

**WYDZIAŁ INFORMATYKI, ELEKTRONIKI I TELEKOMUNIKACJI**

KATEDRA INFORMATYKI

**PROJEKT INŻYNIERSKI**

*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

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia   
4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „ Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór   
w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.”, a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia   
27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.) „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej „sądem koleżeńskim”, oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście, samodzielnie i że nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

*<podpis dyplomanta>*

Spis treści

[Cel i wizja produktu 4](#_Toc471371763)

[Charakterystyka problemu 4](#_Toc471371764)

[Studium wykonalności 4](#_Toc471371765)

[Analiza ryzyka 5](#_Toc471371766)

[Zakres funkcjonalności 5](#_Toc471371767)

[Wymagania funkcjonalne 5](#_Toc471371768)

[Wymagania niefunkcjonalne 6](#_Toc471371769)

[Wybrane aspekty realizacji 7](#_Toc471371770)

[Architektura aplikacji 7](#_Toc471371771)

[Stos technologiczny 8](#_Toc471371772)

[Organizacja pracy 9](#_Toc471371773)

[Metodyka 9](#_Toc471371774)

[Przyrost I 9](#_Toc471371775)

[Przyrost II 9](#_Toc471371776)

[Przyrost III 9](#_Toc471371777)

[Wyniki projektu 10](#_Toc471371778)

[Wyniki, ocena użyteczności 10](#_Toc471371779)

[Propozycje rozwoju systemu 10](#_Toc471371780)

# Cel i wizja produktu

Celem projektu jest stworzenie narzędzia, które w efektywny sposób wykorzysta możliwości współczesnych kart graficznych dla potrzeb renderowanie siatek trójwymiarowych (powierzchniowych oraz objętościowych). Wizualizacja powinna dotyczyć podstawowych typów struktur geometrycznych (wierzchołek, prosta, ściana oraz czworościan). Narzędzie powinno uwzględniać kryteria wizualizacji takie jak ukrywanie oraz kolorowanie elementów w oparciu o ich jakość geometryczną, współrzędne czy też informacje przekazane wraz z przesłaniem struktur do narzędzia np. numer grupy, do której dany element należy.

## Charakterystyka problemu

Wizualizacja siatek trójwymiarowych oraz objętościowych jest nieodłącznym elementem implementacji algorytmów geometrii obliczeniowej. Sprawdzenie poprawności wykonania algorytmu wiąże się z koniecznością wykorzystania zewnętrznego narzędzia dostarczającego możliwości reprezentacji graficznej wyników. Aktualnie dostępne narzędzia tego typu posiadają wiele ograniczeń, głównie wydajnościowych, które wpływają na szybkość oraz komfort pracy programisty.

Opisywane w tym dokumencie narzędzie dostarcza możliwości wydajnej wizualizacji wykonywanego algorytmu bezpośrednio z kodu klienta, również dla siatek o dużej liczbie elementów. Wizualizacja wyniku jest możliwa już podczas kolejnych iteracji algorytmu, co pozwala użytkownikowi na większą kontrolę wykonania programu oraz szybsze wykrycie ewentualnych błędów. Zaawansowani użytkownicy mogą również skorzystać z wielu dodatkowych funkcji takich jak możliwość ukrywania elementów oraz ich odpowiedniego kolorowania, czy też importu i eksportu przesłanych elementów do pliku w formacie .obj, co znacznie ułatwi implementację algorytmu.

## Studium wykonalności

Wykonana na etapie projektowania systemu analiza wymagań funkcjonalnych oraz niefunkcjonalnych pozwala określić, że projekt jest wykonalny w ramach dostępnych zasobów ludzkich, czasowych i technologicznych.

## Analiza ryzyka

Lista potencjalnych zagrożeń, które zostały wykryte na etapie analizy ryzyka została przedstawiona poniżej.

* problemy z zapewnieniem przenośności aplikacji
* nieznajomość wykorzystywanego narzędzi *Protocol Buffers*
* problemy z zapewnieniem wydajności aplikacji przy przesyłaniu dużej ilości elementów
* błędne lub niewystarczające założenia na etapie projektowania rozwiązania
* zbyt mała ilość czasu na implementację pełnej wymaganej funkcjonalności

# Zakres funkcjonalności

W rozdziale przedstawione zostały ustalone z klientem wymagania dotyczące działania narzędzia.

## Wymagania funkcjonalne

* Wizualizacja siatek składających się z podstawowych elementów: wierzchołków, krawędzi, trójkątnych ścian oraz czworościanów
* Elementy mogą posiadać dodatkowe atrybuty:
  + liczba całkowita (identyfikator grupy) wpływająca na filtrowanie oraz kolorowanie elementów
  + liczba rzeczywista z zakresu 0 do 1 interpretowana np. jako jakość geometryczna elementu, wpływająca na kolorowanie oraz filtrowanie; wartości z poza zadanego przedziału informują o błędnej wartości parametru
  + etykieta przypisana do elementu
* Udostępniając programistyczny interfejs umożliwiający tworzonym programom przygotowanie zbioru elementów do wizualizacji, przesłanie go do aplikacji wizualizującej (wraz z opisem tekstowym) i wstrzymanie do odpowiedniej reakcji użytkownika.
* Eksport oraz import przesłanych elementów do pliku w formacie .obj
* Zapewnia prezentację statystyk o ilości przesłanych oraz widocznych elementów
* Interfejs programistyczny dostarczony w językach Java, C++ oraz Python
* Możliwość działania w trybie 2D oraz 3D
* Kolorowanie elementów siatki w oparciu o współczynnik jakości oraz przynależność elementu do grupy
* Możliwość filtrowania elementów siatki w oparciu o
  + typ elementu
  + jakość elementu
  + przynależność do grupy
  + współrzędne elementów
* Możliwości wizualizacji siatki bieżącej oraz pewnej liczby poprzednio pokazywanych

## Wymagania niefunkcjonalne

* Kompatybilność aplikacji na systemie operacyjnym Windows oraz Linux
* Wydajność działania narzędzia w przypadku przesłania dużej liczby elementów
* Łatwa oraz szybka instalacja narzędzia w kodzie użytkownika

# Wybrane aspekty realizacji

## Architektura aplikacji

Struktury.png

Aplikacja została podzielona na następujące moduły zgodnie z powyższym diagramem:

* **Moduł komunikacji CORE** – zapewnia komunikację pomiędzy poszczególnymi komponentami systemu. Do jego głównych zadań należy
  + odbiór wiadomości oraz przekształcenie ich na wewnętrzne obiekty reprezentujące elementy przechowywane w drzewie struktur,
  + przesyłanie wiadomości statystyk do okienka Smeshalist Manager
  + pośredniczenie w filtracji elementów
  + obsługa wiadomości związanych z interakcją pomiędzy systemem a użytkownikiem
* **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.
* **Moduł filtracji CORE -** odpowiada za przechowywanie oraz zarządzanie zestawami filtrów jak również za filtrację.
* **Moduł wizualizacji CORE –** odpowiada za stworzenie instancji okna, obsługę zdarzeń, głównej pętli programu i sterowanie położeniem kamery.
* **Moduł konfiguracji użytkownika –** jego zadaniem jest odczyt konfiguracji użytkownika dostarczonej w pliku *user.config.xml*. W przypadku braku któregokolwiek parametru konfiguracji ustawiana jest wartość domyślna.
* **Smeshalist Manager –** odpowiada za interakcję z użytkownikiem.

## Stos technologiczny

Aplikacja zaimplementowana zostało w języku C++ z wykorzystaniem następujących narzędzi i bibliotek:

* OpenGL
* GLU
* GLUT
* Protocol Buffers

Ponadto dostarczony został interfejs programistyczny w języku Java, Python oraz C++ kompatybilne z platformami Windows i Linux.

## Komunikacja wewnątrz systemu

# Organizacja pracy

## Metodyka

Metodyka zastosowana przez nas w projekcie to model przyrostowy. Prace zostały podzielone na trzy przyrosty, w których każdy kolejny dostarczał nowej funkcjonalności narzędzia.

### Przyrost I

Pierwszy etap obejmował zebranie precyzyjnych wymagań odnośnie funkcjonalności narzędzia. Zaprojektowana została architektura systemu oraz dokonany został podział na moduły. Następnie zaimplementowany został prototypu narzędzia działającego pod systemem Linux, który dostarczał podstawowej funkcjonalności tj. wizualizacji podstawowych typów struktur, oraz ich filtracji. Zaimplementowany został interfejs programistyczny w języku Java w zakresie zdefiniowana wewnętrznego formatu struktur oraz komunikacji z modułem komunikacji CORE oraz okienko Smeshalist Manager w zakresie wyświetlania statystyk

### Przyrost II

Celem drugiego przyrostu było przeniesienie podstawowej funkcjonalności na platformę Windows oraz dopracowanie aspektów wizualnych aplikacji oraz dalszy rozwój funkcjonalności Java API i okienka Smeshalist Manager. Podczas tego etapu ukończona została implementacja interfejsu programistycznego w języku Java. Zostały wykonane prace nad poprawnym działaniem filtracji oraz zaimplementowane została funkcjonalność okienka Smeshalist Manager. Z powodu problemów podczas próby automatycznego budowania narzędzia na platformie Windows funkcjonalność ta nie została przeniesiona na ten system operacyjny.

### Przyrost III

Podczas tego etatu funkcjonalność interfejsu użytkownika została przeniesiona na język C++ oraz Python. Dodana została możliwość importu oraz eksportu drzewa struktur do pliku .obj. Zapewniona została możliwość konfiguracji podstawowych parametrów narzędzia np. kolorów dla poszczególnych grup. Funkcjonalność narzędzia została przeniesiona na system Windows. Ponadto zaimplementowane zostały dodatkowe wymagania określone przez klienta po zapoznaniu się z pierwszym prototypem takie jak wyświetlanie etykiet struktur, możliwość przejścia w tryb 2D czy też ustalenia przezroczystości struktur.

# Wyniki projektu

## Wyniki, ocena użyteczności

## Propozycje rozwoju systemu