AKADEMIA GÓRNICZO-HUTNICZA

IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ INFORMATYKI, ELEKTRONIKI I TELEKOMUNIKACJI



DOKUMENTACJA TECHNICZNA

**NARZĘDZIE DO WIZUALIZACJI SIATEK TRÓJWYMIAROWYCH**

WOJCIECH DYMEK

KATARZYNA GŁĄB

KATARZYNA KONIECZNA

EWA MARCZEWSKA

OPIEKUN:

Dr inż. Tomasz Jurczyk

Spis treści

[Dziedzina problemu 3](#_Toc469252725)

[Siatka trójwymiarowa 3](#_Toc469252726)

[Stos technologiczny 4](#_Toc469252727)

[OpenGL 4](#_Toc469252728)

[GLU 4](#_Toc469252729)

[GLUT 4](#_Toc469252730)

[Protocol Buffers 4](#_Toc469252731)

[Architektura aplikacji 4](#_Toc469252732)

[Opis modułów 5](#_Toc469252733)

[Moduł komunikacji CORE 5](#_Toc469252734)

[Moduł struktur CORE 6](#_Toc469252735)

[Moduł konfiguracji użytkownika 7](#_Toc469252736)

[Moduł filtracji CORE 7](#_Toc469252737)

[Moduł wizualizacji CORE 9](#_Toc469252738)

[Moduł importu / eksportu 9](#_Toc469252739)

[Smeshalist Manager 9](#_Toc469252740)

[Opis API klienckiego 10](#_Toc469252741)

[Spis ilustracji 11](#_Toc469252742)

# Dziedzina problemu

## Siatka trójwymiarowa

# Stos technologiczny

## OpenGL

Jest to specyfikacja opisująca standardy tworzenia grafiki trójwymiarowej. Implementowana jest przez różne języki programowania i dostępna na różnych platformach. Pozwala na tworzenie animacji z wykorzystaniem obiektów trójwymiarowych.

## GLU

Biblioteka dostarczająca interfejs wyższego poziomu dla funkcjonalności OpenGL.

## GLUT

Niezależny zbiór narzędzi implementujący interfejs programistyczny OpenGL'a do dostarczania aplikacji okienkowych.

## Protocol Buffers

Narzędzie umożliwiające szybką i prostą serializację ustrukturyzowanych danych, którego zaletą jest niezależność od języka programowania i platformy systemowej.

# Architektura aplikacji

Struktury.png

Rysunek Architektura aplikacji

# Opis modułów

## Moduł komunikacji CORE

Moduł zapewniający komunikację pomiędzy wszystkimi modułami systemu.

Do jego głównych zadań należą:

* odbiór wiadomości oraz przekształcanie ich na wewnętrzne obiekty reprezentujące elementy przechowywane w głównej strukturze danych
* przesyłanie statystyk dotyczących ilości przesłanych oraz widocznych elementów do modułu Smeshalist Manager
* pośredniczenie w przekazywaniu informacji pomiędzy modułem filtracji a modułem Smeshalist Manager
* obsługa wiadomości związanych z interakcją pomiędzy systemem a użytkownikiem

Komunikacja - diagram klas.png

Rysunek Moduł komunikacji - diagram klas

Najważniejsza część logiki modułu znajduje się w klasie *AbstractServer*. Odpowiada ona za utrzymywanie połączenia z aplikacją kliencką, jak i oknem *Smashalist Manager’a*. Zawiera również logikę potrzebną do deserializacji wiadomości na obiekty wewnętrzne, zapewnia odpowiednią sekwencję wywołań metod odpowiedzialnych za filtrowanie oraz aktualizację statystyk.

Klasy dziedziczące z *AbstractServer*, tj. *LinuxServer* oraz *WindowsServer*, dostarczają implementacji metod służących do przesyłania danych poprzez gniazda systemowe. Takie rozwiązanie było konieczne ze względu na znaczne różnice w implementacji oraz API gniazd dla obsługiwanych systemów.  
Zastosowanie klasy abstrakcyjnej pozwoliło na wyniesienie całej logiki obsługi wiadomości do wspólnej klasy.

## Moduł struktur CORE

Moduł odpowiedzialny za przechowywanie poszczególnych elementów składających się na siatkę. Dostarcza hierarchiczną strukturę umożliwiającą optymalną wizualizację, filtrację i odpowiednie dodawanie obiektów według rodzajów i grup. Ponadto generuje statystyki odnośnie przechowywanych i widocznych elementów.

Struktury - diagram klas.png

Rysunek Moduł struktur - diagram klas

Podstawową logikę dostarcza klasa *Data*. Zapewnia ona interfejs pozwalający na dodawanie pojedynczych struktur, bądź ich kolekcji oraz umożliwia pobieranie statystyk.

Ważnym rozszerzeniem klasy *Data* jest klasa abstrakcyjna *AbstractDataTree*. Synchronizuje ona dostęp do instancji drzewa struktur oraz zarządza jej kolekcją. Udostępnia również bardziej rozbudowane API, pozwalające na przefiltrowanie danych we wszystkich drzewach struktur.  
Klasy *LinuxDataTree* oraz *WindowsDataTree* dostarczają konkretne implementacje metod *LOCK* oraz *UNLOCK*.

## Moduł konfiguracji użytkownika

Moduł, którego zadaniem jest odczyt konfiguracji użytkownika dostarczonej w pliku *user.config.xml*. W przypadku braku któregokolwiek parametru konfiguracji ustawiana jest wartość domyślna.

Konfiguracja użytkownika - diagram klas.png

Rysunek Moduł konfiguracji użytkownika - diagram klas

Odczyt konfiguracji następuje tylko raz, w momencie tworzenia nowego obiektu *UserPreferencesManager* (tj. przy pierwszym wywołaniu metody *getInstance*). Obiekt ten zapewnia wygodne api do pobierania określonych parametrów oraz odpowiada za przechowywania, bądź wyliczanie, wartości domyślnej.  
Logika parsera zawarta jest w klasie *XMLParser*.

## Moduł filtracji CORE

Odpowiada za przechowywanie oraz zarządzanie zestawami filtrów jak również za filtrację.

Obsługiwane rodzaje filtrów:

* typ geometrii (Vertex, Edge, Face, Block)
* numer grupy
* wartość własności "quality"
* współrzędne

Filtry - diagram klas.png

Rysunek Moduł filtracji - diagram klas

Główne klasy modułu (*CoordinatesFilter*, *GroupsFilter*, *QualityFilter*, *TypesFilter*) zawierają logikę pozwalającą na optymalne przechodzenie drzewa struktur – wymaga to odpowiedniej sekwencji wywołań poszczególnych rodzajów filtrów, za które odpowiada moduł komunikacji CORE. Każda z tych klas zawiera kolekcję prostych filtrów zdefiniowanych przez użytkownika. Ze względu na charakter filtru, bądź też wybór użytkownika, filtry te są łączone spójnikiem *OR* lub *AND*.

Klasa *CoordinatesFilter* zawiera dodatkowo logikę pozwalającą na wyrysowanie poszczególnych filtrów na współrzędne zdefiniowanych przez użytkownika na ekranie. Ze względu na stosunkowo duży stopień skomplikowania, część logiki została wydzielona do klasy *CMathUtils*.

## Moduł wizualizacji CORE

Odpowiada za stworzenie instancji okna, obsługę zdarzeń, głównej pętli programu i sterowanie położeniem kamery.

Ze względu na swój charakter, moduł zawiera również logikę inicjalizującą i uruchamiającą wszystkie pozostałe moduły.

## Moduł importu / eksportu

Moduł udostępnia funkcjonalność eksportowania aktywnego drzewa struktur oraz importu struktur z formatu OBJ.

Import Eksport - diagram klas.png

Rysunek Moduł importu / eksportu - diagram klas

## Smeshalist Manager

Moduł odpowiadający za interakcję z użytkownikiem. Realizowany jest w postaci okna napisanego z wykorzystaniem biblioteki Swing języka Java. Umożliwia on użytkownikowi zarządzanie opcjami wyświetlania struktur w module wizualizacji, jak również ustawianie filtrów i prezentowanie statystyk.

# Opis API klienckiego

# Spis ilustracji

[Rysunek 1 Architektura aplikacji 4](#_Toc469252689)

[Rysunek 2 Moduł komunikacji - diagram klas 5](#_Toc469252690)

[Rysunek 3 Moduł struktur - diagram klas 6](#_Toc469252691)

[Rysunek 4 Moduł konfiguracji użytkownika - diagram klas 7](#_Toc469252692)

[Rysunek 5 Moduł filtracji - diagram klas 8](#_Toc469252693)

[Rysunek 6 Moduł importu / eksportu - diagram klas 9](#_Toc469252694)