AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ INFORMATYKI, ELEKTRONIKI I TELEKOMUNIKACJI

KATEDRA INFORMATYKI

PROJEKT INŻYNIERSKI

DOKUMENTACJA PROCESOWA

Narzędzie do wizualizacji siatek trójwymiarowych

Tool for visualization of three-dimensional meshes

Autorzy: *Wojciech Dymek, Katarzyna Głąb, Katarzyna Konieczna, Ewa Marczewska*

Kierunek studiów: *Informatyka*

Opiekun pracy: *dr inż. Tomasz Jurczyk*

Kraków, 2017

Spis treści

Diagramy i scenariusze przypadków użycia

API

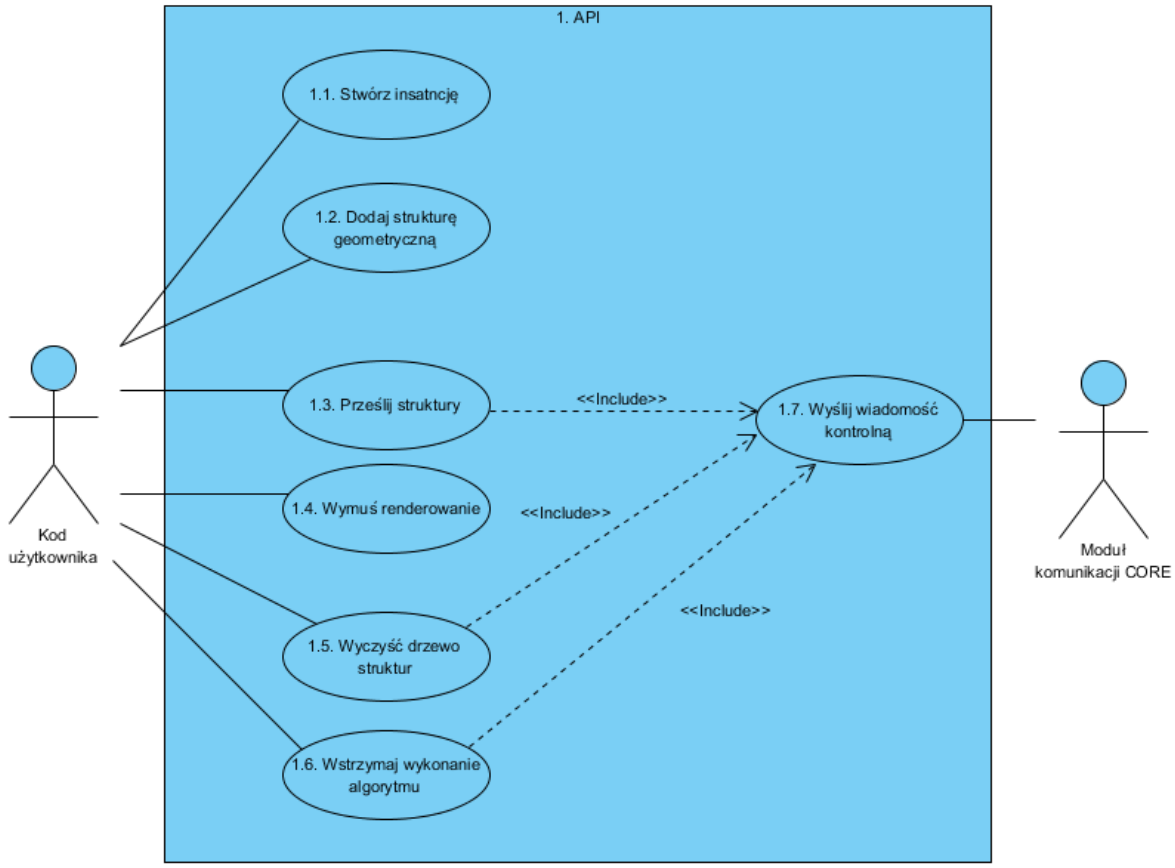

Diagram UC 1 API

Tytuł: 1.1 Stwórz instancję
Aktor główny: Kod użytkownika
Cel: Inicjalizacja środowiska działania aplikacji
Kontekst użycia: Rozpoczęcie pracy z narzędziem Smeshalist
Gwarancja powodzenia: Stworzenie instancji narzędzia w kodzie klienta
Wyzwalacz: Wywołanie konstruktora w kodzie klienta
Scenariusz główny:

1. Wywołanie konstruktora w kodzie klienta
2. Inicjalizacja środowiska działania aplikacji

Tytuł: 1.2. Dodaj strukturę geometryczną
Aktor główny: Kod użytkownika
Cel: Dostarczenie aplikacji elementów do przetwarzania
Kontekst użycia: Inicjalizacja struktur geometrycznych potrzebnych do sprawdzenia działania algorytmu
Gwarancja powodzenia: Struktury dodane do bufora danych
Wyzwalacz: Wywołanie w kodzie klienta metody API odpowiedzialnej za dodanie odpowiedniej struktury geometrycznej
Scenariusz główny:

1. Wywołanie odpowiedniej metody API udostępnianej przez aplikację
2. Zbudowanie danej struktury w formacie proto
3. Dodanie struktury do bufora danych do wysłania

Tytuł: 1.3. Prześlij struktury
Aktor główny: Kod użytkownika
Cel: Przesłanie elementów do modułu komunikacji CORE
Kontekst użycia: Zakończenie dodawania pewnej ilości struktur i zlecenie ich przesłania
Gwarancja powodzenia: Struktury są dostarczone do modułu komunikacji CORE odpowiedzialnej za ich wyświetlenie
Wyzwalacz: Wywołanie w kodzie klienta metody API odpowiedzialnej za przesłanie struktur do modułu komunikacji CORE
Scenariusz główny:

1. Wywołanie odpowiedniej metody API wymuszającej przesłanie danych
2. Przesłanie wiadomości kontrolnej do modułu komunikacji
3. Otrzymanie od modułu komunikacji wiadomości kontrolnej z potwierdzeniem
4. Budowanie paczki danych
5. Przesłanie metadanych do modułu komunikacji CORE
6. Przesłanie paczki danych do modułu komunikacji CORE
7. Otrzymanie od modułu komunikacji potwierdzenia odebrania danych
8. Wykonanie punktów od 4 do 7 do momentu wysłania wszystkich danych zgromadzonych w buforze

Scenariusze alternatywne:

3.1. Brak otrzymania wiadomości kontrolnej z potwierdzeniem od modułu komunikacji
3.2. Zalogowanie błędu aplikacji do pliku
3.3. Przerwanie działania API
7.1. Brak otrzymania potwierdzenia odebrania danych od modułu komunikacji CORE
7.2. Zalogowanie błędu aplikacji do pliku
7.3. Przerwanie działania API

Tytuł: 1.4. Wymuś renderowanie
Aktor główny: Kod użytkownika
Cel: Wymuszenie wyrenderowania struktur znajdujących się w drzewie struktur CORE
Kontekst użycia: Zakończenie dodawania pewnej ilości struktur i zlecenie ich wyrenderowania
Gwarancja powodzenia: Moduł wizualizacji renderuje odpowiedni obraz
Wyzwalacz: Wywołanie w kodzie klienta metody API odpowiedzialnej za renderowanie
Scenariusz główny:

1. Wywołanie odpowiedniej metody API wymuszającej renderowanie obrazu
2. Przesłanie wiadomości kontrolnej do modułu komunikacji CORE

Scenariusze alternatywne:

2.1. Błąd przesyłania wiadomości do modułu komunikacji CORE
2.2. Zalogowanie błędu aplikacji do pliku
2.3. Zakończenie wykonania API

Tytuł: 1.5. Wyczyść drzewo struktur
Aktor główny: Kod użytkownika
Cel: Wymuszenie wyczyszczenia drzewa struktur CORE
Kontekst użycia: Wyczyszczenie aktualnego widoku w celu powtórzenia wykonania algorytmu
Gwarancja powodzenia: Usunięcie elementów znajdujących się w drzewie struktur CORE
Wyzwalacz: Wywołanie w kodzie klienta metody API odpowiedzialnej za wyczyszczenie drzewa struktur
Scenariusz główny:

1. Wywołanie odpowiedniej metody API wymuszającej wyczyszczenie obrazu
2. Przesłanie wiadomości kontrolnej do modułu komunikacji CORE
3. Otrzymanie wiadomości kontrolnej potwierdzającej zakończenie wykonania operacji z modułu komunikacji CORE

Scenariusze alternatywne:

3.1. Brak otrzymania wiadomości kontrolnej z potwierdzeniem od modułu komunikacji
3.2. Zalogowanie błędu aplikacji do pliku
3.3. Przerwanie działania API

Tytuł: 1.6. Wstrzymaj wykonanie algorytmu
Aktor główny: Kod użytkownika
Cel: Wstrzymanie wykonania algorytmu
Kontekst użycia: Osiągnięcie w algorytmie punktu, w którym użytkownik chce zatrzymać jego wykonanie, by móc zdecydować czy go kontynuować
Gwarancja powodzenia: Zatrzymanie wykonywania algorytmu do momentu otrzymania odpowiedzi użytkownika
Wyzwalacz: Wywołanie w kodzie klienta metody API odpowiedzialnej za wstrzymanie wykonania algorytmu
Scenariusz główny:

1. Wywołanie odpowiedniej metody API wymuszającej wstrzymanie wykonania algorytmu
2. Przesłanie wiadomości kontrolnej do modułu komunikacji CORE
3. Otrzymanie wiadomości typu REJECT od modułu komunikacji CORE
4. Zakończenie wykonania API

Scenariusze alternatywne:

2.1. Błąd przesyłania wiadomości do modułu komunikacji CORE
2.2. Zalogowanie błędu aplikacji do pliku
2.3. Zakończenie wykonania API
3.1. Otrzymanie wiadomości typu CONTINUE
3.2. Kontynuacja wykonywania algorytmu

Moduł komunikacji CORE

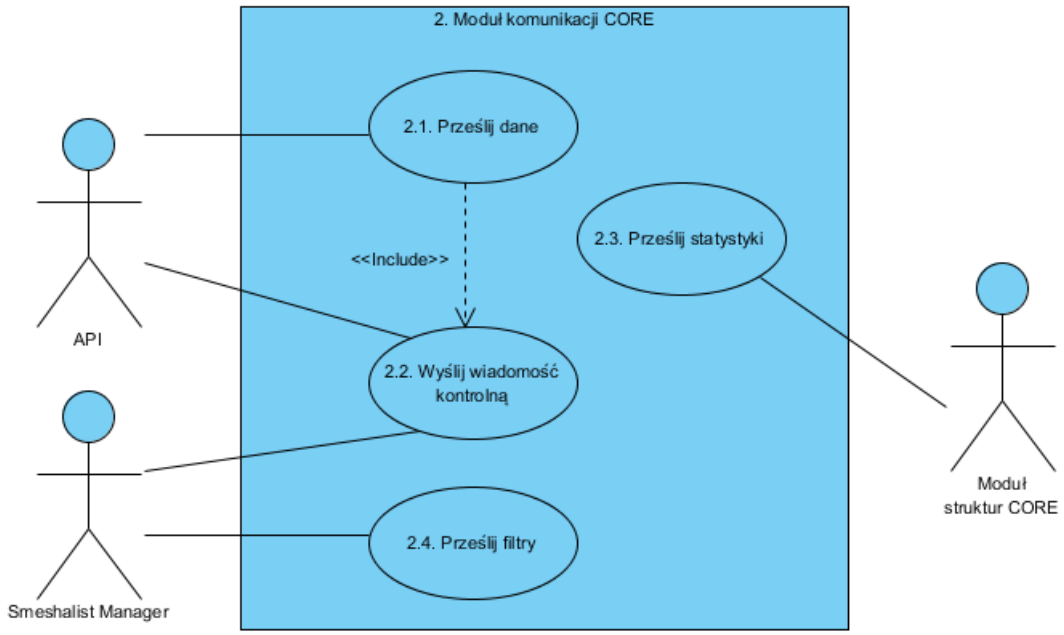

Diagram UC 2 Moduł komunikacji CORE

Tytuł: 2.1. Prześlij dane
Aktor główny: API
Aktor poboczny: moduł komunikacji CORE
Cel: Przesłanie dodanych z kodu użytkownika danych do modułu komunikacji CORE
Kontekst użycia: Chęć wizualizacji pewnego etapu algorytmu przez użytkownika
Gwarancja powodzenia: Dane poprawnie przesłane do modułu komunikacji
Wyzwalacz: Wywołanie w kodzie klienta metody API odpowiedzialnej za przesłanie danych zgromadzonych w buforze
Scenariusz główny:

1. Wywołanie odpowiedniej metody API wymuszającej przesłanie danych
2. Przesłanie wiadomości kontrolnej do modułu komunikacji
3. Otrzymanie od modułu komunikacji wiadomości kontrolnej z potwierdzeniem
4. Budowanie paczki danych
5. Przesłanie metadanych do modułu komunikacji CORE
6. Przesłanie paczki danych do modułu komunikacji CORE
7. Otrzymanie od modułu komunikacji potwierdzenia odebrania danych
8. Wykonanie punktów od 4 do 7 do momentu wysłania wszystkich danych zgromadzonych w buforze

Scenariusze alternatywne:

3.1. Brak otrzymania wiadomości kontrolnej z potwierdzeniem od modułu komunikacji
3.2. Zalogowanie błędu aplikacji do pliku
3.3. Przerwanie działania API
7.1. Brak otrzymania potwierdzenia odebrania danych od modułu komunikacji CORE
7.2. Zalogowanie błędu aplikacji do pliku
7.3. Przerwanie działania API

Tytuł: 2.2. Prześlij wiadomość kontrolną
Aktor główny: API, Smeshalist Manager
Cel: Przesłanie wiadomości kontrolnej do modułu komunikacji CORE
Kontekst użycia: Wywołanie którejś z akcji udostępnianej przez API
Gwarancja powodzenia: Poprawne przesłanie wiadomości kontrolnej do modułu komunikacji CORE
Wyzwalacz: Zbudowanie i przesłanie wiadomości kontrolnej
Scenariusz główny:

1. Wywołanie odpowiedniej metody API
2. Zbudowanie wiadomości kontrolnej o odpowiednim typie
3. Przesłanie wiadomości do modułu komunikacji CORE

Scenariusze alternatywne:

3.1. Błąd podczas próby wysłania wiadomości do modułu komunikacji CORE
3.2. Zalogowanie błędu aplikacji do pliku
3.3. Przerwanie działania API

Tytuł: 2.3. Prześlij statystyki
Aktor główny: Moduł struktur CORE
Cel: Przesłanie statystyk do Smeshalist Manager'a
Kontekst użycia: Dodanie bądź zmiana wyświetlania struktur w drzewie
Gwarancja powodzenia: Poprawne przesłanie statystyk do modułu komunikacji CORE
Wyzwalacz: Dodanie struktur, zmiana aktualnie prezentowanego drzewka lub zmiana filtrów
Scenariusz główny:

1. Przejście po drzewie struktur
2. Zliczenie statystyk
3. Pobranie statystyk przez moduł komunikacji CORE

Tytuł: 2.4. Prześlij filtry
Aktor główny: Smeshalist Manager
Cel: Przesłanie filtrów do modułu komunikacji CORE w celu ich dalszego przetworzenia
Kontekst użycia: Filtracja aktualnie widocznego drzewka struktur
Gwarancja powodzenia: Poprawne przesłanie filtrów do modułu komunikacji CORE
Wyzwalacz: Zatwierdzenie zmian w zestawie filtrów
Scenariusz główny:

1. Sprawdzenie filtrów, które zostały zmienione
2. Zbudowanie wiadomości do przesłania
3. Przesłanie wiadomości do modułu komunikacji CORE

Moduł filtracji CORE

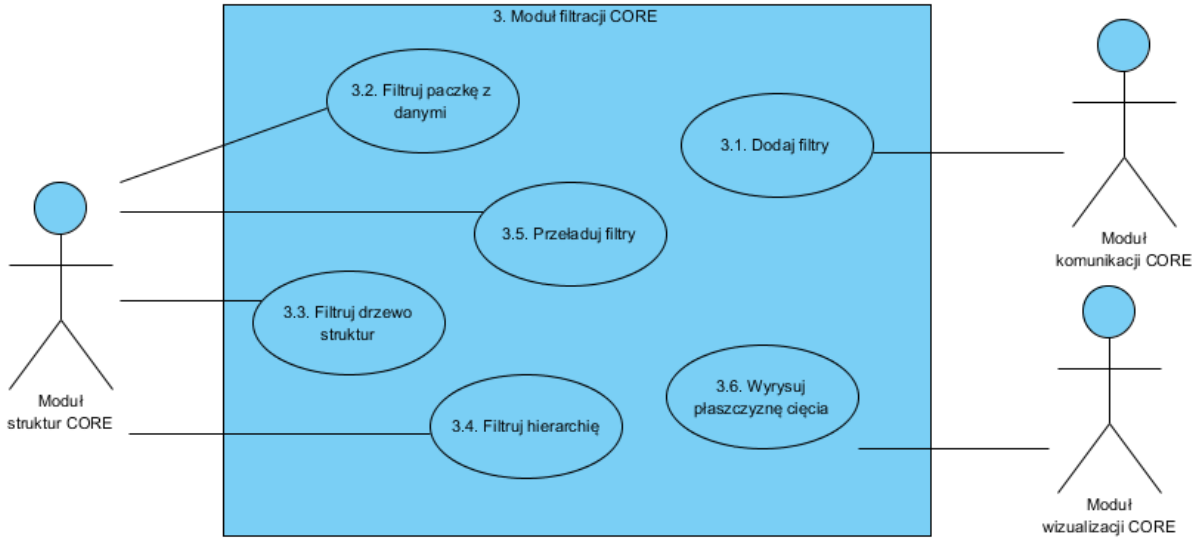

Diagram UC 3 Moduł filtracji CORE

Tytuł: 3.1. Dodaj filtry
Aktor główny: Moduł komunikacji CORE
Cel: Przefiltrowanie drzewa struktur nowym zestawem filtrów
Kontekst użycia: Filtracja aktualnie widocznego drzewka struktur
Gwarancja powodzenia: Dodanie zestawu filtrów
Wyzwalacz: Otrzymanie przez moduł komunikacji CORE filtrów od Smeshalist Manager'a
Scenariusz główny:

1. Przebudowanie filtrów na obiekty wewnętrzne
2. Przesłanie zestawu filtrów

Tytuł: 3.2. Filtruj paczkę z danymi
Aktor główny: Moduł struktur CORE
Cel: Przefiltrowanie danych po współrzędnych i jakości
Kontekst użycia: Dodanie struktur do drzewa
Gwarancja powodzenia: Poprawne ustawienie flag widoczności poszczególnych elementów
Wyzwalacz: Otrzymanie paczki danych przez moduł komunikacji CORE
Scenariusz główny:

1. Przeiterowanie po przedziałach jakości określonych przez użytkownika i ustawienie odpowiednio flagi widoczności
2. Przeiterowanie po filtrach ograniczających współrzędne i ustawienie flagi widoczności

Tytuł: 3.3. Filtruj drzewko struktur
Aktor główny: Moduł struktur CORE
Cel: Przefiltrowanie danych po grupach i typach
Kontekst użycia: Dodanie struktur do drzewa
Gwarancja powodzenia: Poprawne ustawienie flag widoczności poszczególnych struktur agregujących elementy
Wyzwalacz: Otrzymanie paczki danych przez moduł komunikacji CORE
Scenariusz główny:

1. Przeiterowanie po liście odznaczonych przez użytkownika grup i ustawienie odpowiednio flagi widoczności
2. Przeiterowanie po liście odznaczonych typów i ustawienie odpowiednio flagi widoczności

Tytuł: 3.4. Filtruj hierarchię
Aktor główny: Moduł struktur CORE
Cel: Filtracja struktur znajdujących się w drzewie zgodnie z ustawionymi filtrami
Kontekst użycia: Zmiana filtrów w Smeshalist Managerze
Gwarancja powodzenia: Poprawne ustawienie flag widoczności poszczególnych struktur w drzewie
Wyzwalacz: Otrzymanie przez moduł komunikacji CORE filtrów od Smeshalist Manager'a
Scenariusz główny:

1. Przeiterowanie po liście odznaczonych przez użytkownika grup i ustawienie odpowiednio flagi widoczności
2. Przeiterowanie po liście odznaczonych typów i ustawienie odpowiednio flagi widoczności
3. Przeiterowanie po przedziałach jakości określonych przez użytkownika i ustawienie odpowiednio flagi widoczności
4. Przeiterowanie po filtrach ograniczających współrzędne i ustawienie flagi widoczności

Tytuł: 3.5. Przeładuj filtry
Aktor główny: Moduł struktur CORE
Cel: Zapis nowego zestawu filtrów
Kontekst użycia: Zmiana filtrów w Smeshalist Managerze
Gwarancja powodzenia: Dodanie zestawu filtrów do kolekcji znajdujących się w module filtracji CORE
Wyzwalacz: Otrzymanie przez moduł struktur CORE nowego zestawu filtrów
Scenariusz główny:

1. Usunięcie elementów znajdujących się aktualnie w kolekcjach filtrów
2. Dodanie zestawu filtrów do kolekcji znajdujących się w module filtracji CORE

Tytuł: 3.6. Wyrysuj płaszczyznę cięcia
Aktor główny: Moduł wizualizacji CORE
Cel: Wizualizacja filtrów po współrzędnych
Kontekst użycia: Odświeżenie okienka modułu wizualizacji
Gwarancja powodzenia: Poprawnie narysowanie płaszczyzny cięcia
Wyzwalacz: Przejście do następnego wykonania pętli głównej programu

Scenariusz główny:

1. Iteracja po filtrach po współrzędnych
2. Wyrysowanie odpowiednich płaszczyzn cięcia wyznaczonych przez filtry

Moduł struktur CORE

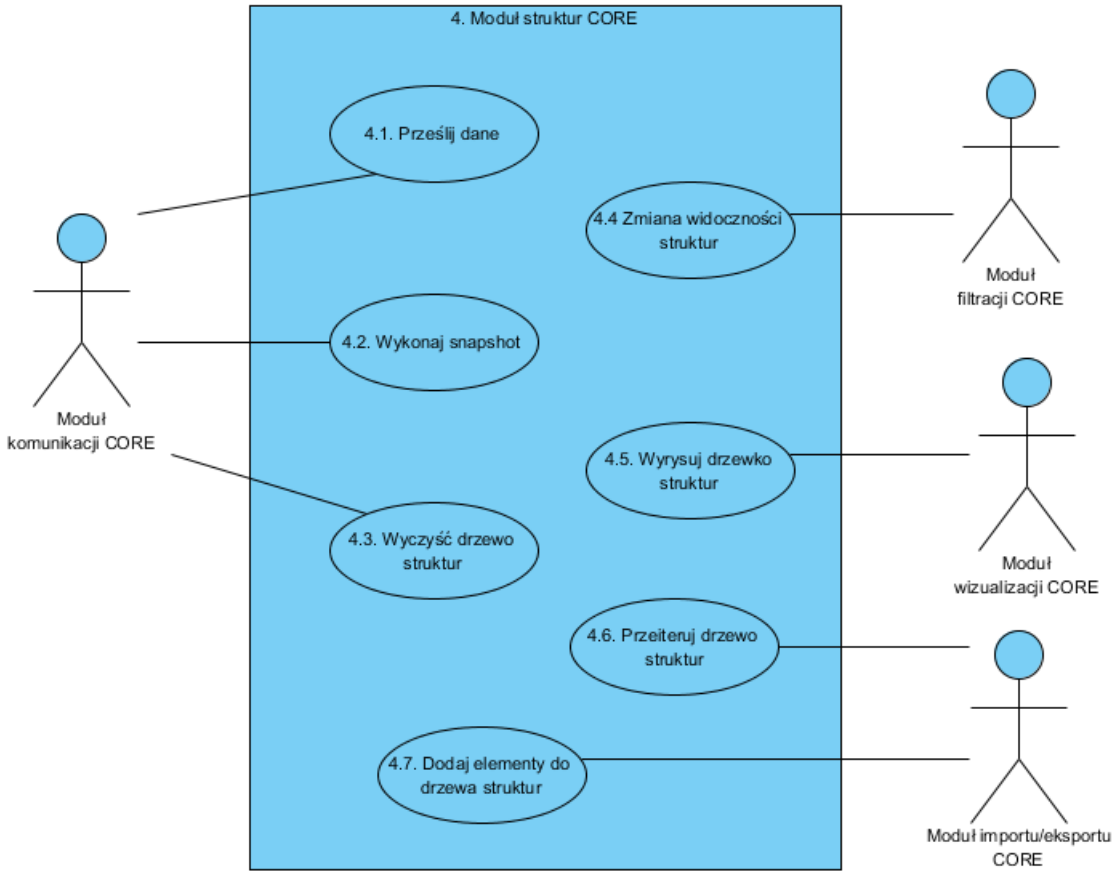

Diagram UC 4 Moduł struktur CORE

Tytuł: 4.1. Prześlij dane
Aktor główny: Moduł komunikacji CORE
Cel: Przesłanie nowych danych do drzewka struktur
Kontekst użycia: Przesłanie z kodu użytkownika nowej paczki danych
Gwarancja powodzenia: Dane są poprawnie przesłane z modułu komunikacji CORE do modułu struktur CORE
Wyzwalacz: Otrzymanie paczek z danymi od API
Scenariusz główny:

1. Uzyskanie wyłącznego dostępu do drzewa
2. Filtrowanie paczki danych
3. Dodanie paczki do drzewa struktur
4. Filtrowanie drzewa struktur
5. Zwolnienie dostępu

Tytuł: 4.2. Wykonaj snapshot
Aktor główny: Moduł komunikacji CORE
Cel: Zapisanie struktury drzewa w pamięci do późniejszego wykorzystania
Kontekst użycia: Otrzymanie ze Smeshalist Manager'a żądania wykonania kopii drzewa
Gwarancja powodzenia: Snapshot drzewa zostaje poprawnie zapisany w pamięci
Wyzwalacz: Otrzymanie wiadomości kontrolnej o odpowiednim typie od Smeshalist Manager'a
Scenariusz główny:

1. Uzyskanie wyłącznego dostępu do drzewa
2. Wykonanie kopii zapisanych danych
3. Stworzenie nowej instancji drzewa i dodanie do niego struktur
4. Zwolnienie dostępu

Tytuł: 4.3. Wyczyść drzewo struktur
Aktor główny: Moduł komunikacji CORE
Cel: Wyczyszczenie drzewa struktur
Kontekst użycia: Otrzymanie ze Smeshalist Manager'a żądania wyczyszczenia aktywnego drzewa
Gwarancja powodzenia: Usunięcie wszystkich elementów drzewa struktur
Wyzwalacz: Otrzymanie wiadomości kontrolnej o odpowiednim typie od Smeshalist Manager'a lub API
Scenariusz główny:

1. Uzyskanie wyłącznego dostępu do drzewa
2. Rekurencyjne usunięcie wszystkich zapisanych elementów
3. Zwolnienie dostępu

Tytuł: 4.4. Zmiana widoczności struktur
Aktor główny: Moduł filtracji CORE
Cel: Ustawienie flag widoczności filtrowanych elementów drzewa struktur
Kontekst użycia: Filtracja struktur
Gwarancja powodzenia: Poszczególne elementy drzewa struktur mają ustawione flagi widoczności zgodnie z przyjętymi filtrami
Wyzwalacz: Otrzymanie ze Smeshalist Manager'a nowego zestawu filtrów
Scenariusz główny:

1. Zmiana stanu flagi widoczności na zadaną przez moduł filtracji CORE

Tytuł: 4.5. Wyrysuj drzewko struktur
Aktor główny: Moduł wizualizacji CORE
Cel: Wizualizacja zawartości drzewa struktur
Kontekst użycia: Wizualizacja algorytmu geometrii obliczeniowej
Gwarancja powodzenia: Poprawnie wyświetlona zawartość drzewa struktur
Wyzwalacz: Wywołanie odpowiedniej metody API
Scenariusz główny:

1. Rekurencyjne przejście hierarchii drzewa
2. Sprawdzenie flagi widoczności aktualnie odwiedzanego węzła drzewa
3. Wyrysowanie struktur zgodnie z ustawieniem flagi widoczności oraz przezroczystości

Tytuł: 4.6. Przeiteruj drzewo struktur
Aktor główny: Moduł importu/eksportu CORE
Cel: Iteracja po elementach drzewa struktur
Kontekst użycia: Eksport zawartości drzewa struktur do pliku
Gwarancja powodzenia: Zapisanie elementów drzewa struktur w module importu/eksportu CORE
Wyzwalacz: Wywołanie odpowiedniej metody przez moduł komunikacji CORE
Scenariusz główny:

1. Iteracja po widocznych grupach struktur
2. Iteracja po widocznych typach struktur
3. Pobranie elementu
4. Zapisanie elementu do pliku

Tytuł: 4.7. Dodaj elementy do drzewa struktur
Aktor główny: Moduł importu/eksportu CORE
Cel: Uzupełnienie drzewa struktur o zaimportowane z pliku elementy
Kontekst użycia: Import struktur z pliku
Gwarancja powodzenia: Poprawne dodanie elementów z pliku OBJ do drzewa struktur
Wyzwalacz: Kliknięcie przycisku *Import* w oknie Smeshalist Manager
Scenariusz główny:

1. Otwarcie pliku
2. Parsowanie pliku
3. Dodanie odpowiednich elementów do drzewa struktur

Scenariusze alternatywne:

1.1. Błąd przy próbie otwarcia pliku
1.2. Przerwanie importu i zalogowanie informacji o błędzie
2.1. Błąd przy parsowaniu pliku
2.2. Przerwanie importu i zalogowanie informacji o błędzie

Smeshalist Manager

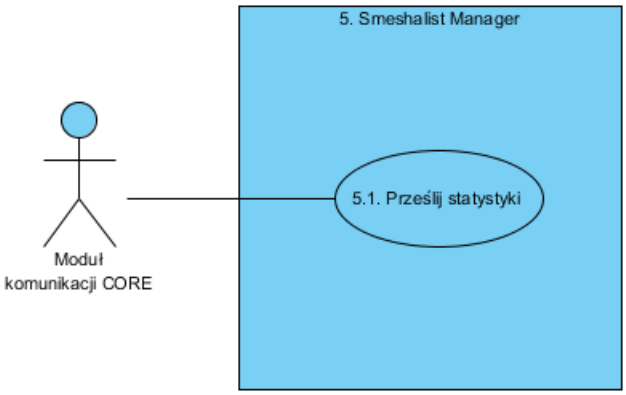

Diagram UC 5 Smeshalist Manager

Tytuł: 5.1. Prześlij statystyki
Aktor główny: Moduł komunikacji CORE
Cel: Przesłanie zaktualizowanych statystyk do Smeshalist Manager'a
Kontekst użycia: Zmiana w drzewie struktur
Gwarancja powodzenia: Statystyki zostają poprawnie przesłane do Smeshalist Manager'a
Wyzwalacz: Dodanie nowych struktur do drzewa, zmiana filtrów, bądź zmiana aktualnie wyświetlanego drzewa
Scenariusz główny:

1. Pobranie statystyk z modułu struktur
2. Zbudowanie odpowiedniego typu wiadomości
3. Przesłanie wiadomości do Smeshalist Manager'a

Moduł importu/eksportu CORE

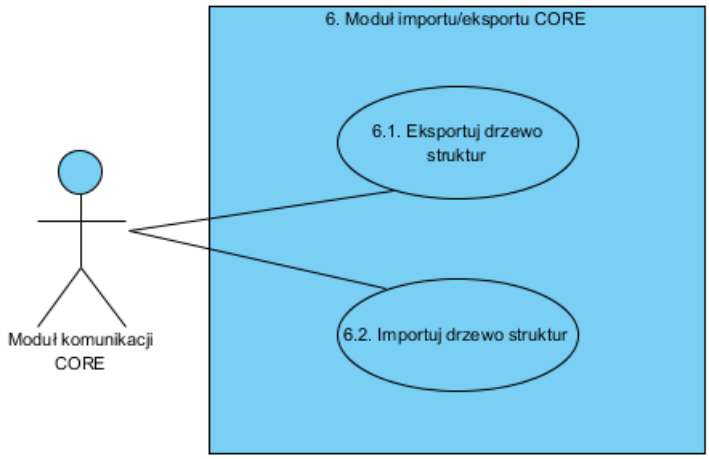

Diagram UC 6 Moduł importu/eksportu CORE

Tytuł: 6.1. Eksportuj drzewo struktur
Aktor główny: Moduł komunikacji CORE
Cel: Zapisanie zawartości drzewa struktur do pliku
Kontekst użycia: Udostępnienie elementów drzewa struktur aplikacjom zewnętrznym
Gwarancja powodzenia: Poprawne wyeksportowanie zawartości drzewa struktur do pliku OBJ
Wyzwalacz: Kliknięcie przycisku *Export* w oknie Smeshalist Manager
Scenariusz główny:

1. Otrzymanie wiadomości o odpowiednim typie od Smeshalist Managera
2. Przetworzenie wiadomości
3. Wywołanie odpowiedniej metody modułu importu/eksportu CORE

Tytuł: 6.2. Importuj drzewo struktur
Aktor główny: Moduł komunikacji CORE
Cel: Wczytanie do drzewa struktur elementów zapisanych w pliku
Kontekst użycia: Umożliwienie wizualizowania struktur wyeksportowanych zewnętrznym narzędziem
Gwarancja powodzenia: Poprawne dodanie elementów z pliku OBJ do drzewa struktur
Wyzwalacz: Kliknięcie przycisku *Import* w oknie Smeshalist Manager
Scenariusz główny:

1. Otrzymanie wiadomości o odpowiednim typie od Smeshalist Managera
2. Przetworzenie wiadomości
3. Wywołanie odpowiedniej metody modułu importu/eksportu CORE

Analiza ryzyka

Lp.	Ryzyko	P-stwo	Skutek	Waga
1.	Problemy z zapewnieniem działania aplikacji na systemie operacyjnym Windows.	30%	Brak przenośności aplikacji	10
2.	Nieumiejętność posługiwania się narzędziem Protocol Buffers	5%	Brak komunikacji między modułami aplikacji	10
3.	Problemy z zapewnieniem odpowiedniej wydajności aplikacji	10%	Wolne działanie aplikacji	3
4.	Błędne lub niewystarczające założenia na etapie projektowania rozwiązania	15%	Poświęcenie dodatkowego czasu na poprawę błędnych rozwiązań	2
5.	Zbyt mała ilość czasu na implementację pełnej wymaganej funkcjonalności	5%	Ograniczona funkcjonalność aplikacji	2

Harmonogram

Lp.	Działania	Data
1.	Prototyp podstawowych funkcjonalności aplikacji działający na systemie Linux	Pierwsza połowa maja 2016
2.	Rozbudowa API w Javie o wszystkie podstawowe struktury geometryczne	Druga połowa maja 2016
3.	Zapewnienie komunikacji dwukierunkowej, dalszy rozwój struktur	Koniec maja 2016
4.	Implementacja podstawowej filtracji danych	Pierwsza połowa czerwca 2016
5.	Przeniesienie podstawowej funkcjonalności pod system Windows	Lipiec 2016
6.	Poprawa wydajności i aspektów wizualnych aplikacji	Sierpień 2016
7.	Implementacja funkcjonalności zapewniającej import/eksport danych	Wrzesień 2016
8.	Dopracowanie funkcjonalności aplikacji, poprawa ewentualnych błędów	Październik - listopad 2016

Metodyka

Metodyka zastosowana przez nas w projekcie to model przyrostowy. Prace zostały podzielone na trzy przyrosty, w których każdy kolejny dostarczał nowej funkcjonalności narzędzia. Początkowe ustalenia z klientem dostarczyły informacji o wymaganej funkcjonalności. Każda funkcja miała przydzielony odpowiedni priorytet.

Podczas pierwszego przyrostu zaimplementowany został prototyp narzędzia działający na platformie Linux dostarczający podstawowej funkcjonalności. W kolejnych przyrostach funkcjonalność była poszerzana o nowe aspekty oraz jej działanie zostało przeniesione na platformę Windows.

Przyjęcie takiej metodyki pozwoliło nam na bieżące monitorowanie postępów prac przez klienta oraz szybką reakcję na zgłoszone przez niego uwagi.

Role i podział prac w zespole

Członek zespołu	Zakres prac
Wojciech Dymek	• Implementacja modułu komunikacji CORE • Implementacja poruszania się po scenie • Implementacja modułu filtracji • Implementacja parsowania pliku konfiguracyjnego
Katarzyna Głąb	• Implementacja drzewa struktur • Implementacja statystyk • Implementacja modułu wizualizacji
Katarzyna Konieczna	• Implementacja okienka Smeshalist Manager wraz z przesyłaniem odpowiednich wiadomości do modułu komunikacji CORE • Implementacja interfejsu programistycznego w języku C++ na platformę Linux oraz Windows
Ewa Marczewska	• Implementacja interfejsu programistycznego w języku Java oraz Python • Implementacja importu oraz eksportu drzewa struktur do formatu .obj

Timeline

Przyrost	Krótki opis
I	Pierwszy przyrost obejmował określenie wymagań klienta. Następnie została zaprojektowana architektura systemu oraz wydzielone zostały jego moduły. W kolejnym kroku zaimplementowany został prototyp narzędzia działający pod systemem Linux wraz z API w Javie. Celem było wizualizacja podstawowych typów struktur na przestrzeni trójwymiarowej oraz ich filtracja. W etapie powstał również prototyp okienka Smeshalist Manager z ograniczoną funkcjonalnością.
II	Podczas drugiego przyrostu skupiliśmy się na rozwoju interfejsu programistycznego w języku Java wzbogacając jego funkcjonalność o potrzebne metody. Ponadto wykonane zostały prace związane z zapewnieniem wymaganej funkcjonalności w okienku Smeshalist Manager. Dodane zostało również kolorowanie struktur oraz wyeliminowane zostały znalezione błędy. Ponadto wykonane zostały pierwsze próby uruchomienia narzędzia na platformie Windows.
III	W czasie trzeciego przyrostu zmieniony został algorytm przesyłania paczek danych z API do modułu komunikacji CORE. Ponadto zaimplementowane zostały analogiczne interfejsy programistyczne użytkownika w języku Python oraz C++ zarówno na platformę Linux jak i Windows. Co więcej zaimplementowany został import i eksport drzewa struktur do pliku .obj. Dostarczona została również możliwość konfiguracji narzędzia z użyciem pliku konfiguracyjnego. Zaimplementowane zostały kolorowanie struktur po jakości oraz możliwość ustawienia ich przezroczystości. Dodatkowo zaimplementowana została funkcjonalność zrzutów drzewa struktur (SNAPSHOT). Narzędzie zostało również w pełni przeniesione na platformę Windows. Wszystkie dodane funkcje znalazły swoje odzwierciedlenie w okienku Smeshalist Manager.

Przebieg prac

Przyrost I

Termin

20 czerwca 2016 r.

Cel

Zebranie precyzyjnych wymagań odnośnie funkcjonalności narzędzia. Zaprojektowanie architektury systemu oraz podział na moduły. Implementacja prototypu narzędzia działającego pod systemem Linux.

Oczekiwany produkt

- Projekt architektury
- Prototyp zapewniający wizualizację podstawowych struktur geometrycznych oraz ich filtrację działający na platformie Linux.

Zrealizowano

- Projekt architektury opierający się na podziale narzędzia na moduły:
    - API
    - moduł komunikacji CORE – wykorzystanie protokołu UDP z narzędziem Protocol Buffers
    - moduł struktur CORE
    - moduł filtracji CORE
    - moduł wizualizacji CORE
    - Smeshalist Manager
- Implementacja wydzielonych modułów zapewniających podstawową funkcjonalność narzędzia tj.:
    - wizualizacja podstawowych typów struktur: wierzchołek, krawędź, ściana, czworościan
    - filtrowanie struktur wg: typu struktury, numeru grupy, do której należy, jakości oraz współrzędnych
    - interfejs programistyczny w języku Java w zakresie zdefiniowana wewnętrznego formatu struktur oraz komunikacji z modułem komunikacji CORE

- okienko Smeshalist Manager w zakresie wyświetlania statystyk

Podsumowanie

Początkowe problemy z wyborem sposobu realizacji komunikacji pomiędzy modułami systemu zostały rozwiązane. Zrealizowane zostały założone na początku przyrostu cele.

Przyrost II

Termin

31 sierpnia 2016 r.

Cel

Przeniesienie podstawowej funkcjonalności na platformę Windows. Dopracowanie aspektów wizualnych aplikacji. Dalszy rozwój funkcjonalności Java API oraz okienka Smeshalist Manager. Optymalizacja działania narzędzia.

Oczekiwany produkt

- działający w zakresie podstawowej funkcjonalności na platformie Windows
- ukończony interfejs programistyczny w języku Java
- optymalnie działająca filtracja struktur
- uzupełnienie funkcjonalności okienka Smeshalist Manager i poprawa błędów w działaniu

Zrealizowano

- Ukończony został interfejs programistyczny w języku Java. Zaimplementowany został algorytm przesyłania paczek z danymi do modułu komunikacji CORE. Ponadto zaimplementowane zostały metody przerwania wykonania algorytmu i wymuszenia renderowania.
- Zmiana implementacji drzewa struktur w celu zoptymalizowania działania filtrowania. Możliwość pobierania elementów drzewa na podstawie ID grupy, do której należą. Zapewnienie bezpiecznego wielowątkowo wykonania algorytmu filtrowania. Poprawa błędów przy filtrowaniu.
- Dodanie możliwości definiowania i zarządzania filtrami w okienku Smeshalist Manager. Zapewnienie interaktywnego wykonania algorytmu (dodanie opcji *Breakpoint*). Ukształtowanie modelu komunikacji wewnętrznej. Poprawienie wykrytych błędów.

- Dostarczenie możliwości automatycznego zbudowania projektu na platformie Linux.
- W zakresie uruchomienia narzędzia na platformie Windows został wydzielony fragment kodu zapewniający komunikację UDP z użyciem socketów platformy Windows.

Podsumowanie

Większość założonych celów została osiągnięta, jednak wyniknęły problemy podczas próby automatycznego budowania narzędzia na platformie Windows, dlatego też funkcjonalność ta nie została w pełni zrealizowana.

Przyrost III

Termin

15 grudnia 2016 r.

Cel

Gotowy produkt zapewniający pełną funkcjonalność oczekiwaną przez klienta. Dostarczenie dokumentacji projektu.

Oczekiwany produkt

- intuicyjne i proste w użyciu narzędzie, działające na platformie Linux i Windows
- możliwość konfiguracji pewnych parametrów przez użytkownika
- możliwość importu oraz eksportu drzewa struktur do plików w formacie .obj
- interfejs programistyczny w językach Python i C++ (w przypadku drugiego, na platformę Windows oraz Linux)
- możliwość kolorowania struktur geometrycznych po jakości
- możliwość ustawienia przezroczystości struktur
- możliwość przełączania trybu wizualizacji z 3D na 2D
- wyświetlanie etykiet struktur geometrycznych
- działające, stabilne okienko Smeshalist Manager z możliwością kontroli wyświetlanych zrzutów drzewa struktur, zmiany sposobu kolorowania struktur, przełączania trybu wizualizacji oraz z obsługą funkcjonalności importu i eksportu

Zrealizowano

- Interfejs programistyczny użytkownika w języku Python oraz C++ działający zarówno na platformie Windows jak i Linux.

- Do funkcjonalności API dodany została metoda *clean*() oraz resetowanie ustawień narzędzia.
- Możliwość personalizowania ustawień kolorystycznych narzędzia przez użytkownika w pliku konfiguracyjnym.
- Utworzenie pliku wykonywalnego narzędzia działającego na platformie Windows.
- Poprawki w działaniu modułu wizualizacji, implementacja możliwości przełączenia narzędzia w tryb 2D.
- Funkcję importu oraz eksportu drzewa struktur w formacie obj.
- Opcję wyświetlania etykiet.
- Konieczne do poprawnego działania narzędzia zmiany w oknie Smeshalist Manager.

Podsumowanie

Założone w przyroście cele zostały w całości zrealizowane. Efektem końcowym przyrostu było udostępnienie narzędzia o pełnej założonej w wizji projektu funkcjonalności działającego na platformie Windows oraz Linux. Dostarczona została również ukończona dokumentacja projektu.

Spis ilustracji