )  [inline], [static]
```

Ustawia położenie kamery perspektywicznej.

Parametry

x	Współrzędna x.
y	Współrzędna y.
z	Współrzędna z.

4.13.2.15 setFillingColor()

```
static void Presentation::setFillingColor (
            wxColor color )  [inline], [static]
```

Zmienia kolor wypełnienia (niezaimplementowana).

Parametry

color	Kolor wypełnienia.

4.13.2.16 setFov()

```
static void Presentation::setFov (
            double fov )  [inline], [static]
```

Ustawia pole widzenia kamery perspektywicznej.

Parametry

fov	Pole widzenia.

4.13.2.17 setFrontRange()

```
static void Presentation::setFrontRange (
            double range )  [inline], [static]
```

Ustawia punkt z którego patrzy się przednia kamera na (0,0,`range`).

Parametry

range	Współrzędna z punktu.

4.13.2.18 setLineColor()

```
static void Presentation::setLineColor (
            wxColor color )  [inline], [static]
```

Zmienia kolor linii.

Parametry

color	Kolor linii.

4.13.2.19 setRightRange()

```
static void Presentation::setRightRange (
            double range )  [inline], [static]
```

Ustawia punkt z którego patrzy się prawa kamera na (`range`,0,0).

Parametry

range	Współrzędna x punktu

4.13.2.20 setTopRange()

```
static void Presentation::setTopRange (
            double range )  [inline], [static]
```

Ustawia punkt z którego patrzy się górna kamera na (0, range, 0).

Parametry

range	Współrzędna y punktu.

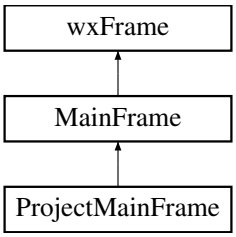
Dokumentacja dla tej klasy została wygenerowana z pliku:

- Include/Presentation.h

4.14 Dokumentacja klasy ProjectMainFrame

```
#include <ProjectMainFrame.h>
```

Diagram dziedziczenia dla ProjectMainFrame



Metody publiczne

- **ProjectMainFrame** (wxWindow ∗parent)
- void **Unimplemented** ()
- void **Draw** ()
- void **SetLineColor** (std::vector< std::string > arguments)
  set_line_color c
- void **CreateLine** (std::vector< std::string > arguments)
  line (x1,y1,z1) (x2,y2,z2)
- void **CreateBox** (std::vector< std::string > arguments)
  box (x1,y2,z1) (x2,y2,z2)
- void **CreateSphere** (std::vector< std::string > arguments)
  sphere (x,y,z) r (n,m)
- void **CreateCone** (std::vector< std::string > arguments)
  cone (x1,y1,z1) r1 (x2,y2,z2) r2 n
- void **CreateCylinder** (std::vector< std::string > arguments)
  cylinder (x1,y1,z1) (x2,y2,z2) r n
- void **Delete** (std::vector< std::string > arguments)
  delete id

- void **ClearAll** (std::vector< std::string > arguments)
  clear_all
- void **Move** (std::vector< std::string > arguments)
  move id (x,y,z)
- void **Rotate** (std::vector< std::string > arguments)
  rotate id (x,y,z) (,,)
- void **Save** (std::vector< std::string > arguments)
  save name
- void **Load** (std::vector< std::string > arguments)
  load name
- void **RenderToFile** (std::vector< std::string > arguments)
  Niezaimplementowana.
- void **SetFillStyle** (std::vector< std::string > arguments)
  Niezaimplementowana.
- void **SetFillColor** (std::vector< std::string > arguments)
  Niezaimplementowana.
- void **View** (std::vector< std::string > arguments)
  Niezaimplementowana.
- void **SetViewRange** (std::vector< std::string > arguments)
  set_view_range right | front | top r
- void **CameraLookAt** (std::vector< std::string > arguments)
  Niezaimplementowana.
- void **CameraAt** (std::vector< std::string > arguments)
  camera_at (x,y,z)
- void **CameraFov** (std::vector< std::string > arguments)
  camera_fov alfa
- void **Touch** (std::vector< std::string > arguments)
  Niezaimplementowana.
- void **DebugScale** (std::vector< std::string > arguments)
  debug_scale s
- void **DebugCulling** (std::vector< std::string > arguments)
  debug_culling true | false

Metody chronione

- void **MainFrameOnClose** (wxCloseEvent &event)
- void **MainFrameOnPaint** (wxPaintEvent &event)
- void **_promptFieldOnTextEnter** (wxCommandEvent &event)

Atrybuty chronione

- **Console** ∗ **m_console**
- **ShapeContainer** ∗ **m_shapeContainer**

4.14.1 Opis szczegółowy

Implementing **MainFrame** (str. **??**)

4.14.2 Dokumentacja konstruktora i destruktora

4.14.2.1 ProjectMainFrame()

```
ProjectMainFrame::ProjectMainFrame (
            wxWindow * parent )
```

Constructor

Dokumentacja dla tej klasy została wygenerowana z pliku:

- Include/ **ProjectMainFrame.h**

4.15 Dokumentacja struktury Segment

Struktura reprezentująca linię w przestrzeni trójwymiarowej.

`#include <Segment.h>`

Metody publiczne

- **Segment** (**Point** _begin, **Point** _end)

Atrybuty publiczne

- **Point begin**
- **Point end**

4.15.1 Opis szczegółowy

Struktura reprezentująca linię w przestrzeni trójwymiarowej.

Dokumentacja dla tej struktury została wygenerowana z pliku:

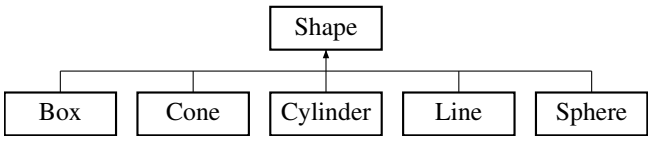
- Include/Segment.h

4.16 Dokumentacja klasy Shape

Klasa wirtualna reprezentująca bryłę, przechowuje jej identyfikator i segmenty.

`#include <Shape.h>`

Diagram dziedziczenia dla Shape



Metody publiczne

- **Shape** (unsigned id)

  Konstruktor przypisujący identyfikator do bryły.
- unsigned **getId** ()

  Zwraca identyfikator bryły.
- virtual void **move** (double x, double y, double z)

  Przesuwa bryłę o wektor [x, y, z].
- virtual void **rotate** (double x, double y, double z, double a, double b, double g)

  Obraca bryłę wokół punktu (x, y, z) o kąt a wzdłuż osi X, b wzdłuż osi Y i kąt g wzdłuż osi Z.
- std::vector< **Segment** > **getData** ()

  Zwraca kontener segmentów.
- std::vector< **Segment** > * **getDataPtr** ()

  Zwraca wskaźnik na wektor segmentów.
- virtual std::string **toString** ()

  Zwraca podstawowe informacje o bryle w formacie odpowiednim do wyświetlania na liście brył.

Statyczne metody publiczne

- static std::vector< **Segment** > **findCircle** (double r)

  Funkcja realizująca <midpoint circle algorithm>.

Atrybuty chronione

- unsigned **m_id**
- std::vector< **Segment** > **m_data**

4.16.1 Opis szczegółowy

Klasa wirtualna reprezentująca bryłę, przechowuje jej identyfikator i segmenty.

4.16.2 Dokumentacja konstruktora i destruktora

4.16.2.1 Shape()

```
Shape::Shape (
            unsigned id ) [inline]
```

Konstruktor przypisujący identyfikator do bryły.

Parametry

id	Identyfikator.

4.16.3 Dokumentacja funkcji składowych

4.16.3.1 findCircle()

```
static std::vector< Segment> Shape::findCircle (
            double r )  [static]
```

Funkcja realizująca <midpoint circle algorithm>.

Parametry

r	Promień tworzonego okręgu.

Zwraca

Segmenty utworzonego okręgu.

4.16.3.2 getData()

```
std::vector< Segment> Shape::getData ( )  [inline]
```

Zwraca kontener segmentów.

Zwraca

Kontener segmentów.

4.16.3.3 getDataPtr()

```
std::vector< Segment>* Shape::getDataPtr ( )  [inline]
```

Zwraca wskaźnik na wektor segmentów.

Zwraca

Wskaźnik na wektor segmentów.

4.16.3.4 getId()

```
unsigned Shape::getId ( )  [inline]
```

Zwraca identyfikator bryły.

Zwraca

Identyfikator.

4.16.3.5 move()

```
virtual void Shape::move (
            double x,
            double y,
            double z )  [virtual]
```

Przesuwa bryłę o wektor [x, y, z].

Parametry

x	Wartość x wektora przesunięcia.
y	Wartość y wektora przesunięcia.
z	Wartość z wektora przesunięcia.

Reimplementowana w **Cylinder** (str. **??**), **Cone** (str. **??**) i **Sphere** (str. **??**).

4.16.3.6 rotate()

```
virtual void Shape::rotate (
            double x,
            double y,
            double z,
            double a,
            double b,
            double g )  [virtual]
```

Obraca bryłę wokół punktu (x, y, z) o kąt a wzdłuż osi X, b wzdłuż osi Y i kąt g wzdłuż osi Z.

Parametry

x	Współrzędna x środka obrotu.
y	Współrzędna y środka obrotu.
z	Współrzędna z środka obrotu.
a	Kąt obrotu wokół osi X.
b	Kąt obrotu wokół osi Y.
g	Kąt obrotu wokół osi Z.

Reimplementowana w **Cylinder** (str. **??**), **Cone** (str. **??**) i **Sphere** (str. **??**).

4.16.3.7 toString()

```
virtual std::string Shape::toString ( )  [inline], [virtual]
```

Zwraca podstawowe informacje o bryle w formacie odpowiednim do wyświetlania na liście brył.

Zwraca

Łańcuch znaków z informacjami o bryle.

Reimplementowana w **Cylinder** (str. **??**), **Cone** (str. **??**), **Sphere** (str. **??**), **Box** (str. **??**) i **Line** (str. **??**).

Dokumentacja dla tej klasy została wygenerowana z pliku:

- Include/Shape.h

4.17 Dokumentacja klasy ShapeContainer

Klasa opisujca kontener na bryy.

```
#include <ShapeContainer.h>
```

Metody publiczne

- **ShapeContainer** (wxTextCtrl ∗shapeList)
  *Konstruktor klasy **ShapeContainer** (*str. **??***).*
- void **addShape** (**Shape** ∗shape)
  Dodaje bry konteneru.
- bool **deleteShape** (unsigned id)
  Usuwa zadan bryDoxyCompactList
  *void **deleteAllShapes** ()*
  Usuwa wszystkie bryy z konteneru.
  *void **moveShape** (unsigned id, double x, double y, double z)*
  Przesuwa bryektor [x, y, z].
  *void **rotateShape** (unsigned id, double x, double y, double z, double a, double b, double g)*
  Obraca brykntu (x, y, z) o kt a wzdu osi X, b wzdu osi Y i kt g wzdu osi Z.
  *void **refreshList** ()*
  Odwiea listy.
  *void **saveToFile** (std::string filename)*
  Zapisuje dane bry i ustawienia do pliku.
  *void **loadFromFile** (std::string filename)*
  Wczytuje dane bry i ustawienia z pliku.
  *std::vector< double > **validateWords** (std::vector< std::string > words)*
  Przetwarza linijkiku wygenerowanego przez aplikacjetoda przeznaczona do uytku loadFromFile.
  *unsigned **getNextId** ()*
  Zwraca kolejn wartoentyfikatora, kta zostazypisana nastej utworzonej bryle.
  *std::vector< **Shape** ∗ > **getShapes** ()*
  Zwraca kontener bry.

Atrybuty chronione

- std::vector< **Shape** * > **m_shapes**
- wxTextCtrl * **m_shapeList**
- unsigned **m_nextId**

4.17.1 Opis szczegółowy

Klasa opisujca kontener na bryy.

4.17.2 Dokumentacja konstruktora i destruktora

4.17.2.1 ShapeContainer()

```
ShapeContainer::ShapeContainer (
            wxTextCtrl * shapeList )  [inline]
```

Konstruktor klasy **ShapeContainer** (str. **??**).

Parametry

shapeList	Kontrolka w kt wypisywane b bryy.

4.17.3 Dokumentacja funkcji składowych

4.17.3.1 addShape()

```
void ShapeContainer::addShape (
             Shape * shape )
```

Dodaje bry konteneru.

Parametry

shape	Brya.

4.17.3.2 deleteShape()

```
bool ShapeContainer::deleteShape (
            unsigned id )
```

Usuwa zadan bry

Parametry

id	Identyfikator bryy do usunia.

Zwraca

Zwraca true jeli jaka brya zostaa usuni, w przeciwnym wypadku false.

4.17.3.3 getNextId()

```
unsigned ShapeContainer::getNextId ( )
```

Zwraca kolejn wartoentyfikatora, kta zostazypisana nastej utworzonej bryle.

Zwraca

Identyfikator.

4.17.3.4 getShapes()

```
std::vector< Shape* > ShapeContainer::getShapes ( )
```

Zwraca kontener bry.

Zwraca

Kontener bry.

4.17.3.5 loadFromFile()

```
void ShapeContainer::loadFromFile (
            std::string filename )
```

Wczytuje dane bry i ustawienia z pliku.

Wczytuje dane bry i ustawienia z pliku. Obecne dane zostan usuni. Weryfikuje poprawnoadni pliku, ale nie weryfikuje poprawnoci danych pod ktem sensownoci ich wykorzystania w programie.

Parametry

filename	Nazwa pliku.

Wyjątki

std::runtime_error	Rzucany jeli odczyt pliku sie powiedzie. Obecne dane nie zostan wtedy usuni.

4.17.3.6 moveShape()

```
void ShapeContainer::moveShape (
            unsigned id,
            double x,
            double y,
            double z )
```

Przesuwa bryektor [x, y, z].

Parametry

id	Identyfikator bryy do przesunia.
x	Wartoektora przesunia.
y	Wartoektora przesunia.
z	Wartoektora przesunia.

4.17.3.7 rotateShape()

```
void ShapeContainer::rotateShape (
            unsigned id,
            double x,
            double y,
            double z,
            double a,
            double b,
            double g )
```

Obraca bryknktu (x, y, z) o kt a wzdu osi X, b wzdu osi Y i kt g wzdu osi Z.

Parametry

id	Identyfikator bryy do obria.
x	Wspa x rodka obrotu.
y	Wspa y rodka obrotu.
z	Wspa z rodka obrotu.
a	Kt obrotu woki X.
b	Kt obrotu woki Y.
g	Kt obrotu woki Z.

4.17.3.8 saveToFile()

```
void ShapeContainer::saveToFile (
            std::string filename )
```

Zapisuje dane bry i ustawienia do pliku.

Parametry

filename	Nazwa pliku.

Wyjątki

std::runtime_error	Rzucany jeli zapis pliku sie powiedzie.

4.17.3.9 validateWords()

```
std::vector<double> ShapeContainer::validateWords (
            std::vector< std::string > words )
```

Przetwarza linijkiku wygenerowanego przez aplikacjetoda przeznaczona do uytku loadFromFile.

Parametry

words	Linijka podzielona na sowa.

Zwraca

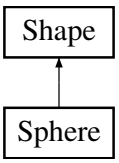
Kontener zawierajcy wartoci wyuskane z linijki.

Dokumentacja dla tej klasy została wygenerowana z pliku:

– Include/ShapeContainer.h

4.18 Dokumentacja klasy Sphere

Diagram dziedziczenia dla Sphere



Metody publiczne

– **Sphere** (unsigned id, double x, double y, double z, double r, double m, double n)
  Tworzy sferę.
– void **move** (double x, double y, double z) override
  Przesuwa bryłę o wektor [x*,* y*,* z*].*
– void **rotate** (double x, double y, double z, double a, double b, double g) override
  Obraca bryłę wokół punktu (x*,* y*,* z*) o kąt* a *wzdłuż osi X,* b *wzdłuż osi Y i kąt* g *wzdłuż osi Z.*
– std::string **toString** () override
  zwraca informacje o obiekcie w formie std::string.

Atrybuty chronione

– **Point m_center**
– double **m_radius**
– double **m_meridians**
– double **m_parallels**

Dodatkowe Dziedziczone Składowe

4.18.1 Dokumentacja konstruktora i destruktora

4.18.1.1 Sphere()

```
Sphere::Sphere (
            unsigned id,
            double x,
            double y,
            double z,
            double r,
            double m,
            double n )
```

Tworzy sferę.

Parametry

id	Identyfikator.
x	Współrzędna x środka.
y	Współrzędna y środka.
z	Współrzędna z środka.
r	Promień.
m	Ilość południków.
n	Ilość równoleżników.

4.18.2 Dokumentacja funkcji składowych

4.18.2.1 move()

```
void Sphere::move (
            double x,
            double y,
            double z )  [override], [virtual]
```

Przesuwa bryłę o wektor [x, y, z].

Parametry

x	Wartość x wektora przesunięcia.
y	Wartość y wektora przesunięcia.
z	Wartość z wektora przesunięcia.

Reimplementowana z **Shape** (str. **??**).

4.18.2.2 rotate()

```
void Sphere::rotate (
            double x,
            double y,
            double z,
            double a,
            double b,
            double g )  [override], [virtual]
```

Obraca bryłę wokół punktu (x, y, z) o kąt a wzdłuż osi X, b wzdłuż osi Y i kąt g wzdłuż osi Z.

Parametry

x	Współrzędna x środka obrotu.
y	Współrzędna y środka obrotu.
z	Współrzędna z środka obrotu.
a	Kąt obrotu wokół osi X.
b	Kąt obrotu wokół osi Y.
g	Kąt obrotu wokół osi Z.

Reimplementowana z **Shape** (str. **??**).

4.18.2.3 toString()

```
std::string Sphere::toString ( )  [override], [virtual]
```

zwraca informacje o obiekcie w formie std::string.

Zwraca

String z informacjami.

Reimplementowana z **Shape** (str. **??**).

Dokumentacja dla tej klasy została wygenerowana z pliku:

– Include/Shape.h

4.19 Dokumentacja struktury Triplet

Argument konsoli zawierający trzy liczby.

`#include <ArgumentTypes.h>`

Metody publiczne

– **Triplet** (double v1, double v2, double v3)

Atrybuty publiczne

– double **value1**
– double **value2**
– double **value3**

4.19.1 Opis szczegółowy

Argument konsoli zawierający trzy liczby.

Dokumentacja dla tej struktury została wygenerowana z pliku:

– Include/ArgumentTypes.h

Rozdział 5

Dokumentacja plików

5.1 Dokumentacja pliku Include/ProjectMainFrame.h

```
#include "Project.h"
#include <wx/dcbuffer.h>
#include <wx/dcclient.h>
#include <wx/dcmemory.h>
#include "Console.h"
#include "Presentation.h"
#include "ShapeContainer.h"
```

Komponenty

- class **ProjectMainFrame**

5.1.1 Opis szczegółowy

Subclass of **MainFrame** (str. **??**), which is generated by wxFormBuilder.