



AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ INFORMATYKI, ELEKTRONIKI I TELEKOMUNIKACJI

KATEDRA INFORMATYKI

PROJEKT INŻYNIERSKI
DOKUMENTACJA UŻYTKOWNIKA

Narzędzie do wizualizacji siatek trójwymiarowych

Tool for visualization of three-dimensional meshes

Autorzy:	<i>Wojciech Dymek, Katarzyna Głąb, Katarzyna Konieczna, Ewa Marczevska</i>
Kierunek studiów:	<i>Informatyka</i>
Opiekun pracy:	<i>dr inż. Tomasz Jurczyk</i>

Kraków, 2017

Spis treści

Podstawowe informacje	4
Wymagania aplikacji	4
Konfiguracja środowiska.....	5
Kompilacja – system Linux	5
Kompilacja – system Windows	5
Uruchomienie aplikacji.....	5
Plik konfiguracyjny <i>user.config.xml</i>	6
Struktura pliku	6
Interfejs programistyczny użytkownika API.....	8
Java	8
Pakiet geometry	8
Pakiet tool.....	11
C++	14
Geometry.h.....	14
Smeshalist.h.....	16
Python.....	19
Moduł geometry.py	19
Moduł Smeshalist.py	20
Wykorzystanie API w różnych językach programowania	22
Java.....	22
Python.....	22
C++.....	22
Nawigacja w widoku	23
Przybliżanie/oddalanie widoku.....	23
Przesuwanie widoku	23
Obracanie widoku.....	24

Resetowanie widoku	24
Smeshalist Manager	25
Statistics	25
Options	26
Filters	27
Types	27
Groups	28
Quality	29
Coordinates	30
Import/Export	31
Spis ilustracji	32

Podstawowe informacje

Smeshalist to aplikacja służąca do wizualizacji siatek trójwymiarowych oraz ich elementów. Jej celem jest ułatwienie m. in. tworzenia algorytmów z zakresu geometrii obliczeniowej poprzez wyświetlanie kolejnych kroków rozwiązania, zaimplementowanego w języku Java, C++ lub Python. Program może być uruchamiany na systemie operacyjnym Linuks lub Windows.

Wymagania aplikacji

Do poprawnego uruchomienia aplikacji zainstalować należy:

1. Java JRE 1.8
2. GLUT – OpenGL Utility Toolkit

Konfiguracja środowiska

Kompilacja – system Linux

Znajdując się w folderze głównym aplikacji uruchomić polecenie *make all*. Zostanie wygenerowany plik wykonywalny *SmeshalistCore*, który odpowiada za komunikację między poszczególnymi częściami systemu oraz wizualizację.

Kompilacja – system Windows

Dla systemu operacyjnego Windows, zostały przygotowane gotowe pliki uruchomieniowe .exe, dostępne w katalogach x86 oraz x64.

Uruchomienie aplikacji

Kroki potrzebne do poprawnego uruchomienia aplikacji:

1. W razie konieczności zmiana pliku konfiguracyjnego *user.config.xml*.
2. Uruchomienie *SmeshalistCore*.

W przypadku używania aplikacji *Smeshalist* na systemie operacyjnym Windows należy samodzielnie uruchomić okno *SmeshalistManagera* – poleceniem *java -jar SmeshalistManager.jar* lub poprzez dwukrotne kliknięcie przyciskiem myszy, bądź utworzenie katalogu *lib* w folderze z plikiem *Smeshalist.exe* oraz przekopiowanie do niego pliku *SmeshalistManager.jar*. Plik ten znajduje się w katalogu *lib*.

Plik konfiguracyjny *user.config.xml*

Plik konfiguracyjny powinien znajdować się w tej samej lokalizacji co plik wykonywalny *SmeshalistCore*.

Struktura pliku

```
<preferences>
  <port CORE="8383"/>
  <background theme="DARK"/>
  <groups>
    <g id="0" r="255" g="0" b="0"/>
    <g id="1" r="0" g="255" b="0"/>
  </groups>
  <points size="3"/>
  <qualityColors>
    <negQualityColor r="111" g="111" b="111"/>
    <color q="0.0" r="0" g="0" b="255"/>
    <color q="0.5" r="255" g="255" b="0"/>
    <color q="1.0" r="255" g="0" b="0"/>
  </qualityColors>
  <axes>
    <x r="255" g="0" b="0"/>
    <y r="0" g="255" b="0"/>
    <z r="0" g="0" b="255"/>
  </axes>
  <cuttingPlane r="255" g="255" b="255" a="20"/>
</preferences>
```

Rysunek 1 Przykładowa struktura pliku konfiguracyjnego

Konfiguracja znajduje się w węźle *<preferences>*. Dostępne opcje to:

1. *<port>* - umożliwia zmianę portu, na którym nasłuchuje serwer aplikacji. Nowy port należy podać jako atrybut *CORE*. Musi on być zgodny z wartością argumentu przekazywanego do metody *getInstance*. Domyślnie wykorzystywany jest port 8383.
2. *<background >* - konfiguracja koloru tła obszaru roboczego. W przypadku nie podania węzła, bądź nie uzupełnienia atrybutu *theme* domyślnie kolor jest biały. W celu ustawienia ciemnego tła należy dodać powyższy węzeł wraz z atrybutem *theme="DARK"*.

3. *<groups>* - konfiguracja kolorów zdefiniowanych dla poszczególnych grup. Aby ustawić kolor wybranej grupie należy dodać potomka w postaci węzła *<g id="X">* gdzie *X* oznacza numer grupy, wraz z atrybutami *r*, *g*, *b*.
4. *<points>* - umożliwia zmianę rozmiaru wyświetlanych wierzchołków. Wartość należy umieścić w atrybucie *size* węzła. Domyślną wartością jest 3.
5. *<qualityColors>* - umożliwia podanie palety kolorów wraz z odpowiadającymi im wartościami współczynnika jakości. Definicja polega na dodaniu potomka *<color>* wraz z atrybutami *q* – wartość współczynnika jakości oraz trójki *r*, *g*, *b*.
W przypadku wartości spoza przedziału [0; 1] można zdefiniować dodatkowy kolor poprzez dodanie potomka *<negQualityColor>* wraz z trójką *r*, *g*, *b*. W przypadku braku tej wartości zostaną użyta wartości domyślne – (159, 0, 255). Wartości pośrednie w przedziałach wyliczane są jako średnie ważone kolorów zdefiniowanych na końcach odpowiednich przedziałów. W przypadku nieokreślenia kolorów dla jakości równej 0 bądź 1, zostaną użyte wartości domyślne, odpowiednio (0,0,0) oraz (255,255,255).
6. *<axes>* - konfiguracja kolorów poszczególnych osi. Aby ustawić kolor wybranej osi należy dodać potomka odpowiednio *<x>/<y>/<z>*, wraz z podaniem trójki *r*, *g*, *b*. W przypadku niezdefiniowania kolorów zostaną użyte wartości domyślne, odpowiednio: dla osi x: (255, 0, 0), y: (0, 255, 0), z: (0, 0, 255).
7. *<cuttingPlane>* - umożliwia zmianę koloru oraz współczynnika przezroczystości płaszczyzn, będących wizualizacji filtrów po współrzędnych. Wartości (R, G, B, A) podawane są jako kolejne atrybuty *r*, *g*, *b*, *a* węzła. W przypadku niezdefiniowania koloru zostanie użyty kolor domyślny – (127, 25, 25, 25).

Interfejs programistyczny użytkownika API

Java

Pakiet geometry

Zawiera klasy stanowiące wewnętrzny dla narzędzia Smeshalyst model struktur geometrycznych.

Klasa Point3D

Klasa przechowująca współrzędne punktu, będąca składową wszystkich klas struktur geometrycznych.

- konstruktor *Point3D(double x, double y, double z)*
- konstruktor *Point3D(double x, double y)*
- metody dostępne do pól współrzędnych: *double getX(), double getY(), double getZ(), void setX(double x), void setY(double y), void setZ(double z)*

Klasa Vertex

- Point3D point
- double quality
- String label
- int groupId
- konstruktor *Vertex(Point3D point)* – tworzy obiekt klasy Vertex
- metody dostępne do pól:
 - *void setPoint(Point3D point)*
 - *Point3D getPoint()*
 - *void setQuality(double quality)*
 - *double getQuality()*
 - *void setLabel(String label)*
 - *String getLabel()*
 - *void setGroupId(int groupId)*
 - *int getGroupId()*

Klasa Edge

- Point3D v1
- Point3D v2

- double quality
- String label
- int groupId
- konstruktor Edge(Point3D v1, Point3D v2) – tworzy obiekt klasy Edge
- metody dostępne do pól:
 - void setV1(Point3D point)
 - Point3D getV1()
 - void setV2(Point3D point)
 - Point3D getV2()
 - void setQuality(double quality)
 - double getQuality()
 - void setLabel(String label)
 - String getLabel()
 - void setGroupId(int groupId)
 - int getGroupId()

Klasa TriangleFace

- Point3D v1
- Point3D v2
- Point3D v3
- double quality
- String label
- int groupId
- konstruktor TriangleFace(Point3D v1, Point3D v2, Point3D v3) – tworzy obiekt klasy TriangleFace
- metody dostępne do pól:
 - void setV1(Point3D point)
 - Point3D getV1()
 - void setV2(Point3D point)
 - Point3D getV2()
 - void setV3(Point3D point)
 - Point3D getV3()
 - void setQuality(double quality)
 - double getQuality()

- void setLabel(String label)
- String getLabel()
- void setGroupId(int groupId)
- int getGroupId()

Klasa Block

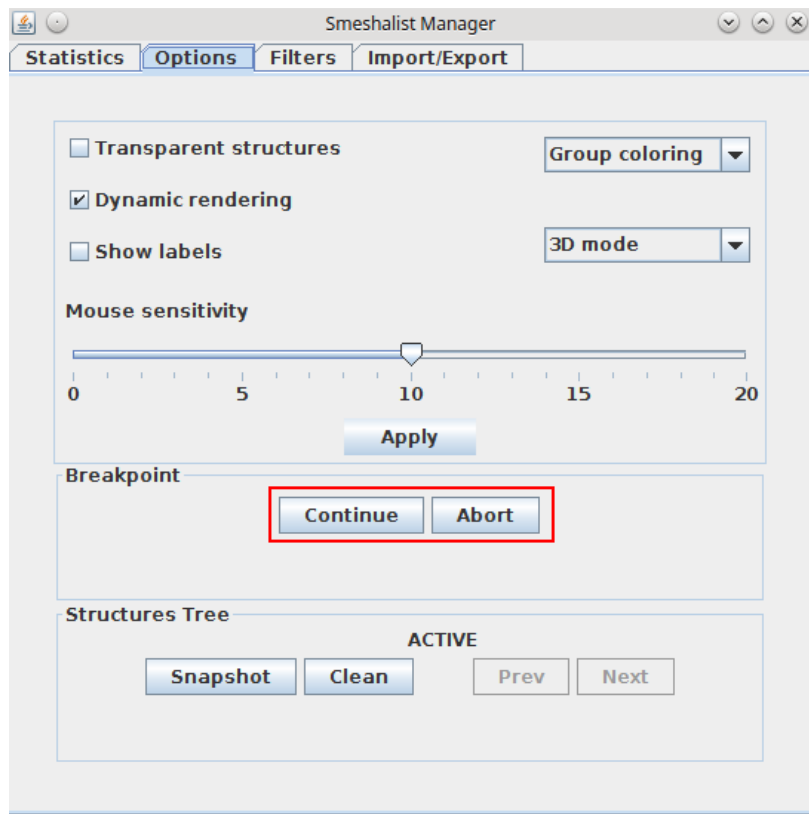
- Point3D v1
- Point3D v2
- Point3D v3
- Point3D v4
- double quality
- String label
- int groupId
- konstruktor Block(Point3D v1, Point3D v2, Point3D v3, Point3D v4) – tworzy obiekt klasy Block
- metody dostępne do pól:
 - void setV1(Point3D point)
 - Point3D getV1()
 - void setV2(Point3D point)
 - Point3D getV2()
 - void setV3(Point3D point)
 - Point3D getV3()
 - void setV4(Point3D point)
 - Point3D getV4()
 - void setQuality(double quality)
 - double getQuality()
 - void setLabel(String label)
 - String getLabel()
 - void setGroupId(int groupId)
 - int getGroupId()

Pakiet tool

Klasa Smeshalist

Jest to główna klasa narzędzia, dostarczająca metod umożliwiających dodawanie geometrii do wizualizacji.

- *Smeshalist getInstance(boolean hardReset)* – statyczna metoda, tworzy instancję lub zwraca istniejącą instancję narzędzia działającego na domyślnym porcie numer 8383. Flaga *hardReset* określa, czy aplikacja *Smeshalist* powinna wrócić do stanu początkowe – usunąć wszystkie filtry, instancje prezentowanych siatek etc.
- *Smeshalist getInstance(int portNumber, boolean hardReset)* – statyczna metoda, tworzy instancję lub zwraca istniejącą instancję narzędzia działającego na porcie przekazanym jako argument wywołania metody. Flaga *hardReset* określa, czy aplikacja *Smeshalist* powinna wrócić do stanu początkowe – usunąć wszystkie filtry, instancje prezentowanych siatek etc.
- *void destroySmeshalist()* – statyczna metoda, wołana po zakończeniu pracy z narzędziem *Smeshalist*, zamyka socket służący do komunikacji wewnętrznej
- *void addGeometry(Vertex vertex)* – metoda dodaje daną strukturę do bufora danych, które będą przesłane do wizualizacji
- *void addGeometry(Edge edge)*
- *void addGeometry(TriangleFace triangleFace)*
- *void addGeometry(Block block)*
- *void flushBuffer()* – przesyła zgromadzone w buforze struktury geometryczne do modułu wizualizacji
- *void breakpoint()* – metoda wstrzymująca działanie algorytmu do momentu wybrania jednej z dwóch opcji (*Continue*, *Abort*) dostępnych w oknie *Smeshalist Manager*



Rysunek 2 Obsługa zdarzenia breakpoint w oknie Smeshalist Manager

- *void render()* – metoda wymuszająca wyświetlenie przesłanych struktur w przypadku odznaczonej w oknie Smeshalist Manager opcji *Dynamic rendering*
- *void clean()* – metoda czyszcząca zawartość aktywnego drzewa struktur

Przykład użycia

```
import java.util.Random;
import geometry.Edge;
import geometry.Point3D;
import geometry.Vertex;
import helpers.CoreNotRunningException;
import tool.Smeshalist;

public class Example01 {
    public static void main(String[] args) {
        Smeshalist tool;
        try {
            tool = Smeshalist.getInstance(true);
            Random r = new Random();
            for (int i=0; i<100; i++) {
                Vertex v = new Vertex(new Point3D(r.nextDouble()*10-5,
                    r.nextDouble()*10-5, r.nextDouble()*10-5));
                v.setGroupId(3);
                tool.addGeometry(v);
            }
            for (int i=0; i<100; i++) {
                Point3D v1 = new Point3D(r.nextDouble()*10-5,
                    r.nextDouble()*10-5, r.nextDouble()*10-5);
                Point3D v2 = new Point3D(r.nextDouble()*10-5,
                    r.nextDouble()*10-5, r.nextDouble()*10-5);
                Edge edge = new Edge(v1, v2);
                edge.setGroupId(4);
                tool.addGeometry(edge);
            }
            tool.breakpoint();
            Smeshalist.destroySmeshalist();
        } catch (CoreNotRunningException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

C++

Geometry.h

Klasa Geometry

Klasa bazowa wszystkich dostępnych geometrii. Zawiera elementy służące do opisania każdej z nich.

- *void SetQuality(double q)* – ustawia pole *quality*
- *double GetQuality()* – zwraca wartość pola *quality*
- *void SetGroupId(int id)* – ustawia pole *group_id*
- *int GetGroupId()* – zwraca wartość pola *group_id*
- *void SetLabel(string l)* – ustawia pole *label*
- *string GetLabel()* – zwraca wartość pola *label*

Klasa Point3D

Klasa przechowująca współrzędne punktu, będąca składową wszystkich klas struktur geometrycznych.

- konstruktor *Point3D()*
- konstruktor *Point3D(double x, double y, double z)*
- konstruktor *Point3D(double x, double y)* – współrzędna *z* zostaje ustawiona domyślnie na wartość 0
- metody dostępne do pól współrzędnych: *double GetX()*, *double GetY()*, *double GetZ()*, *void SetX(double x)*, *void SetY(double y)*, *void SetZ(double z)*

Klasa Vertex

- *Point3D point*
- konstruktor *Vertex()*
- konstruktor *Vertex(double x, double y, double z)*
- konstruktor *Vertex(Point3D point)*
- metody dostępne do pól:
 - *void SetPoint(Point3D point)*
 - *Point3D GetPoint()*

Klasa Edge

- Point3D v1
- Point3D v2
- konstruktor Edge()
- konstruktor Edge(Point3D v1, Point3D v2) – tworzy obiekt klasy Edge
- metody dostępne do pól:
 - void SetV1(Point3D point)
 - Point3D GetV1()
 - void SetV2(Point3D point)
 - Point3D GetV2()

Klasa Face

- Point3D v1
- Point3D v2
- Point3D v3
- konstruktor Face()
- konstruktor Face(Point3D v1, Point3D v2, Point3D v3) – tworzy obiekt klasy Face
- metody dostępne do pól:
 - void SetV1(Point3D point)
 - Point3D GetV1()
 - void SetV2(Point3D point)
 - Point3D GetV2()
 - void SetV3(Point3D point)
 - Point3D GetV3()

Klasa Block

- Point3D v1
- Point3D v2
- Point3D v3
- Point3D v4
- konstruktor Block()
- konstruktor Block(Point3D v1, Point3D v2, Point3D v3, Point3D v4) – tworzy obiekt klasy Block
- metody dostępne do pól:

- void SetV1(Point3D point)
- Point3D GetV1()
- void SetV2(Point3D point)
- Point3D GetV2()
- void SetV3(Point3D point)
- Point3D GetV3()
- void SetV4(Point3D point)
- Point3D GetV4()

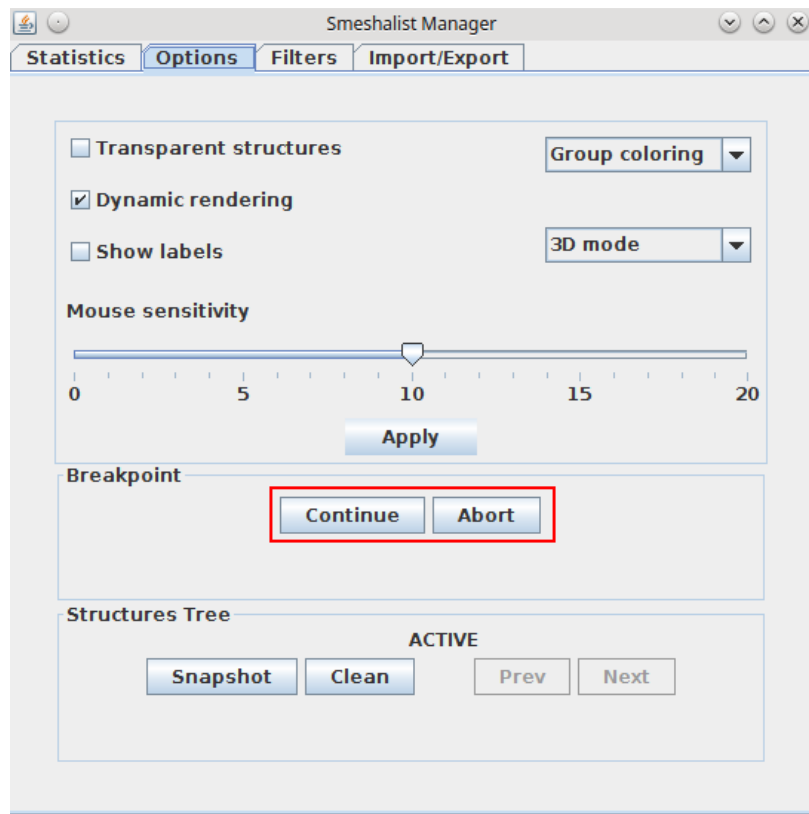
Smeshalist.h

Klasa Smeshalist

Jest to główna klasa narzędzia, dostarczająca metod umożliwiających dodawanie geometrii do wizualizacji.

- *Smeshalist& GetInstance()* – statyczna metoda, tworzy instancję lub zwraca istniejącą instancję narzędzia działającego na domyślnym porcie numer 8383.
- *Smeshalist& GetInstance(int port_number)* – statyczna metoda, zwraca istniejącą instancję narzędzie lub tworzy instancję działającą na podanym porcie.
- *Smeshalist& GetInstance(bool hard_reset)* – statyczna metoda, zwraca istniejącą instancję narzędzia lub tworzy instancję działającą na domyślnym porcie 8383. Flaga *hardReset* określa, czy aplikacja *Smeshalist* powinna wrócić do stanu początkowe – usunąć wszystkie filtry, instancje prezentowanych siatek etc.
- *Smeshalist& GetInstance(int port_number, bool hard_reset)* – statyczna metoda, zwraca istniejącą instancję narzędzia lub tworzy instancję działającą na podanym porcie. Flaga *hardReset* określa, czy aplikacja *Smeshalist* powinna wrócić do stanu początkowego – usunąć wszystkie filtry, instancje prezentowanych siatek etc.
- *void AddGeometry(Vertex &vertex)* – metoda dodaje daną strukturę do bufora danych, które będą przesłane do wizualizacji
- *void AddGeometry(Edge &edge)*
- *void AddGeometry(Face &face)*
- *void AddGeometry(Block &block)*
- *void FlushBuffer()* – przesyła zgromadzone w buforze struktury geometryczne do modułu wizualizacji

- *void Breakpoint()* – metoda wstrzymująca działanie algorytmu do momentu wybrania jednej z dwóch opcji (*Continue*, *Abort*) dostępnych w oknie Smesalist Manager



Rysunek 3 Obsługa zdarzenia brakepoint w oknie Smesalist Manager

- *void Render()* – metoda wymuszająca wyświetlenie przesłanych struktur w przypadku odznaczonej w oknie Smesalist Manager opcji *Dynamic rendering*
- *void Clean()* – metoda czyszcząca zawartość aktywnego drzewa struktur

Przykład użycia

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <cerrno>
#include <ctime>
#include "Smeshalist.h"
#include "Geometry.h"

using namespace std;

int N = 10;
double frand() {
    return ((double) ((double) rand() / (double) RAND_MAX));
}
Point3D genPoint() {
    return Point3D(frand()*3.0, frand()*3.0, frand()*3.0);
}
int main() {
    Smeshalist tool = Smeshalist::GetInstance(true);
    srand(time(NULL));

    for (int i = 0; i < N; i++) {
        Face face = Face(genPoint(), genPoint(), genPoint());
        face.SetGroupId(4);
        tool.AddGeometry(face);
    }
    for (int i = 0; i < N; i++) {
        Block block = Block(genPoint(), genPoint(), genPoint(), genPoint());
        block.SetGroupId(5);
        tool.AddGeometry(block);
    }
    tool.FlushBuffer();
    tool.Render();
    tool.Breakpoint();
}
```

Python

Moduł geometry.py

Point3D

Klasa przechowująca współrzędne punktu, będąca składową wszystkich klas struktur geometrycznych.

- `quality` – pole typu `double`, wskaźnik jakości struktury
- `label` – pole typu `string`, jest to etykieta struktury
- `groupId` – pole typu `int`, identyfikator grupy, do której należy struktura
- `__init__(self, x, y, z)` – konstruktor klasy, gdzie `x, y, z` to współrzędne punktu
- `__init__(self, x, y)` – konstruktor klasy, gdzie `x, y` to współrzędne punktu (współrzędna `z` przyjmuje w tym przypadku wartość 0)

Edge

- `quality` – pole typu `double`, wskaźnik jakości struktury
- `label` – pole typu `string`, jest to etykieta struktury
- `groupId` – pole typu `int`, identyfikator grupy, do której należy struktura
- `__init__(self, v1, v2)` – konstruktor klasy, gdzie `v1, v2` są typu *Point3D*

TriangleFace

- `quality` – pole typu `double`, wskaźnik jakości struktury
- `label` – pole typu `string`, jest to etykieta struktury
- `groupId` – pole typu `int`, identyfikator grupy, do której należy struktura
- `__init__(self, v1, v2, v3)` – konstruktor klasy, gdzie `v1, v2, v3` są typu *Point3D*

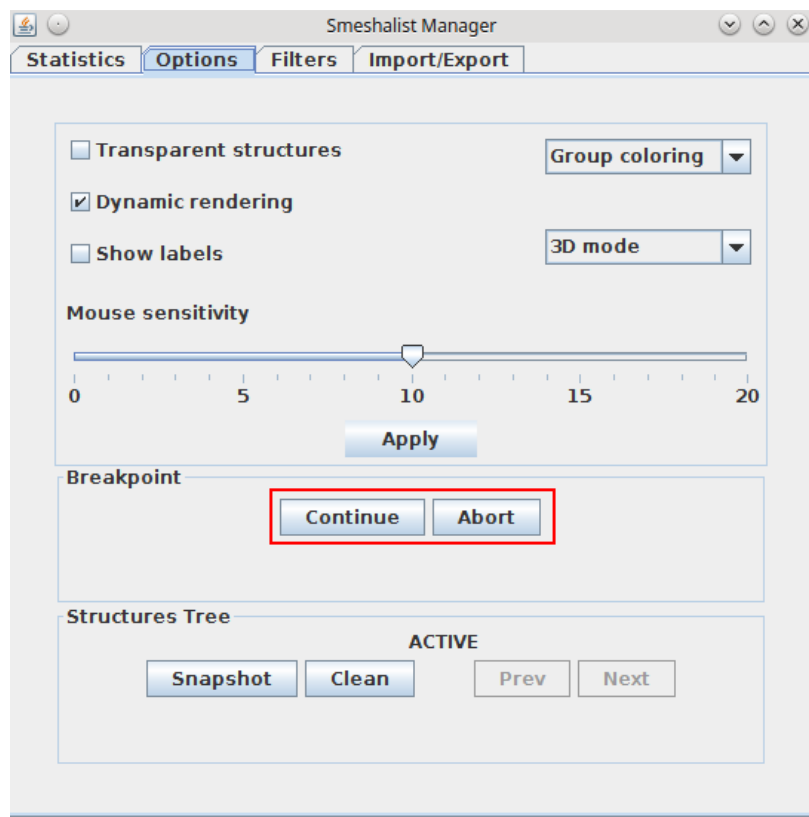
Block

- `quality` – pole typu `double`, wskaźnik jakości struktury
- `label` – pole typu `string`, jest to etykieta struktury
- `groupId` – pole typu `int`, identyfikator grupy, do której należy struktura
- `__init__(self, v1, v2, v3, v4)` – konstruktor klasy, gdzie `v1, v2, v3, v4` są typu *Point3D*

Moduł Smeshalist.py

Jest to główna klasa narzędzia, dostarczająca metod umożliwiających dodawanie geometrii do wizualizacji.

- *getInstance(portNumber, hardReset)* – tworzy instancję lub zwraca istniejącą instancję narzędzia działającego na porcie przekazanym jako argument wywołania metody. Flaga *hardReset* określa, czy aplikacja *Smeshalist* powinna wrócić do stanu początkowego – usunąć wszystkie filtry, instancje prezentowanych siatek etc.
- *addPoint3D(point3D)* – metoda dodaje daną strukturę do bufora danych, które będą przesłane do wizualizacji
- *addEdge(edge)*
- *addTriangleFace(triangleFace)*
- *addBlock(block)*
- *flushBuffer()* – przesyła zgromadzone w buforze struktury geometryczne do modułu wizualizacji
- *breakpoint()* – metoda wstrzymująca działanie algorytmu do momentu wybrania jednej z dwóch opcji (*Continue*, *Abort*) dostępnych w oknie Smeshalist Manager



Rysunek 4 Obsługa zdarzenia breakpoint w oknie Smeshalist Manager

- *render()* – metoda wymuszająca wyświetlenie przesłanych struktur w przypadku odznaczonej w oknie Smeshalist Manager opcji *Dynamic rendering*
- *clean()* – metoda czyszcząca zawartość aktywnego drzewa struktur

Przykład użycia

```
import Smeshalist
import geometry
import random

Smeshalist.getInstance(8383, False)

counter = 0
while counter < 1000:
    counter = counter + 1
    point1 = geometry.Point3D(random.uniform(-10.0, 10.0), random.uniform(
        -10.0, 10.0), random.uniform(-10.0, 10.0))
    vertex = geometry.Vertex(point1)
    vertex.groupId = 1
    Smeshalist.addVertex(vertex)

counter = 0
while counter < 1000:
    counter = counter + 1
    point1 = geometry.Point3D(random.uniform(-10.0, 10.0), random.uniform(
        -10.0, 10.0), random.uniform(-10.0, 10.0))
    point2 = geometry.Point3D(random.uniform(-10.0, 10.0), random.uniform(
        -10.0, 10.0), random.uniform(-10.0, 10.0))
    point3 = geometry.Point3D(random.uniform(-10.0, 10.0), random.uniform(
        -10.0, 10.0), random.uniform(-10.0, 10.0))

    triangleFace = geometry.TriangleFace(point1, point2, point3)
    triangleFace.groupId = 1
    Smeshalist.addTriangleFace(triangleFace)

Smeshalist.flushBuffer()
Smeshalist.breakpoint()
Smeshalist.clean()
```

Wykorzystanie API w różnych językach programowania

Java

Aby wykorzystać narzędzie w języku Java należy dołączyć do projektu archiwum *jar* znajdujące się w katalogu *JavaAPI*.

Python

Aby wykorzystać narzędzie w języku Python należy zaimportować do skryptu moduły *Smeshalist* oraz *geometry* znajdujące się w katalogu *PythonAPI*.

C++

Linux

Aby wykorzystać narzędzie w języku C++ należy zlinkować projekt z biblioteką *Smeshalist.a* oraz *libprotobuf.a*, znajdującymi się w katalogu *CppAPI*. Wymagane pliki nagłówkowe znajdują się w katalogu *CppAPI/include*.

Windows

Aby wykorzystać narzędzie w języku C++ należy zlinkować projekt z biblioteką *Smeshalist.lib* oraz *libprotobuf.lib*, znajdującymi się w katalogu *CppAPI/x64/x86*. Wymagane pliki nagłówkowe znajdują się w katalogu *CppAPI/include*.

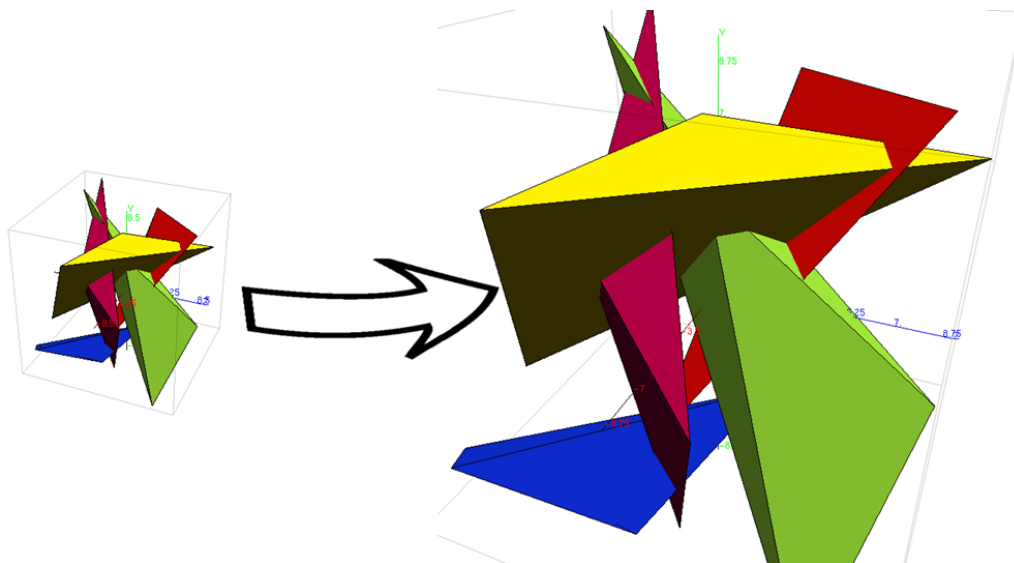
Kompilacja programów korzystających z CppAPI w systemie operacyjnym Windows wymaga dodatkowego kroku zmiany sposobu korzystania z bibliotek. Dla większości kompilatorów domyślną wartością jest */MD* (użycie wielowątkowej, dynamicznej wersji biblioteki). Opcję tą należy zmienić na */MT* – biblioteka wielowątkowa, statyczna.

Nawigacja w widoku

Aplikacja *Smeshalist* umożliwia pracę w trybie 3D oraz 2D. W trybie 2D użytkownik ma możliwość przybliżania/oddalania widoku oraz przesuwania się wzdłuż osi XY. Natomiast w trybie 3D istnieje dodatkowo możliwość obrotu całą sceną.

Przybliżanie/oddalanie widoku

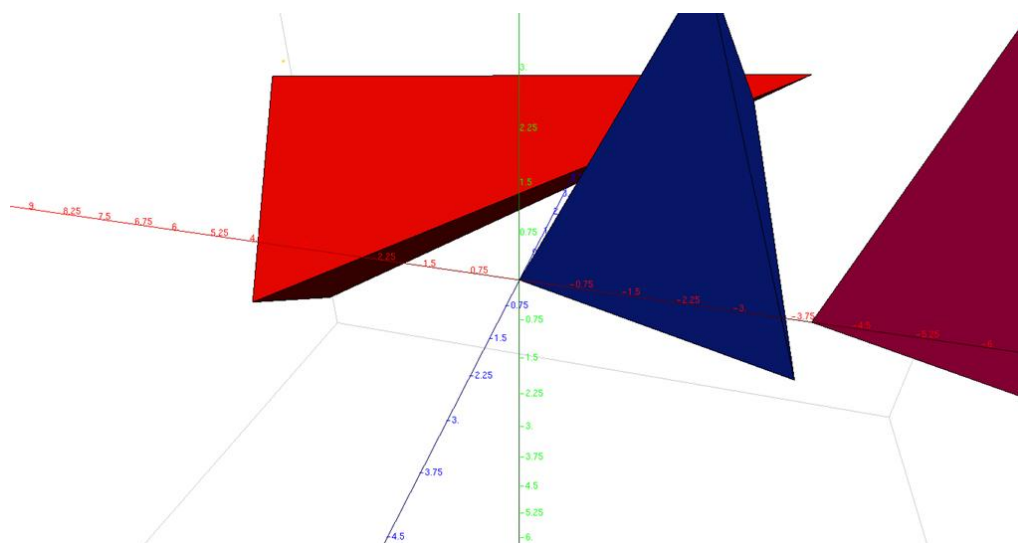
Funkcjonalność przybliżania/oddalania widoku jest dostępna poprzez użycie kółka myszy.



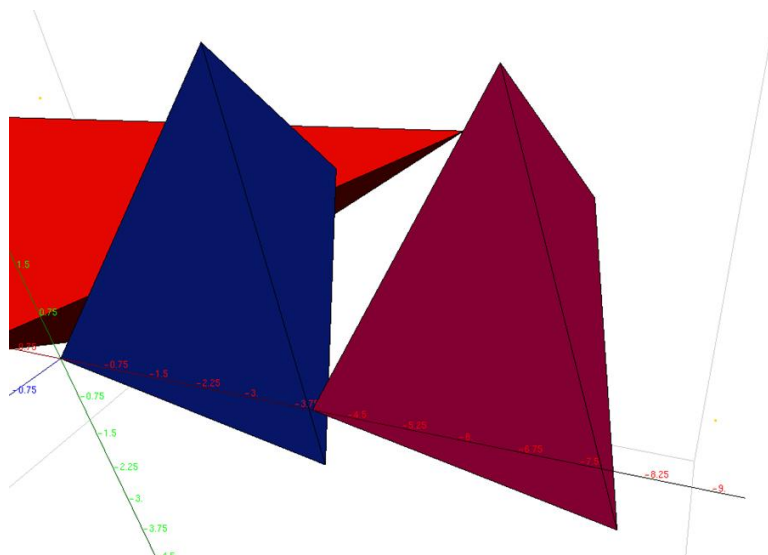
Rysunek 5 Widok - przybliżanie

Przesuwanie widoku

W celu przesunięcia widoku należy przytrzymać klawisz Shift i przeciągnąć lewy przycisk myszy w widoku.



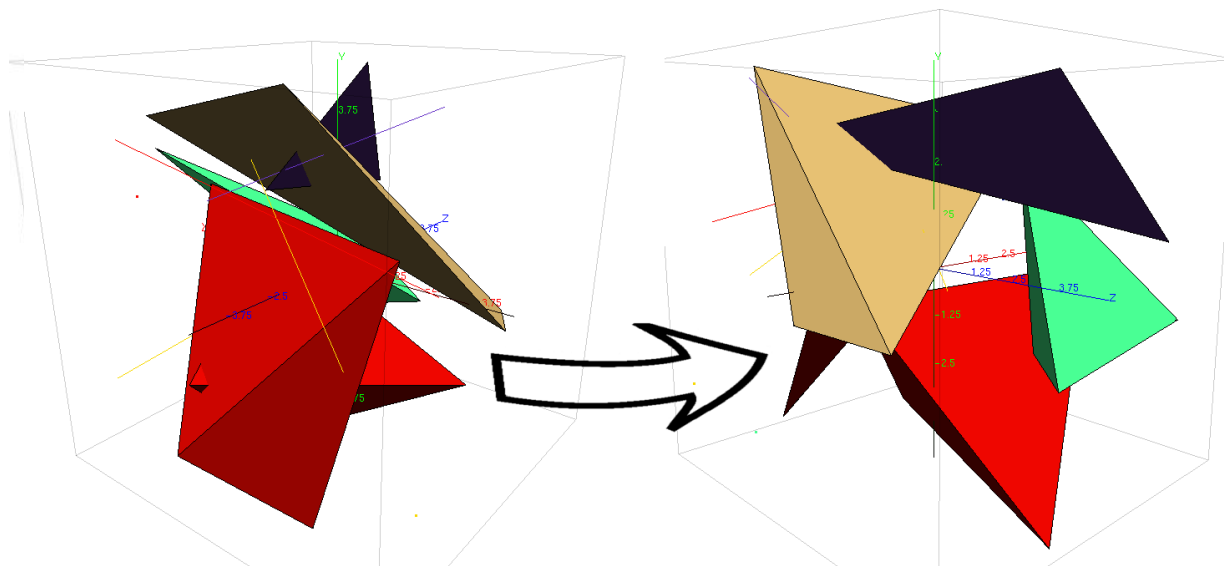
Rysunek 6 Widok - przesuwanie



Rysunek 7 Widok - przesuwanie (2)

Obracanie widoku

W celu obrócenia całego widoku należy przeciągnąć lewy przycisk myszy po scenie. Obrót następuje wokół środka sceny, o ile nie został on zmieniony w wyniku przesunięcia widoku. Funkcjonalność dostępna jest tylko w trybie 3D.



Rysunek 8 Widok - rotacja

Resetowanie widoku

Funkcjonalność dostępna poprzez naciśnięcie prawego przycisku myszy. Kamera zostaje ustawiona i odwrócona w taki sposób, aby środek układu współrzędnych znalazł się na środku widoku.

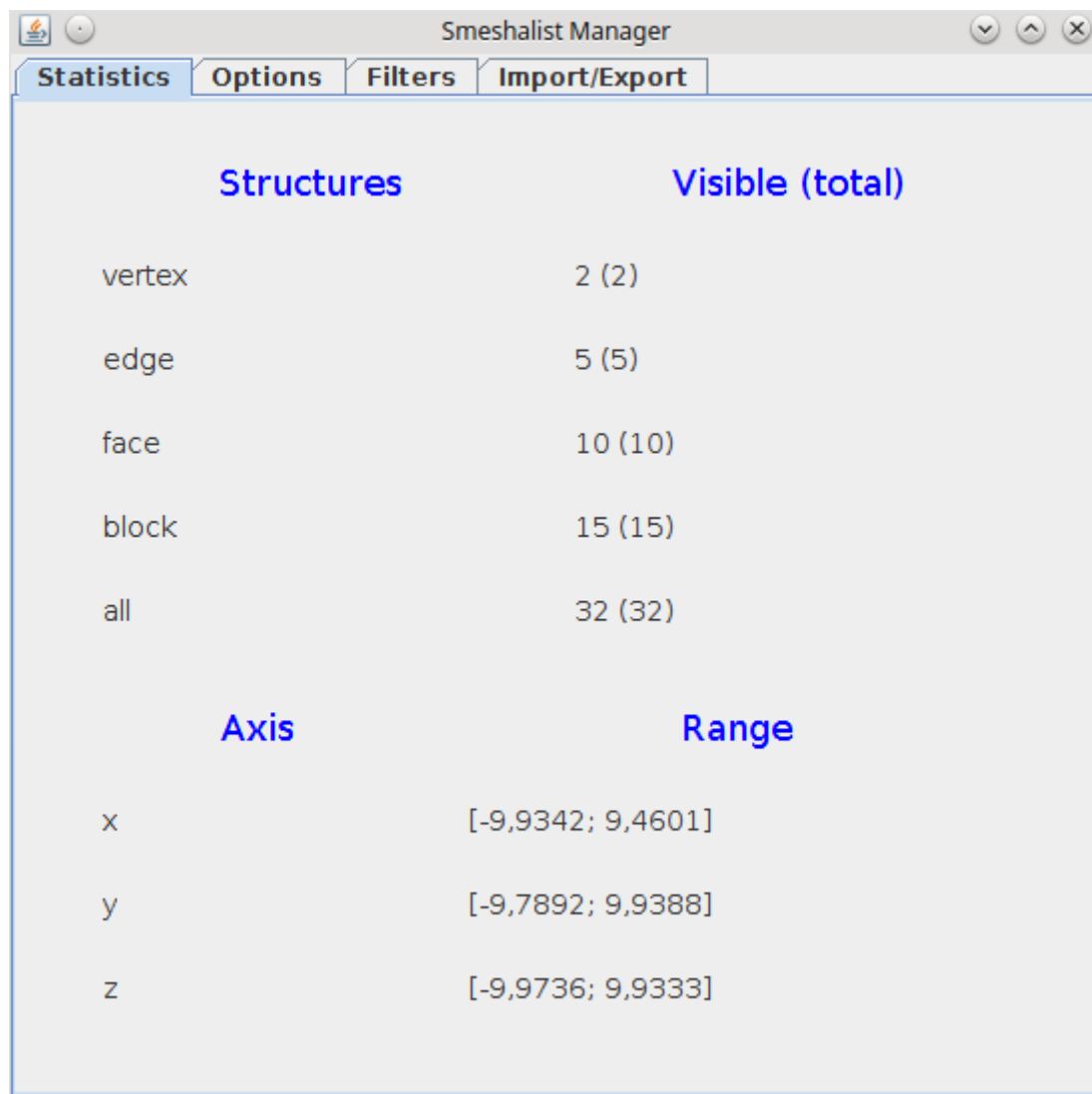
Smeshalist Manager

Smeshalist Manager to okienko umożliwiające kontrolę nad strukturami wyświetlanymi w obszarze roboczym oraz wyświetlanie informacji o tych strukturach.

Statistics

W zakładce statystyk wyświetlane są:

- ilość przesłanych i zwizualizowanych struktur danego typu
- współrzędne prostopadłościanu ograniczającego



Structures	Visible (total)
vertex	2 (2)
edge	5 (5)
face	10 (10)
block	15 (15)
all	32 (32)

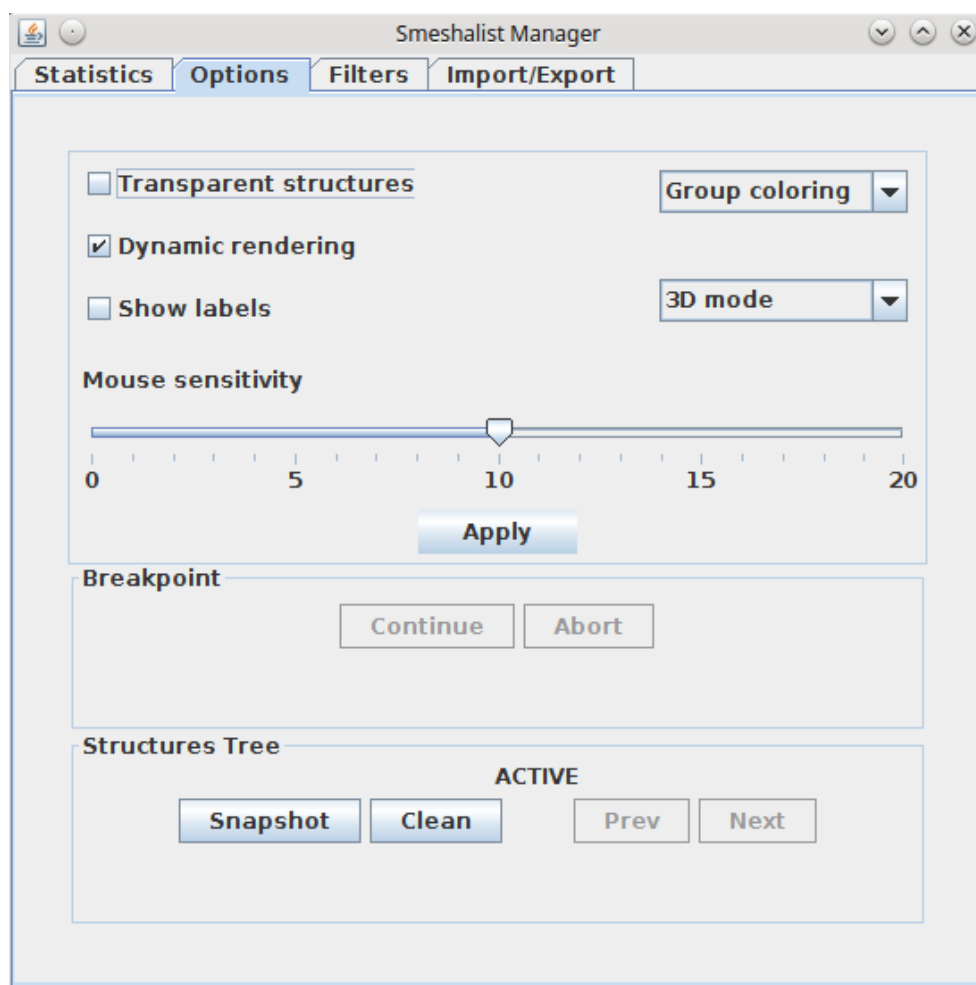
Axis	Range
x	[-9,9342; 9,4601]
y	[-9,7892; 9,9388]
z	[-9,9736; 9,9333]

Rysunek 9 Smeshalist Manager - Statystyki

Options

W zakładce opcji użytkownik ma możliwość:

- włączyć/wyłączyć dynamiczne renderowanie (*Dynamic rendering*)
- włączyć/wyłączyć wyświetlanie przezroczystych struktur (*Transparent structures*)
- włączyć/wyłączyć wyświetlanie etykiet struktur (*Show labels*)
- zmienić sposób kolorowania struktur (*Group coloring/Quality coloring*)
- zmienić tryb prezentacji danych (*3D mode/2D mode*)
- ustawić czułość myszki (*Mouse sensitivity*)
- zareagować na wywołanie metody *Breakpoint* (przyciski *Continue* i *Abort*)
- zrobić nowy zrzut drzewa struktur (przycisk *Snapshot*)
- wyczyścić aktywne drzewo struktur (przycisk *Clean*)
- przełączać się pomiędzy prezentowanymi instancjami drzewa struktur (przyciski *Prev* i *Next*)



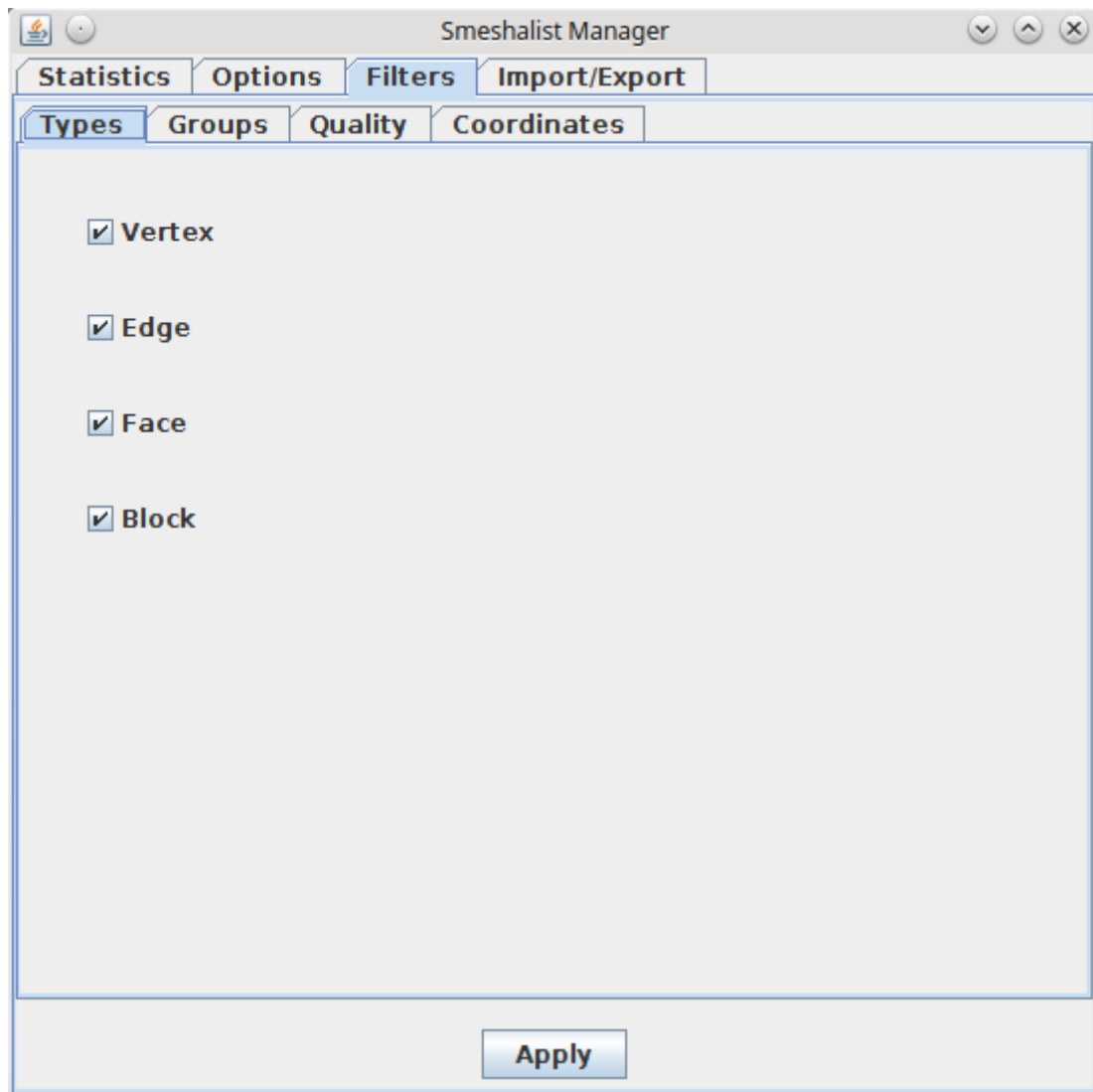
Rysunek 10 Smeshalist Manager - Opcje

Filters

Zakładka filtrów oferuje użytkownikowi możliwość filtrowania wyświetlonych struktur z wykorzystaniem czterech rodzajów filtrów. Zmiany w filtrach należy zatwierdzić przyciskiem *Apply*, co spowoduje uruchomienie filtrowania.

Types

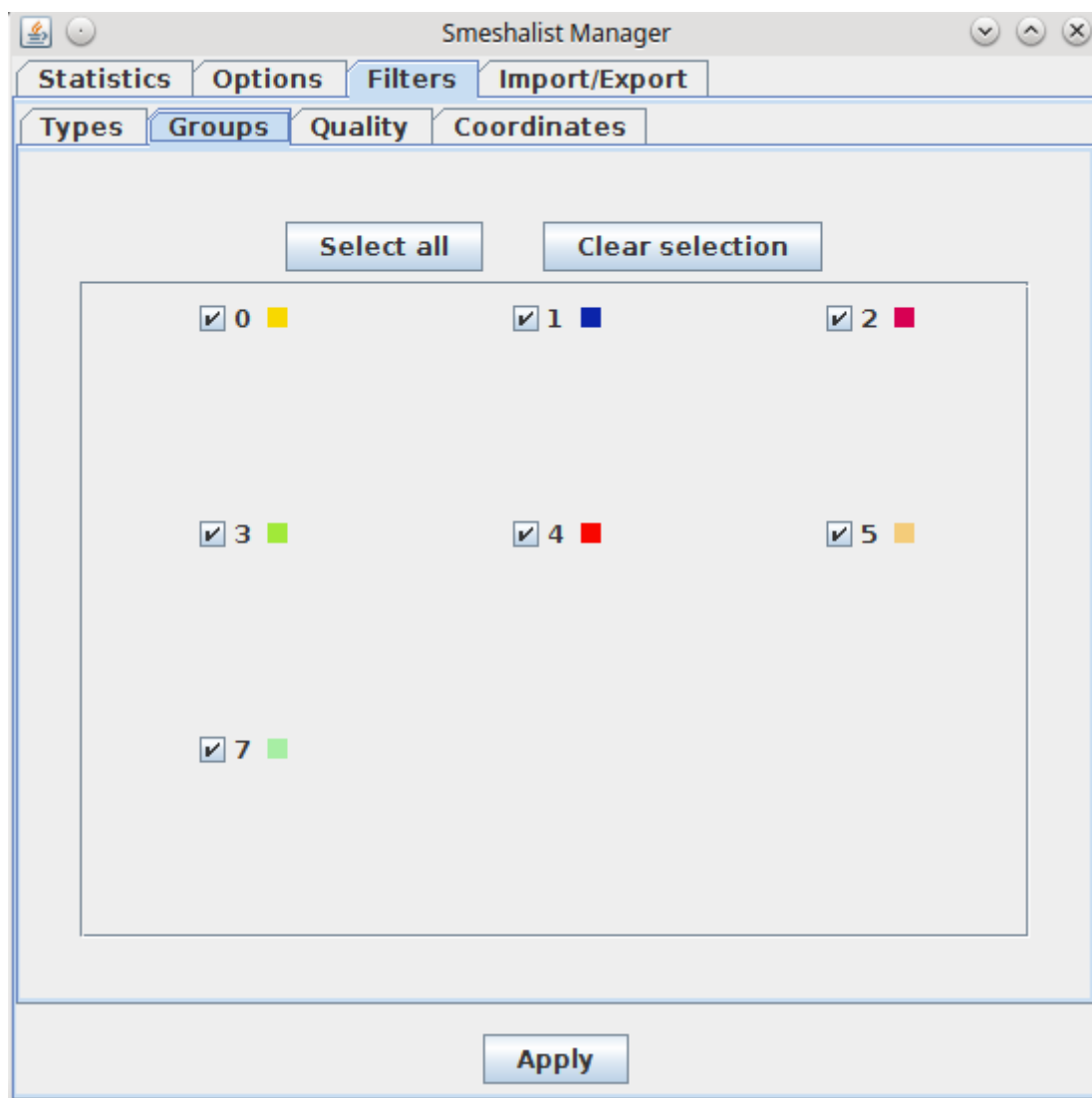
Umożliwia filtrowanie po typie struktury (*Vertex*, *Edge*, *Face*, *Block*).



Rysunek 11 Smeshalist Manager - Filtr Typy

Groups

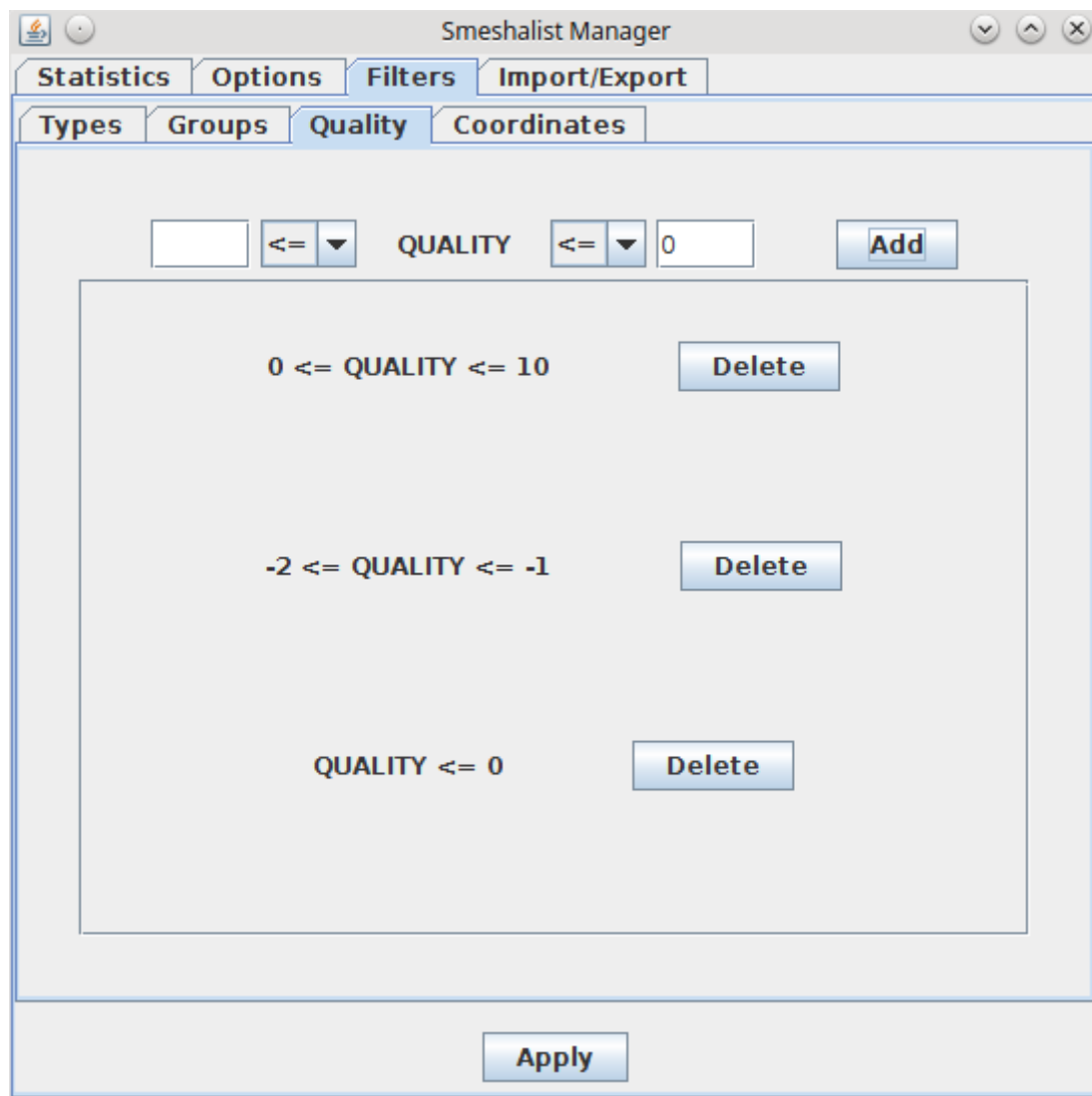
Umożliwia filtrowanie po ID grupy.



Rysunek 12 Smeshalist Manager - Filtr Grupy

Quality

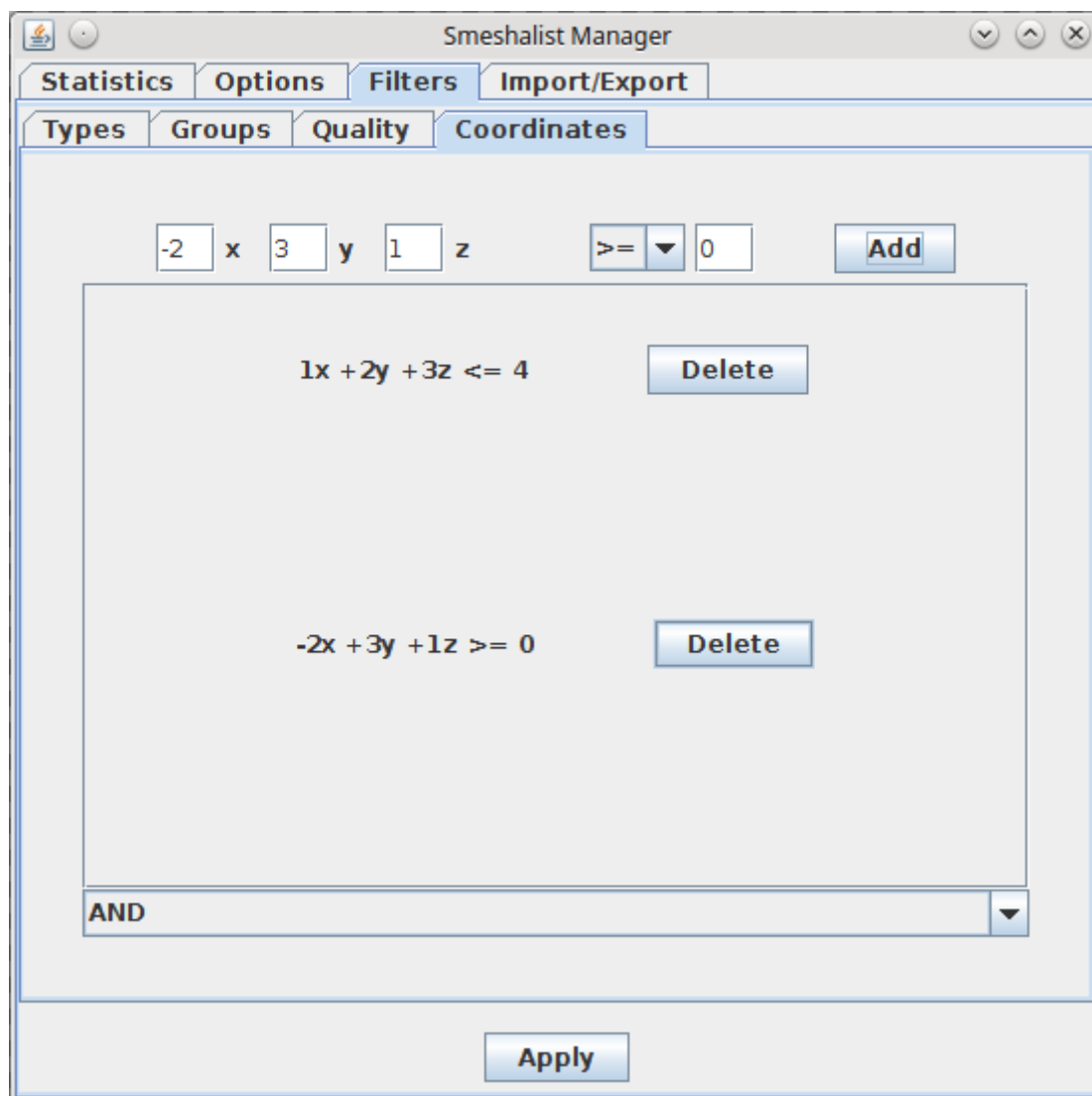
Umożliwia filtrowanie po właściwości Quality każdej struktury. Użytkownik dodaje warunek przyciskiem *Add*. Wszystkie warunki są łączone spójnikiem LUB. Aby usunąć odpowiedni warunek należy użyć odpowiadającemu mu przycisku *Delete*.



Rysunek 13 Smeshalist Manager - Filtr Jakość

Coordinates

Umożliwia filtrowanie po współrzędnych struktur. Użytkownik dodaje warunek przyciskiem *Add*. Aby usunąć odpowiedni warunek należy użyć odpowiadającemu mu przycisku *Delete*. Warunki mogą być łączone spójnikiem I albo LUB. Filtr działa tylko na struktury, które w całości spełniają warunek (a nie np. tylko jeden z wierzchołków ściany).

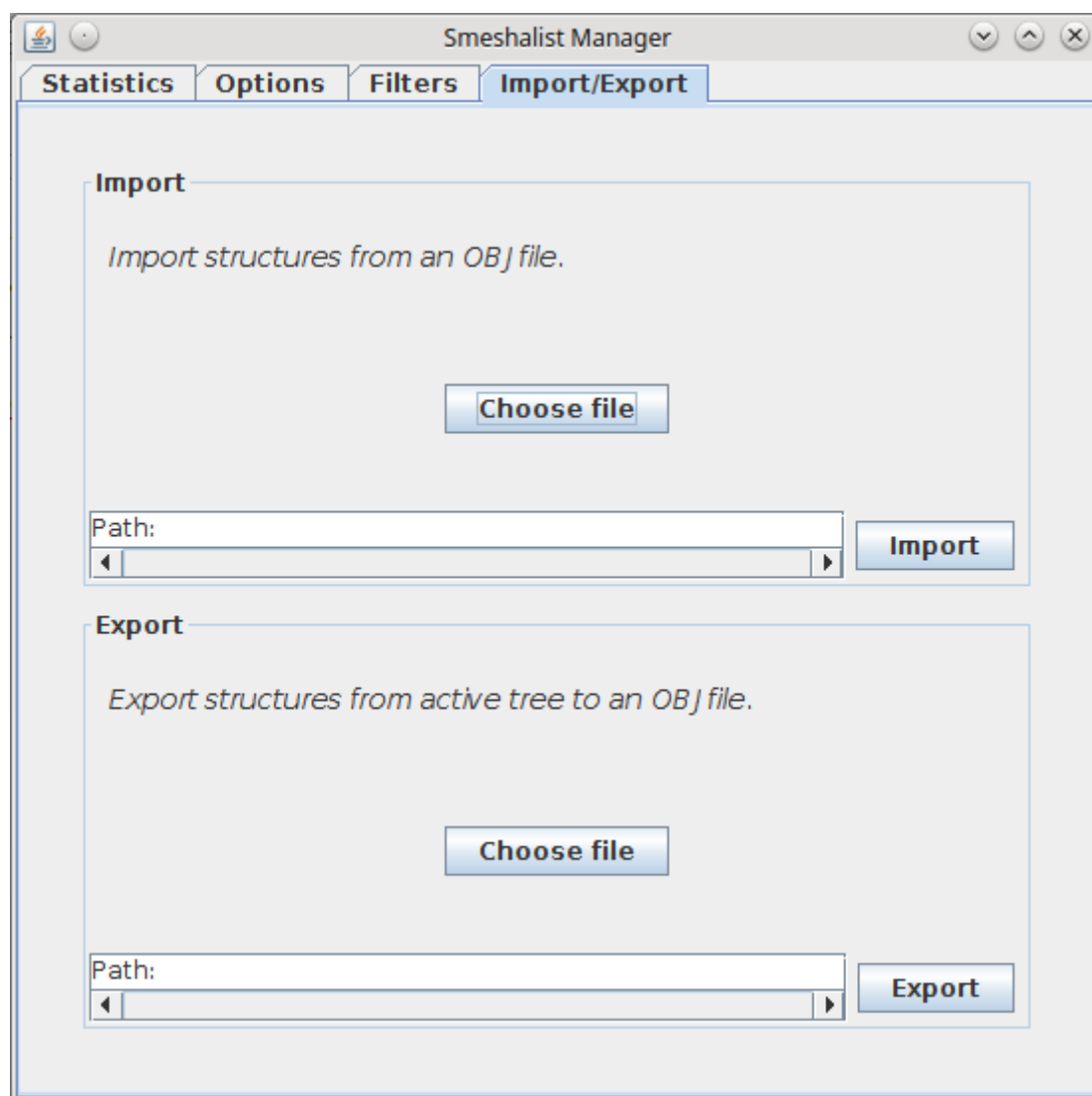


Rysunek 14 Smeshalist Manager - Filtr Współrzędne

Import/Export

Zakładka *Import/Export* umożliwia importowanie danych zapisanych w pliku .obj do aktywnego drzewa struktur oraz na eksport aktualnie widocznych struktur do pliku.

W obu przypadkach należy określić docelową ścieżkę pliku, poprzez kliknięcie przycisku *Choose file*.



Rysunek 15 Smeshalist Manager - Import/Export

Spis ilustracji

Rysunek 1 Przykładowa struktura pliku konfiguracyjnego	6
Rysunek 2 Obsługa zdarzenia breakpoint w oknie Smeshalist Manager	12
Rysunek 3 Obsługa zdarzenia brakepoint w oknie Smeshalist Manager	17
Rysunek 4 Obsługa zdarzenia breakpoint w oknie Smeshalist Manager	20
Rysunek 5 Widok - przybliżanie	23
Rysunek 6 Widok - przesuwanie.....	23
Rysunek 7 Widok - przesuwanie (2)	24
Rysunek 8 Widok - rotacja.....	24
Rysunek 9 Smeshalist Manager - Statystyki	25
Rysunek 10 Smeshalist Manager - Opcje.....	26
Rysunek 11 Smeshalist Manager - Filtr Typy.....	27
Rysunek 12 Smeshalist Manager - Filtr Grupy	28
Rysunek 13 Smeshalist Manager - Filtr Jakość	29
Rysunek 14 Smeshalist Manager - Filtr Współrzędne	30
Rysunek 15 Smeshalist Manager - Import/Export	31