Politechnika Wrocławska

Wydział Elektroniki Mikrosystemów i Fotoniki

# Kierunek: EiT

## PRACA DYPLOMOWA

## INŻYNIERSKA

Promotor:

#### Oprogramowanie stanowiska do pomiarów krzywych fotometrycznych oraz całkowitego strumienia świetlnego

#### The software for photometric curves and total luminous flux measurements

AUTOR: Tomasz Wilk

Tomasz Wilk

# Dr inż. Damian Pucicki

Ocena pracy i podpis promotora:

WROCŁAW 2017

Spis treści

[1. Wstęp 3](#_Toc471941414)

[2. Wprowadzenie 4](#_Toc471941415)

[2.1. Wprowadzenie teoretyczne 4](#_Toc471941416)

[2.1.1. Bryła fotometryczna 4](#_Toc471941417)

[2.1.2. Krzywa fotometryczna 5](#_Toc471941418)

[2.1.3. Strumień świetlny 6](#_Toc471941419)

[2.2. Opis dostępnych rozwiązań aplikacji do edycji i prezentacji danych fotometrycznych 7](#_Toc471941420)

[2.2.1. QLumEdit2 8](#_Toc471941421)

[2.2.2. IESviewer 9](#_Toc471941422)

[2.2.3. Photometrics Pro - Luminaire Analysis Software 10](#_Toc471941423)

[2.2.4. Podsumowanie 11](#_Toc471941424)

[3. Opis Stanowiska i projekt aplikacji 12](#_Toc471941425)

[3.1. Opis stanowiska pomiarowego 12](#_Toc471941426)

[3.2. Założenia projektowe 14](#_Toc471941427)

[3.3. Uzasadnienie wyboru narzędzi oraz technik programistycznych 15](#_Toc471941428)

[3.4. Opis narzędzi użytych podczas pisania aplikacji. 16](#_Toc471941429)

[3.5. Architektura aplikacji 17](#_Toc471941430)

[3.6. Sposób liczenia całkowitego strumienia świetlnego 18](#_Toc471941431)

[4. Opis wykonanego programu 22](#_Toc471941432)

[4.1. Opis funkcjonalności 22](#_Toc471941433)

[4.1.1. Okno Powitalne 22](#_Toc471941434)

[4.1.2. Moduł „Ustawienia” 22](#_Toc471941435)

[4.1.3. Główne okno 23](#_Toc471941436)

[4.1.4. Moduł „Krzywa fotometryczna” 24](#_Toc471941437)

[4.1.5. Moduł „Przekrój” 25](#_Toc471941438)

[4.1.6. Moduł „Strumień Świetlny” 26](#_Toc471941439)

[4.2. Weryfikacja poprawności aplikacji narzędziami inżynierii oprogramowania 27](#_Toc471941440)

[4. Wnioski 29](#_Toc471941441)

[4.1. Zalecenia do dalszego rozwoju aplikacji 29](#_Toc471941442)

[4.2. Podsumowanie 29](#_Toc471941443)

[Literatura 30](#_Toc471941444)

[Załączniki 31](#_Toc471941445)

# Wstęp

Na Wydziale Elektroniki Mikrosystemów i Fotoniki na Politechnice Wrocławskiej w ramach pracy inżynierskiej skonstruowano stanowisko do pomiarów krzywych fotometrycznych  
 i całkowitego strumienia świetlnego oraz arkusz kalkulacyjny do przetwarzania wyników pomiarów uzyskiwanych na w/w stanowisku [1].

Przeznaczeniem stanowiska utworzonego w Wydziałowym Zakładzie Mikroelektroniki   
i Nanotechnologii jest prowadzenie zajęć dydaktycznych mających na celu zapoznanie studentów z pojęciami z zakresu fotometrii. Studenci podczas ćwiczeń z aparaturą zapoznają się z podstawowymi pojęciami techniki świetlnej oraz nowoczesnymi metodami pomiarów fotometrycznych, prowadzonych przy pomocy obiektywnego detektora jakim jest spektrometr. Budowa stanowiska umożliwia przede wszystkim pomiar bryły fotometrycznej, a więc rozkładu oświetlenia w przestrzeni, generowanego przez źródła światła montowane bezpośrednio przed stanowiskiem pomiarowym. Pomiary tego typu wykonuje się między innymi celem weryfikacji oraz opisu własności opraw świetlnych. Dane takie w ustandaryzowanych formatach (IES, ELUMDATA) przez organizacje typu Illuminating Engineering Society [2] są ważnymi informacjami pozwalającymi konsumentom dobrać źródła oświetlenia do oczekiwanych efektów. Szczególnie ważne są informacje na temat natężenia oświetlenia, równomierności, stopnia oddawania barw.

Oprogramowanie wizualizujące rozkład oświetlenia na podstawie danych pomiarowych jest powszechnie dostępne, jednak wykonanie programu przeznaczonego do skonstruowanego stanowiska uwzględni procedurę pomiaru, stosowane jednostki oraz umożliwi w pełni wykonanie celu dydaktycznego jakim jest zrozumienie podstaw techniki świetlnej   
oraz orientacja w cyfrowych metodach analizy źródeł światła.

# 2. Wprowadzenie

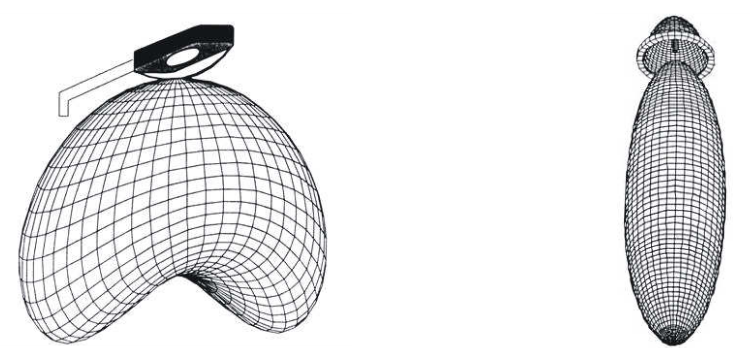
## 2.1. Wprowadzenie teoretyczne

Poniżej krótko zostaną omówione pojęcia wykorzystywane w pracy. Pozwoli to na lepsze zrozumienie pracy na stanowisku a także przyjętych założeń projektowych.

## 2.1.1. Bryła fotometryczna

Najpełniejszym sposobem ukazania właściwości fotometrycznych źródła światła   
jest prezentacja w przestrzeni trójwymiarowej jej bryły fotometrycznej. Jest to zamknięta powierzchnia, utworzona przez zakończenia wektorów wielkości fotometrycznej (strumienia świetlnego, światłości, luminancji, natężenia oświetlenia) o wspólnym początku w środku świetlnym oprawy, których wartość i kierunek odpowiada wartości światłości w tym kierunku [9].

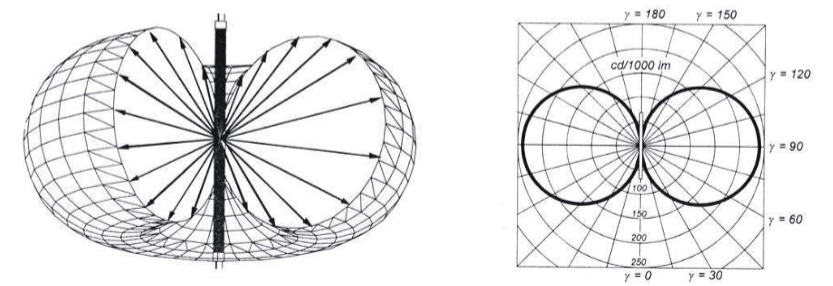
Na przykład dla punktowego źródła światła bryła fotometryczna jest ograniczona sferą o środku w punkcie źródła światła. Oznacza to także, że światłość jest uznawana za taką samą we wszystkich kierunkach.



Rysunek 1. Przykłady bryły fotometrycznej a) bryła fotometryczna oprawy oświetlenia drogowego (bryła nieobrotowo symetryczna); b) bryła fotometryczna reflektora zwierciadlanego (bryła obrotowo-symetryczna) [10]

## 2.1.2. Krzywa fotometryczna

Jednym ze sposobów prezentacji bryły fotometrycznej jest wykreślanie krzywych fotometrycznych (światłości). Taki sposób prezentacji umożliwia pokazanie wycinka bryły – przekroju wykonanego w jednej płaszczyźnie. Jeśli badana bryła fotometryczna charakteryzuje się symetrią obrotową, jest to prosty i jasny sposób przedstawienia bryły. Wykresy światłości typowo przedstawia się w biegunowym układzie współrzędnych, jednakże wykresy w układzie kartezjańskim są również spotykane, jednak zazwyczaj bywają mniej czytelne.



Rysunek 2. Wykres przedstawiający krzywą fotometryczną linii świetlnej w płaszczyźnie prostopadłej do osi świetlnej [10]

## 2.1.3. Strumień świetlny

Strumień świetlny jest jedną z podstawowych jednostek fotometrycznych charakteryzujących źródło światła. Strumień świetlny **Φ** jest wywiedziony od strumienia energetycznego   
na podstawie oceny promieniowania za pomocą odbiornika, którego względna czułość widmowa odpowiada czułości widmowej oka przystosowanego do jasności.

(1)

Km = 683 lm/W – fotometryczny równoważnik promieniowania,

Φe,λ – rozkład widmowy mocy promienistej (strumienia energetycznego),

Vλ – względna skuteczność świetlna promieniowania monochromatycznego.

Powyższe wyrażenie różni się od definicji całkowitej mocy promienistej uwzględnieniem skuteczności promieniowania dla zdefiniowanych długości fali Vλ (w przypadku fal widzialnych przez ludzkie oko będzie to 380 - 780 nm). Ze względu na brak jednolitych jednostek energetycznych i świetlnych w równaniu znalazł się również fotometryczny równoważnika promieniowania Km [9].

Wybór metody liczenia całkowitego strumienia świetlnego uzależniony jest od sposobu reprezentacji światłości. Aby policzyć całkowity strumień świetlny zgodnie ze wzorem (1) potrzebna jest znajomość algebraicznej reprezentacji strumienia świetlnego w przestrzeni.   
Jest to możliwe wyłącznie w teoretycznych przypadkach takich jak liczenie całkowitego strumienia świetlnego, którego źródłem jest punkt bądź linia świetlna.

W praktyce dostępny jest zazwyczaj zmierzony obliczony rozsył światłości, który mógłby być opisany analitycznie jedynie w sposób przybliżony. Najczęściej bazą obliczeń jest znajomość rozkładu natężenia światła oprawy oświetleniowej oraz ich relacje geometryczne (kątowe położenie i odległości) z oświetlanym obiektem. Istnieje kilka metody obliczania całkowitego strumienie świetlnego, m.in. metoda, metoda Wohaluera, metoda Rousseau. Każda z nich przyjmuje pewne uproszczenia, więc nie sprawdzają się one w przypadku rozkładów światłości niesymetrycznych obrotowo.

Aby dokładnie określić całkowity strumień świetlny w rzeczywistych przypadkach, należy uwzględnić sposób pomiaru. Powierzchnię pomiarową dzieli się na drobne obszary, odpowiadające punktom pomiarowym. Tak zdyskretyzowany strumień świetlny przez fragmenty powierzchni można poddać operacji sumy odpowiadający całkowaniu   
w przypadku analitycznym.

Mając całą powierzchnię S otaczającą źródło światła,

Strumień świetlny ΦS na powierzchni S otaczającej źródło światła można policzyć dzieląc obszar S na podobszary Δs, mierząc strumień światła przechodzący przez pojedynczy obszar, a następnie dokonując operacji sumowania strumieni cząstkowych

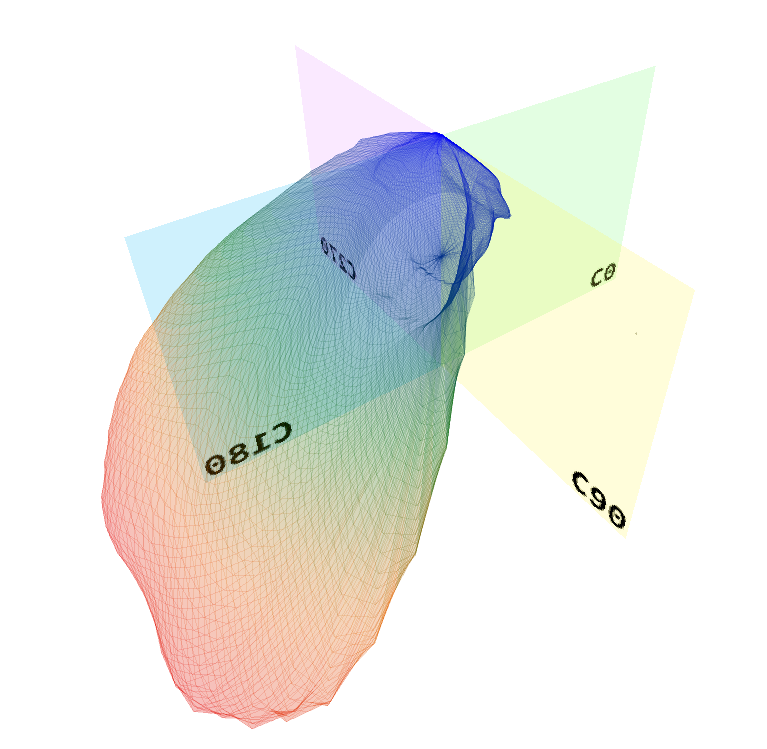
(2)

## 2.2. Opis dostępnych rozwiązań aplikacji do edycji i prezentacji danych fotometrycznych

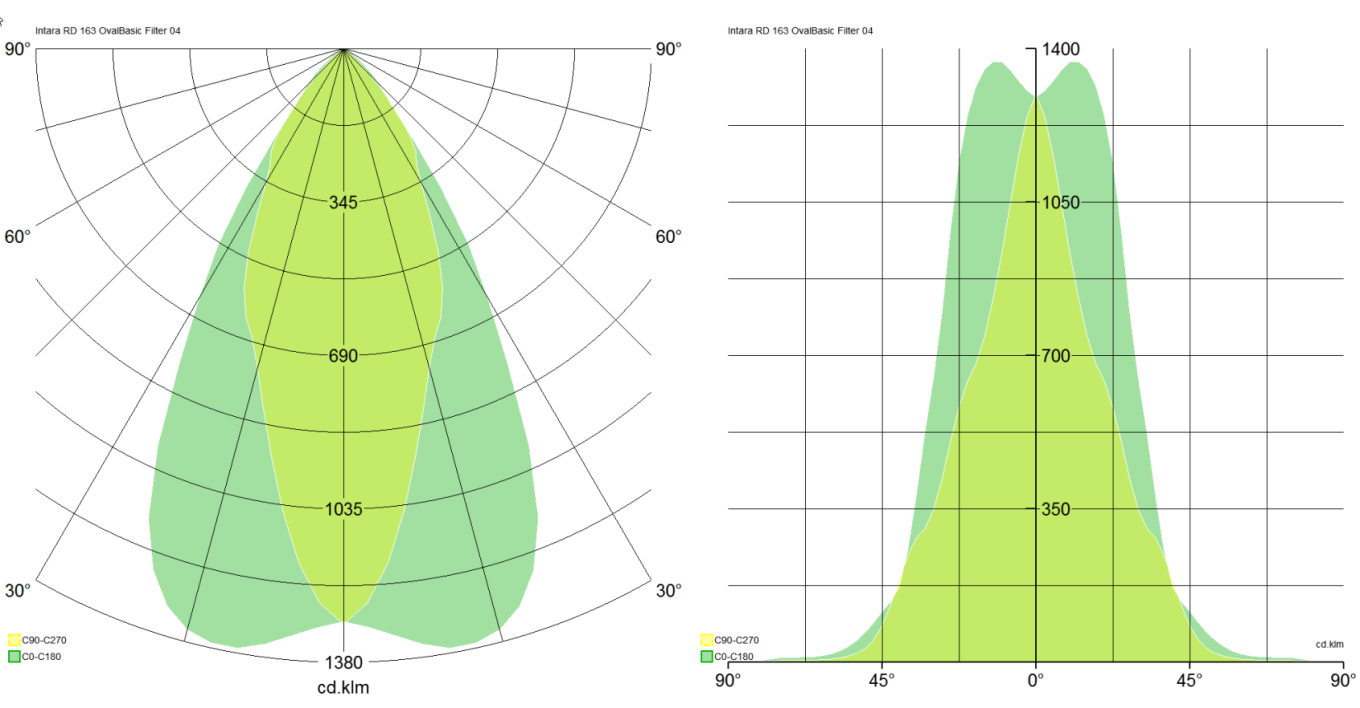
Ogólnodostępne w Internecie aplikacje pozwalają na tworzenie, modyfikacje oraz wyrysowanie wykresów rozkładu oświetlenia z plików w formacie IES,LDT (EULUMDAT)  
i TM-14 [3]. Poniżej przedstawione zostaną najpopularniejsze rozwiązania.

### 2.2.1. QLumEdit2

Program umożliwia wyświetlanie graficznej reprezentacji plików formatu LDT w przestrzeni trójwymiarowej [Rysunek 1], wykreślenie wartości luminancji na krzywej stożkowej oraz  
w układzie kartezjańskim [Rysunek 2], tworzenie nowych plików z predefiniowanych szablonów, porównanie dwóch źródeł, wydrukowanie wyników do plików. Aplikacja jest dystrybuowana w oparciu o licencję GPLv2.



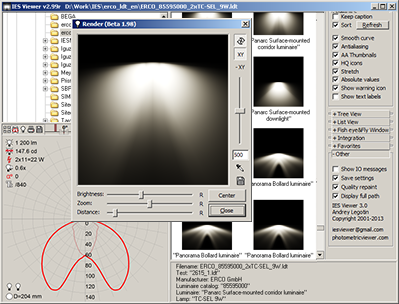
Rysunek 3. Kształt bryły fotometrycznej obrazowanej w programie QLumEdit2[4]



Rysunek 4. Krzywe fotometryczne w układzie biegunowym i liniowym w programie QLumEdit2 [4].

### 2.2.2. IESviewer

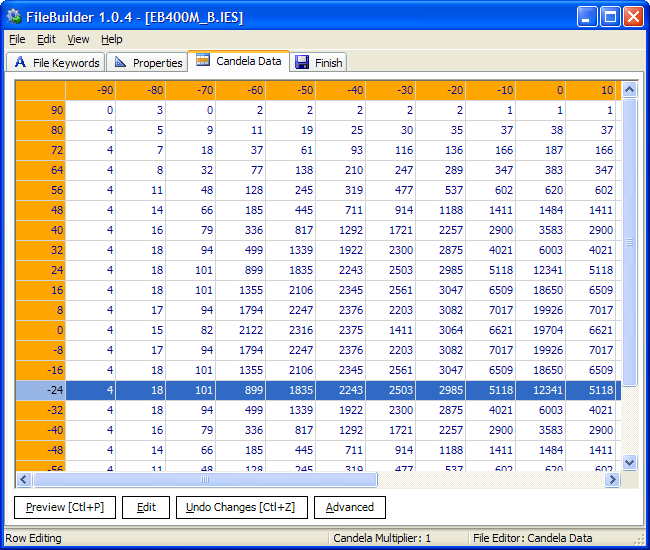
Strona domowa projektu deklaruje, że oprogramowanie jest jednym z najpopularniejszych sposobów prezentowania krzywych fotometrycznych. Dodatkowo umożliwia szybkie przeglądanie plików LDT i IES z uwzględnieniem producenta, numeru katalogowego, opisu źródła, typu fotometrycznego, intensywności, mocy wejściowej, renderowanie w czasie rzeczywistym, porównanie dwóch krzywych, konwersje plików między formatami, integrację z 3ds Max. Tworzy również bardzo dobre wizualizacje natężenia oświetlenia[Rysunek 3]. Jest to bezpłatna aplikacja.



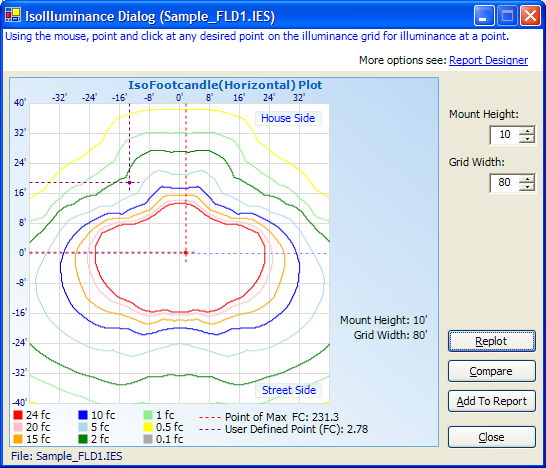
Rysunek 5. Zrzut ekranu głównego okna programu IESVIER obrazujący sposób przechodzenia światła przez konkretną oprawę oraz jej krzywą fotometryczną [5]

### 2.2.3. Photometrics Pro - Luminaire Analysis Software

Oprogramowanie w nowoczesnej formie umożliwia tworzenie plików opisujących bryłę fotometryczną źródła światła w przejrzystej tabeli [Rysunek 4], porównywanie krzywych fotometrycznych, generowanie raportów o rozkładzie świetlnym [Rysunek 5], przeszukiwanie, przeglądanie i organizowanie lokalnych danych fotometrycznych. Pełny, konfigurowalny opis danych (rozmieszczenie, kolory, wygląd), które następnie można porównać dla różnych krzywych. Umożliwia pracę na plikach w formatach IESNA, EULUMDAT. Omawiane oprogramowanie jest oprogramowaniem płatnym i umożliwia szerokie zastosowanie komercyjne [6].



Rysunek 6. Zrzut ekranu okna służącego do edycji pliku opisującego bryłę fotometryczną oprawy świetlnej w programiePhotometrics Pro [6]



Rysunek 7. Wykresy izokandeli w programie Photometrics Pro [6]

### 2.2.4. Podsumowanie

Inne programy z podobną funkcjonalnością to Eulumdat Tools[7] i LDTeditor [8]. Jakkolwiek istnieje wiele komercyjnych i otwartych rozwiązań, żadne nie spełnia wymagań dydaktycznych stanowiska. Skupiają się one głównie na możliwości porównań charakterystyk fotometrycznych różnych opraw, ale już nie prostego tworzenia nowych wykresów.  
Tylko jeden program QLumEdit2 umożliwia prezentację bryły fotometrycznej w przestrzeni trójwymiarowej. Stąd potrzeba zaprojektowania i wykonania oprogramowania uwzględniającego specyfikę stanowiska pomiarowego oraz cel dydaktyczny.

# 3. Opis Stanowiska i projekt aplikacji

Typowo, pomiary strumienia świetlnego wykonuje się za pomocą kuli Ulbrichta. Jednak   
ta technika nie pozwala na zobrazowanie bryły fotometrycznej. Pełniejsze pomiary zazwyczaj wykonuje się na ławie optycznej lub specjalistycznym goniometrem. Ten sposób umożliwia określenie światłości w wielu miejscach powierzchni otaczającej źródło światła przez obrót przetwornika fotoelektrycznego zamocowanego na ramieniu o stałej długości dookoła nieruchomo zamontowanego źródła światła. Położenie źródła światła zmienia się jedynie przez obrót dookoła osi pionowej o kąt β, natomiast przez obrót ramienia dookoła osi poziomej zmienia się położenie przetwornika fotoelektrycznego o kąt α [11].

# 3.1. Opis stanowiska pomiarowego

Na Wydziale Elektroniki Mikrosystemów w roku akademickim 2015/2016 Pani Ewa Szwaba w ramach swojej pracy dyplomowej [1] zaprojektowała i wykonała stanowisko pomiarowe odbiegające od klasycznych rozwiązań.

Zwyczajowo, pomiary bryły fotometrycznej prowadzone są specjalistycznym goniometrem, jednak p. Szwaba skonstruowała prostszy, bardziej niezawodny mechanizm odpowiadający potrzebom dydaktycznym.



Rysunek . Wykonane stanowisko do pomiaru krzywych fotometrycznych pomiarów wraz z oprawą świetlną umieszczoną w centralnej części głowicy pomiarowej [1]

Stanowisko składa się z:

* podstawy montowanej do stołu optycznego (czy ta nazwa jest prawidłowa? Ława?),
* półokrągłej szyny pozwalającej na przesuwanie detektora po „południkach” półsfery pomiarowej w zakresie kątów 5o – 175o stopni z krokiem 5o,
* obrotowej głowicy pomiarowej, która wraz z szyną pozwala na poruszanie się po „równoleżnikach” półsfery,
* spektrofotometru StellarrNet BLUE-Wave z detektorem kosinusoidalnym zamontowanym na półokrągłej szynie,
* komputer umożliwiający odczyt danych ze spektrofotometru oraz obróbkę danych  
  za pomocą programu, który jest przedmiotem tej pracy.

Oprawa świetlna zamontowana jest w ten sposób, aby jej domniemany środek świetlny znajdował się w środku symetrii półsfery pomiarowej, tak więc każdy punkt sfery znajduje się w równej odległości od odprawy. Pozwala to na pomiar natężenia światła detektorem sinusoidalnym z krokiem nie mniejszym niż 5 stopni w każdym kierunku (na „południkach” i „równoleżnikach” półsfery).

Spektrometr, do którego podłączony jest detektor obsługiwany jest przez komputer PC będący częścią stanowiska pomiarowego. Student odczytuje natężenie fali w zadanym zakresie częstotliwości (z reguły 380-780 nm dla światła widzialnego) z oprogramowania SPECTRAWIZ@. Docelowo, uczestnik zajęć dydaktycznych będzie wprowadzał wynik do tabeli w programie „Krzywe Fotometryczne”, którego opisem jest niniejsza praca. Student będzie miał możliwość obserwacji bryły fotometrycznej, krzywych fotometrycznych, przekrojów przez bryłę (przekrój równoległy do płaszczyzny źródła światła) oraz zapoznania się ze strumieniem świetlnym wyliczonym przez aplikację.

# 3.2. Założenia projektowe

Oprogramowanie ma spełniać następujące założenia:

* będzie działać na platformach Windows (używanej w laboratorium) i Linux,
* zostanie wykonane przy wykorzystaniu przenośnych bibliotek i narzędzi programistycznych QT,
* zapewni możliwość stworzenia nowego projektu lub otworzenia już istniejącego  
  z pliku,
* umożliwi eksport danych pomiarowych do pliku,
* umożliwi wybór gęstości punktów pomiarowych na sferze (z predefiniowanej listy – 5o, 10o, 15o, 45o),
* umożliwi wybór jednostek pomiaru natężenia światła (W/m2, µW/cm2, lm/m2),
* projekt oraz wykresy oznaczone będą nazwą mierzonej oprawy lub źródła światła,
* umożliwi wprowadzenie wartości natężenia światła z detektora umieszczanego w różnych punktach półsfery pomiarowej,
* umożliwi modyfikację danych pomiarowych,
* będzie potrafiło narysować krzywą fotometryczną ze zdefiniowanej płaszczyzny,
* będzie potrafiło narysować bryłę fotometryczną,
* umożliwi policzenie całkowitego strumienia świetlnego padający na półsferę pomiarową,
* umożliwi eksport narysowanych krzywych fotometrycznych i bryły do plików graficznych,
* umożliwi wybór języka interfejsu użytkownika (polski/angielski),
* zostanie wykonane zgodnie z nowoczesnymi zasadami inżynierii oprogramowania,
* zostanie sprawdzone statycznymi i dynamicznymi narzędziami do analizy kodu (Clang, Valgrind).

## 3.3. Uzasadnienie wyboru narzędzi oraz technik programistycznych

Jako język do napisania programu wybrano C++. Jako powszechnie znany język kompilowalny, ze statyczną kontrolą typów pozwala on na swobodny rozwój aplikacji oraz utrzymywanie jej przez studentów wydziału, którzy zapoznają się z nim na zajęciach. Wraz z wersją C++11 elementem języka stała się również biblioteka standardowa (stdc++), która zawiera dodatkowe struktury danych w stosunku do standardu C++98 (wektory, listy, pary) oraz moduł z prostymi algorytmami (sumą, poszukiwaniem dowolnej wartości w kontenerze, wyszukiwaniem skrajnej wartości w kontenerze, przekształceniem każdej wartości w kontenerze wg określonego wzorca itd.) umożliwiającymi łatwe, intuicyjne manipulowaniem danymi, często redukując skomplikowane wyrażenia do jednej linii.

Założenia projektu wymuszają wykonanie Graficznego Interfejsu Użytkownika (GUI) oraz prezentacji wykresów zarówno dwu- jaki i trójwymiarowych. Rozpatrywano wybór różnych środowisk i bibliotek programistycznych pozwalających uniknąć bezpośredniego użycia OpenGLa:

* **Microsoft Foundation Classes (MFC)** – działa tylko na systemach operacyjnych firmy Microsoft, do rysowania wykresów należałoby użyć osobnych bibliotek takich jak QCustomPlot, QPlot,
* **WxWidgets** – biblioteka międzyplatformowa, posiada możliwość rysowania wykresów dwuwymiarowych, jednakże nie jest ona na bieżąco aktualizowana. Wykresy 3D nie są dostępne, ich użycie wymagałoby użycia zewnętrznych bibliotek.
* **QT** – biblioteka międzyplatformowa. Od wersji 5.7 (najnowszej w chwili pisania aplikacji – stan na przełom 2016/2017 r.) dodano możliwość użycia modułów Data Visualisation i Charts na licencji LGPL, które idealnie pasują do wstępnych wymagań programu, tj. umożliwiają rysowanie wykresów trójwymiarowych i krzywych w układzie współrzędnych biegunowych.

QT było więc najlepszym, naturalnym wyborem.

## 3.4. Opis narzędzi użytych podczas pisania aplikacji.

Aplikacja została napisana w jęzku **C++,** z zastosowaniem bibliotek Qt oraz zintegrowanego środowiska programistycznego **Qt Creator**. Służą one do pisania aplikacji (przede wszystkim z graficznym interfejsem użytkownika), które mogą być używane w różnych systemach operacyjnych, na wielu platformach sprzętowych z takim samym kodem lub po niewielkich zmianach. Tworzenie kodu w QT 5.7 odbywa się z użyciem języka C++11 z rozszerzeniami włączając SYGNAŁY/SLOTY upraszczające obsługę zdarzeń takich jak interakcje  
z użytkownikiem czy konieczność obsługi interfejsu, zapewnia abstrakcję operacji wejścia-wyjścia, które normalnie różnią się na systemach operacyjnych. Qt z założenia jest w pełni obiektowe. Z perspektywy programisty, każdy przycisk, okno czy widget jest obiektem. Poprzez dziedziczenie po wspólnych klasach (QWindow, QWidget) każdy z obiektów ma podobny interfejs służący do jego obsługi. Znacznie ułatwia to i przyśpiesza rozwój programu.

Użyto następujących modułów QT:

* **Qt Core** – podstawowa funkcjonalność biblioteki Qt używana przez inne moduły, włączając system meta-objektów, kontenery typów danych, obsługę systemu zdarzeń i operacji wejścia-wyjścia.
* **Qt GUI** – Główny moduł GUI.
* **Qt Widgets** – zawiera klasy z podstawowymi widżetami (okna, przyciski, listy, pola tekstowe, wyboru itp.)
* **Qt Print Support** - moduł umożlwiający drukowanie. W projekcie używany  
  do drukowania grafiki do pliku w formacie \*.pdf.
* **QT Data Visualization** – moduł pozwalający na wizualizację danych w przestrzeni trójwymiarowej łącznie z obracaniem i powiększaniem obiektu graficznego używający OpenGLa do renderowania grafiki.
* **Qt Charts** - służy do rysowania wykresów na płaszczyźnie (także we współrzędnych biegunowych)

Aby w pełni wykorzystać możliwości QT, użyto zintegrowanego środowiska programistycznego (IDE) Qt Creator. Oprócz standardowej funkcjonalności (możliwość pisania i kompilacji kodu, zarządzanie wieloma projektami), zawiera moduły do szybkiego projektowania interfejsu graficznego, sprawdzanie poprawności kodu narzędziami Clang  
i Valgrind.

Podczas implementacji poprawność kodu weryfikowana była kompilatorem **g++** w wersji 6.3. Współczesne kompilatory nie tylko służą do przetwarzania tekstu na ciągi instrukcji zrozumiałe dla procesora, ale potrafią również wychwycić i zgłosić jako ostrzeżenie problemy w kodzie, które mogą się objawić dopiero podczas działania programu (takie jak użycie niezainicjalizowanej zmiennej, porównywanie zmiennych całkowitych o różnej reprezentacji itd). Dlatego podczas pisania, aplikacja była kompilowana z flagami -Wall, -Werror, które uniemożliwiały zbudowanie programu w przypadku wystąpienia jakichkolwiek ostrzeżeń.

Użyto również Statycznego Analizatora Kodu Clag, który jest zbiorem algorytmów i technik używanych do analizy kodu źródłowego w celu znalezienia błędów. Idea jest podobna  
do ostrzeżeń kompilatora, jednak Clang pozwala na pokazanie problemów, które znajdowane były do tej pory już podczas działania programu lub debuggowania. Clang przeszedł drogę od prostego sprawdzania składni do ewaluowania semantyki kodu [12].

Aby uniknąć błędów związanych z używaniem pamięci użyto oprogramowania **Valgrind**, który służy do debuggowania pamięci, wyszukiwania wycieków pamięci i profilowania programów. Valgrind jest typem maszyny wirtualnej, która używając techniki kompilacji just-in-time (JIT) zamienia binarny program na jego tymczasową reprezentację, która używa zmodyfikowanych wywołań systemowych do obsługi pamięci. Umożliwia to wykrywanie  
i śledzenie błędów związanych z użyciem niezainicjalizowanej pamięci, niezwalnianiem zasobów, lokalizowania odwołań poza przestrzeń adresową procesu.

Kontrola wersji prowadzona była narzędziem **GIT**. Pozwala on na śledzenie zmian w kodzie, wprowadzanie ich w spójny sposób włączając opisy oraz prostą modyfikację historii.

Pozwolił to na przechowywanie kodu nie tylko lokalnie ale też posiadanie zdalnej, zapasowej wersji, szybkie wycofywanie niepożądanych zmian jak również proste znajdowanie zmian, które uszkodziły wcześniejszą funkcjonalność.

Kod aplikacji powstał w oparciu o wytyczne programowania w C++ opublikowane przez Google (Google C++ Style Guide) [14] z nielicznymi wyjątkami wymuszanymi przez IDE. Trzymanie się jednego standardu kodowania pozwoli na utrzymywanie tego samego stylu podczas utrzymania aplikacji jaki został użyty przy jej pisaniu.

## 3.5. Architektura aplikacji

Do opisu architektury aplikacji użyto języka UML oraz oprogramowania StarUML. Hierarchia typów klas oraz Diagram klas znajdują się w załącznikach do pracy.

Główną osią każdej aplikacji pisanej w QT jest pętla zdarzeń, która obsługuje interakcje  
z interfejsami, w tym, z użytkownikiem. Aby swobodnie korzystać z pełnego zestawu funkconalności tego mechanizmu do GUI, należy swoje obiekty opierać na już istniejących klasach specyficznych dla QT. W hierarchii typów klas (Załącznik 1) widocznym jest,  
że klasy implementujące okna aplikacji (PolarGraphWindow, SurfaceWindow, SetupLuminousFluxWindow, IntroductionDialogWindow), dziedziczą po elementach modułu Widget QT. Jest to klasyczne wykorzystanie wzorca projektowego Dekorator sprowadzające się do rozszerzenia funkcjonalności znanego obiektu.

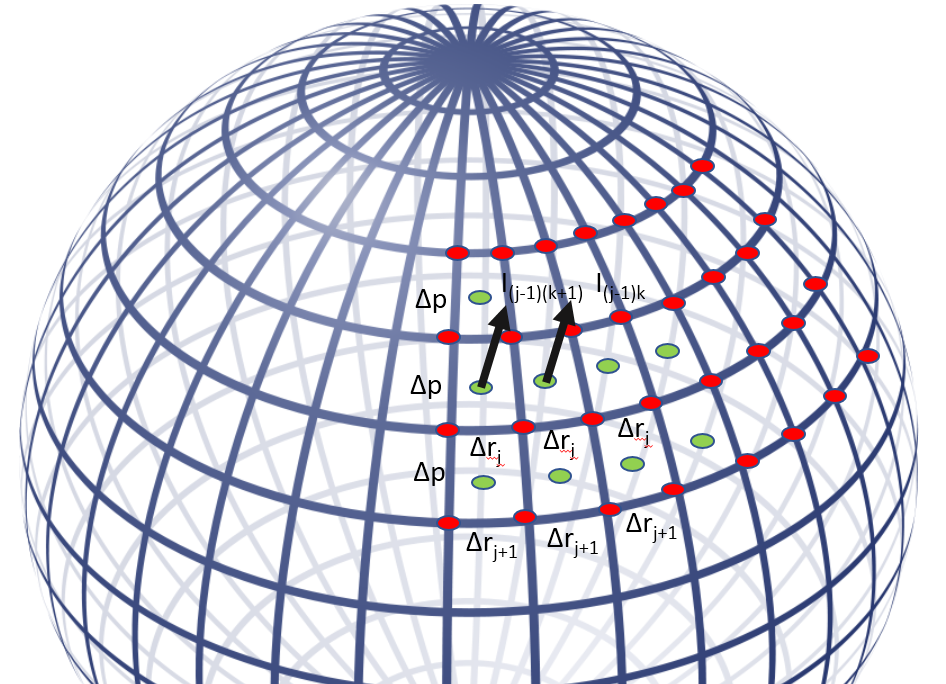
Z uproszczonego diagramu klas (Załącznik 2) jasno wynika, że głównym komponentem aplikacji jest klasa MainWindow, której delegaci potrafią zareagować na obsługę przycisków. Podczas pracy aplikacji zgodnie z potrzebami tworzone są instancje klas LuminousFluxWindow, PolarGraphWindow, IntroducionDialogWindow,  
czy SetupProjectDialogWindow. Klasa ProjectSettings przechowuje ustawienia aplikacji,  
ale odnieść się do niej można bezpośrednio z klasy MainWindow. Klasa Point reprezentuje punkt na półsferze łącznie ze zmierzoną wartością. Podczas konstrukcji obiektu wyliczane  
są wszystkie parametry charakteryzujące punkt pomiarowy poprzez odwzorowanie układu współrzędnych sferycznych do układu kartezjańskiego. Klasa Point agregowana jest przez wszystkie klasy wymagające dostępu do danych pomiarowych.

Każdej z klas prezentującej wykresy (PolarGraphWindow, SurfaceWindow) bądź obliczenia (LuminousFluxWindow) przydzielono klasę funkcyjną bez interfejsu użytkownika, ukrywającą implementację przed jej nadrządcą.

## 3.6. Sposób liczenia całkowitego strumienia świetlnego

Przytaczanie szczegółów implementacyjnych całej aplikacji nie jest istotne dla całości pracy, jednakże zdecydowano się na przedstawienie numerycznej metody obliczania strumienia świetlnego.

Mierzone źródło światła ma swój środek świetlny w środku geometrycznym wyobrażonej sfery przedstawionej na rysunku X. Stanowisko pomiarowe umożliwia określenie strumienia świetlnego w punktach znajdujących się na półsferze w równych odstępach kątowych, począwszy od jej „wierzchołka” (znajdującego się na osi świetlnej źródła w odległości promienia sfery od źródła światła. Stosując terminologię geograficzną, punkty pomiarowe znajdują się na przecięciach „południków” i równoleżników półsfery.



Rysunek Sfera pomiarowa z prostymi na przecięciu których znajdują się punkty pomiarowe [15]

Z równania (2), wiadomo, że strumień świetlny ΦS na powierzchni S otaczającej źródło światła można policzyć dzieląc obszar S na podobszary Δs, mierząc strumień światła przechodzący przez pojedynczy obszar, a następnie dokonując operacji sumowania strumieni cząstkowych przechodzących przez pojedyncze obszary:

(2)

Wiedząc, że punkty pomiarowe są rozłożone w równych, kątowych odstępach na powierzchni sfery, naturalnym staje się podział sfery na trapezy, których podstawą są fragmenty równoleżników, a wysokością fragment południka.

Mając półsferę pomiarową o promieniu R oraz zakładając, że południk został podzielony  
na J równych części, a równoleżnik na części K, oraz kąt pomiędzy punktem wspólnym południków, środkiem geometrycznym półsfery i równoleżnikiem to , możemy określić wysokość trapezów:

Oraz długość jego podstaw:

Korzystając ze wzoru na pole powierzchni trapezu otrzymujemy pole pojedynczego trapezu na powierzchni sfery:

Strumień świetlny przechodzący przez każdy z trapezów dla uproszczenia przyjęto jako średnią arytmetyczną z otaczających punktów pomiarowych:

Zatem podstawiając do równania (2), całkowity strumień świetlny liczony jest według formuły:

Z powodu braku technicznej możliwości pomiaru wartości IJ został on aproksymowany za pomocą regresji liniowej z ostatnich dwóch punktów.

Weryfikacja poprawności powyższego rozumowania oraz dokładności obliczeń odbyła się poprzez sprawdzenie wyników dla promienia sfery R = 1 oraz założenia, że każdy z otrzymanych wyników pomiaru strumienia wyniósł Przy takim założeniu jako wynik powinno otrzymać się jako wynik powierzchnię półsfery, czyli:

Tabela . Wartości błędów określenia powierzchni półsfery dla różnych ilości punktów pomiarowych

|  |  |  |  |
| --- | --- | --- | --- |
| Krok pomiaru | Ilość pól pomiarowych | Uzyskany  Wynik | Błąd pomiar w stosunku do teoretycznej wartości [%] |
| 15 | 6 \* 24 = 144 | 6.2472574 | 0,572 |
| 10 | 9 \* 36 = 324 | 6.2672275 | 0,254 |
| 5 | 18 \* 72 = 1296 | 6.2791974 | 0,063 |

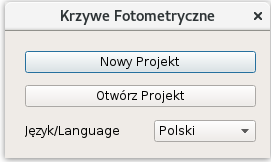
Z tabeli 1. widocznym jest, że metody numeryczne i przybliżanie powierzchni połowy sfery metoda sumowania trapezów na jej powierzchni ma wpływ na dokładność wyniku. Jednak przy wartościach błędu niższych niż 1% i wyłącznie dydaktycznemu przeznaczeniu stanowiska, nie będzie stanowiło to problemu.

# 4. Opis wykonanego programu

## 4.1. Opis funkcjonalności

### 4.1.1. Okno Powitalne

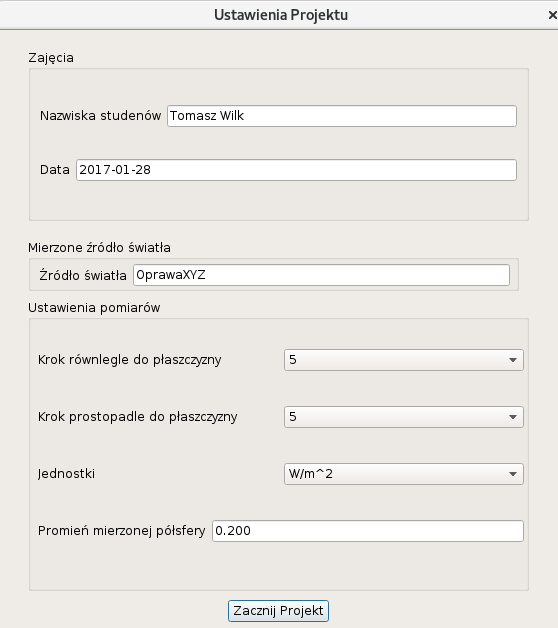
Po otwarciu aplikacji „Krzywe Fotometryczne” użytkownik ma możliwość stworzenia nowego projektu lub otworzenia już istniejącego (Rysunek 9). Język obsługi aplikacji (polski lub angielski) odczytywany jest automatycznie z ustawień systemu operacyjnego, jednak pozostawiono również opcję ręcznej zmiany.



Rysunek Okno wyboru ukazujące się po otwarciu aplikacji

### 4.1.2. Moduł „Ustawienia”

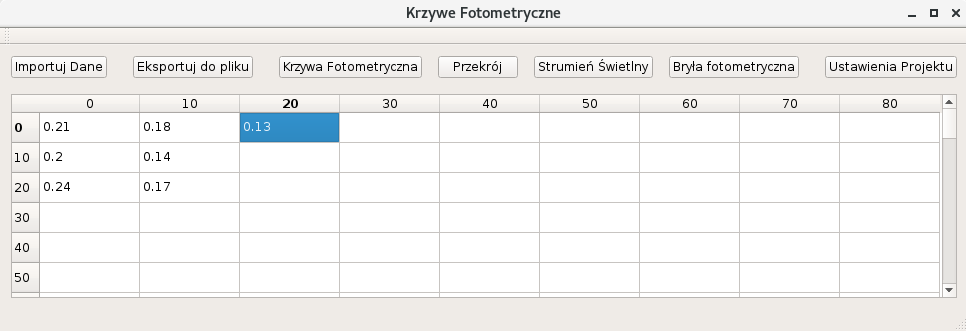
Jeśli zostanie wybrana pierwsza opcja, to istnieje możliwość ustawienia parametrów projektu (Rysunek 10). Nazwisko studenta, data oraz nazwa źródła światła będą pojawiać się później na wykresach. Kroki pomiarowe definiują gęstość pomiaru (co 5o, 10o, 15o). Jednostki pomiaru natężenia światła zostały także zdefiniowane na etapie wymagań: W/m2, µW/cm2, lm/m2.



Rysunek Okno konfiguracyjne programu

### 4.1.3. Główne okno

Po akceptacji ustawień przyciskiem „Ustaw”, otworzone zostaje główne okno programu (Rysunek 11) zawierające tabelę w której można wpisywać wyniki pomiarów. Kolumny oznaczone są „równoleżnikami” (współrzędnymi na „południku”, szerokością geograficzną) półsfery w stopniach rozpoczynając od ich punktu wspólnego („bieguna”). Wiersze analogicznie podpisano współrzędną południków. Student ma możliwość wpisywania wartości pomiarów natężenia promieniowania.



Rysunek Główne okno aplikacji "Krzywe fotometryczne"

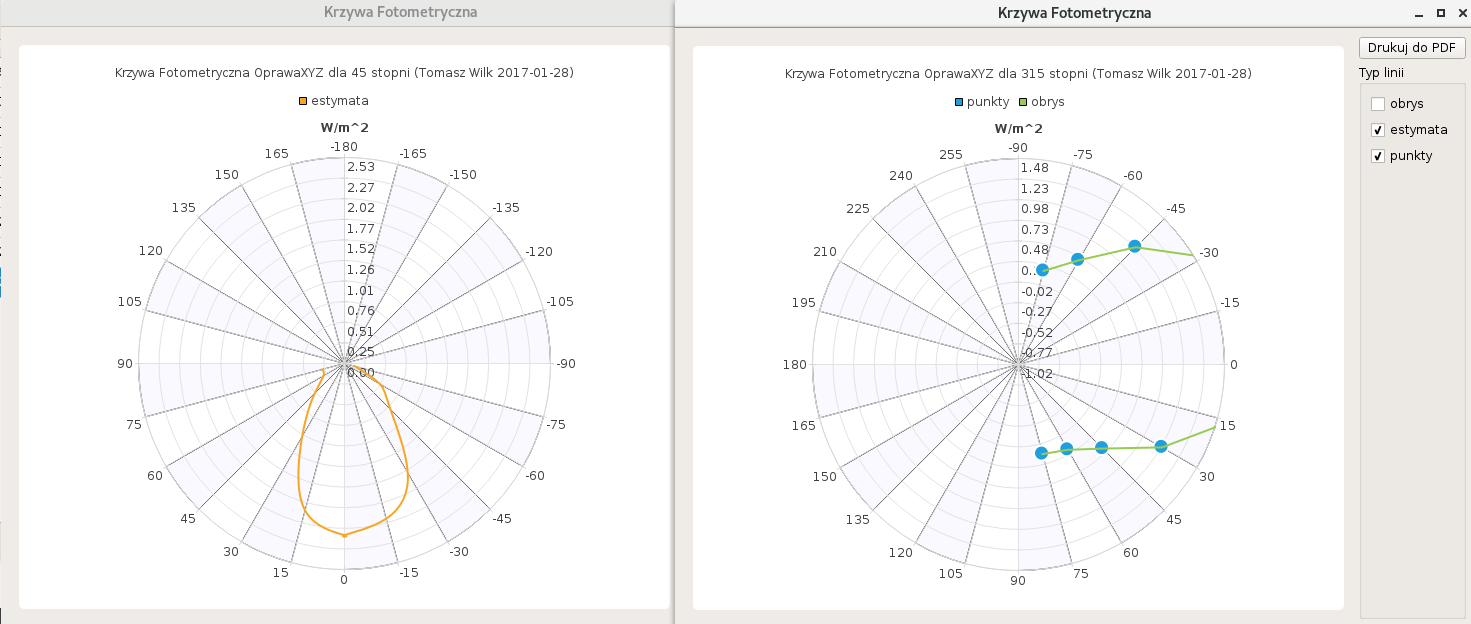
Odpowiednie przycisku w głównym oknie aplikacji umożliwiają:

* zapisanie danych pomiarowych do pliku CSV (comma -separated values)
* odczytanie danych pomiarowych z pliku CSV zapisanych przez aplikację
* zmianę ustawień projektu w oknie z *Rysunku 10.*
* wyrysowanie krzywej fotometrycznej po wybraniu wierszy z których ma być obrazowana krzywa
* wykreślenie przekroju przez bryłę fotometryczną po wybraniu wiersza dla którego ma być sporządzony graf,
* policzenie strumienia świetlnego,
* narysowanie bryły fotometrycznej w układzie współrzędnych XYZ.

Cztery ostatnie punkty zostały szczegółowo opisane poniżej.

### 4.1.4. Moduł „Krzywa fotometryczna”

Głównym zadaniem modułu jest narysowanie krzywej fotometrycznej dla zadanego „południka” sfery. W terminologii geograficznej jest to linia na powierzchni sfery łącząca bieguny, w niniejszym opisie zaczyna się w punkcie wspólnym wszystkich południków  
(na osi świetlnej źródła) i prowadzi do krawędzi półsfery pomiarowej prostopadle do niej.  
Po wybraniu wiersza w głównym oknie, wartości pomiarów są pobierane z jego linii.  
Aby uzyskać przekrój przez całość sfery, pobierane są też wartości z komplementarnego do niego wiersza po przeciwnej stronie sfery. To umożliwia narysowanie krzywej w zakresie <-85o, 85o>.



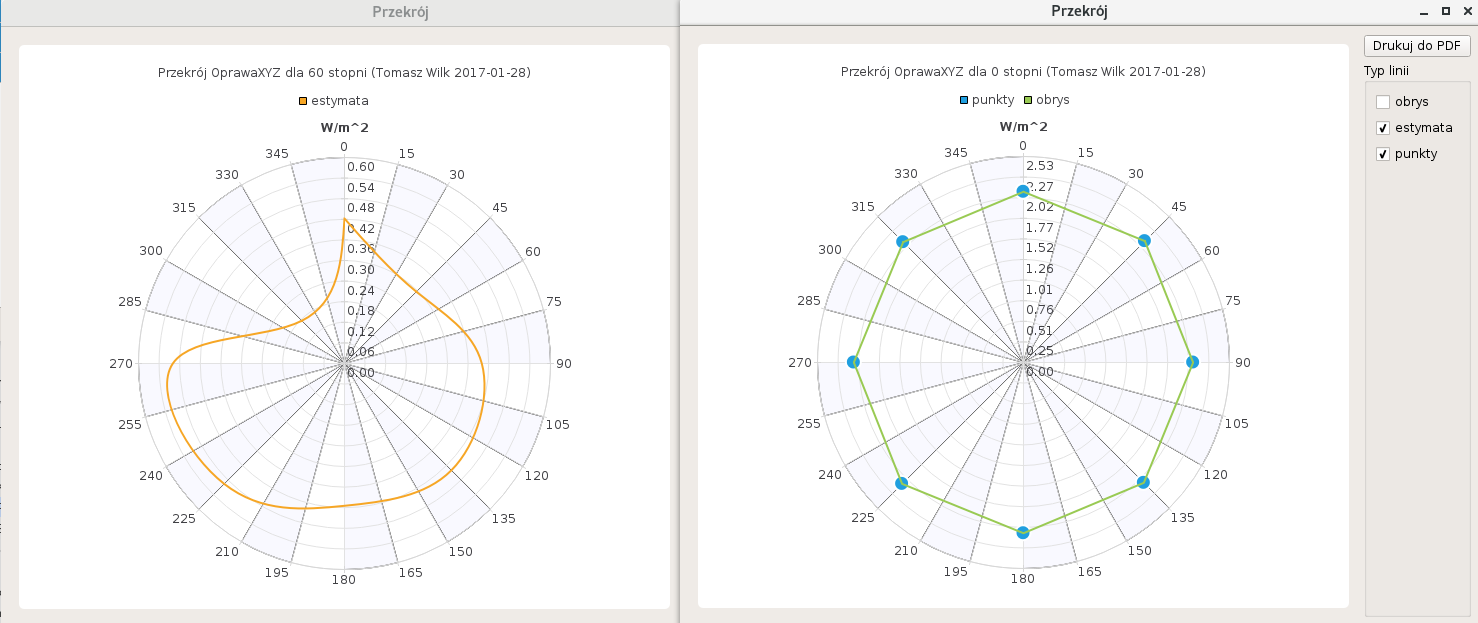
Rysunek Okna z krzywą fotometryczną w układzie współrzędnych biegunowych.

Student ma możliwość wyboru jak wykreślona krzywa ma wyglądać – czy powinna być estymowana (pomarańczowa linia), czy łączyć (zielona linia) wartości pomiarów (niebieskie punkty), co zobrazowano na Rysunku 12. Wykres może być powiększany i obracany za pomocą strzałek na klawiaturze, co zostało przedstawione w niższym oknie.

Wygenerowany obraz może być zapisany do pliku pdf.

### 4.1.5. Moduł „Przekrój”

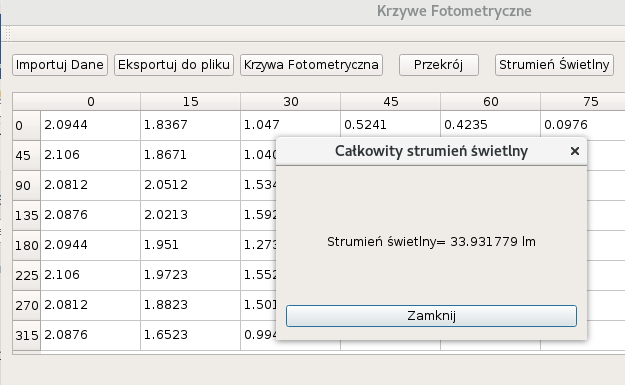
Po wybraniu kolumny z danymi przekroju przez konkretny „równoleżnik” półsfery, program rysuje go we współrzędnych biegunowych w oknie identycznym jak w module „Krzywe fotometryczne”. Opis funkcjonalności nie różni się od zawartego powyżej.



Rysunek . Okna z przekrojami przez bryłę fotometryczną

### 4.1.6. Moduł „Strumień Świetlny”

Moduł wylicza całkowity strumień świetlny zgodnie z metodą opisaną w podrozdziale 2.6 biorąc pod uwagę jednostkę natężenia światła ustawioną przez użytkownika we wcześniejszych krokach. Wartość wyświetla się w oknie z możliwością skopiowania jego zawartości (Rysunek 14).

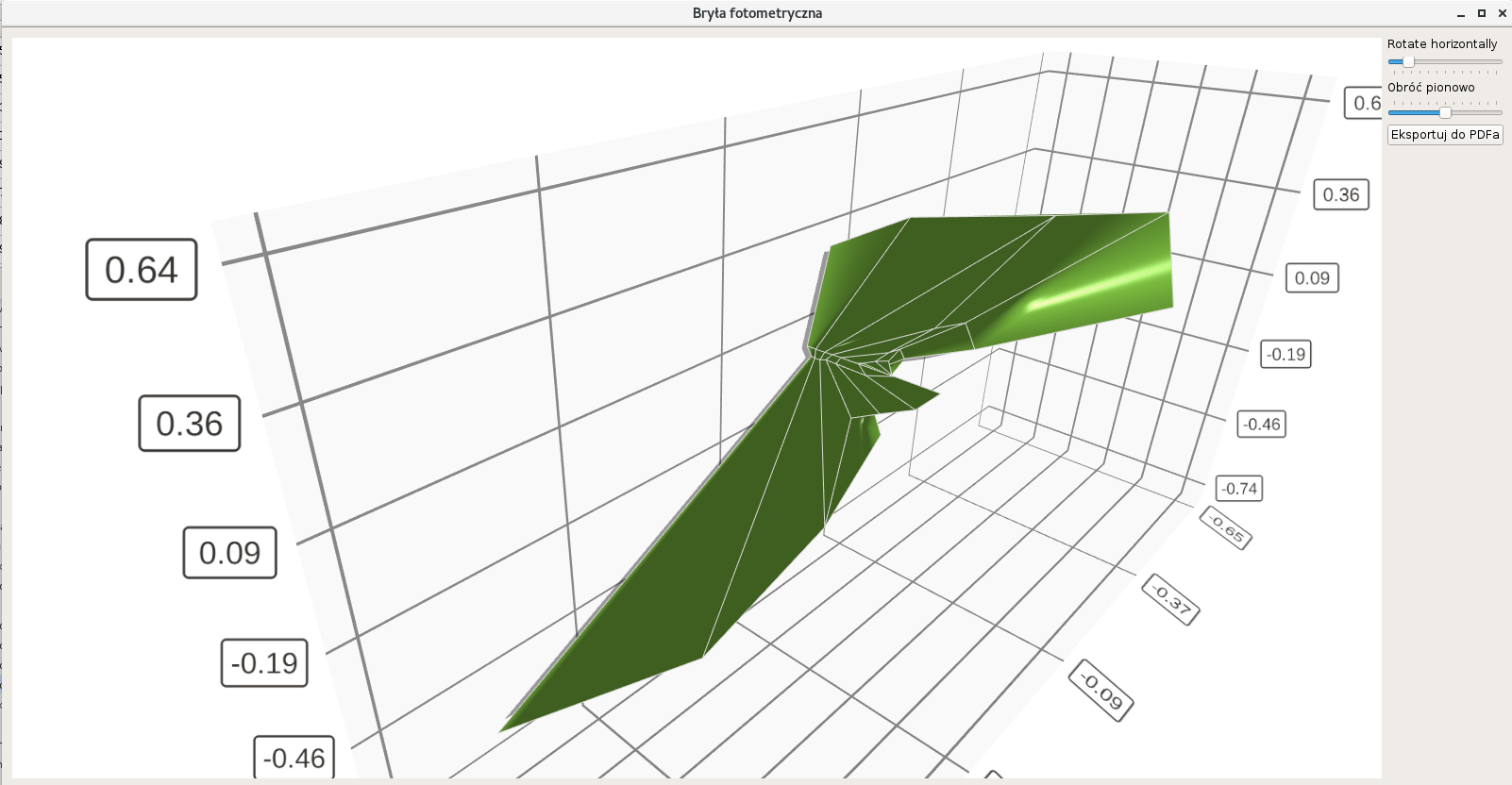
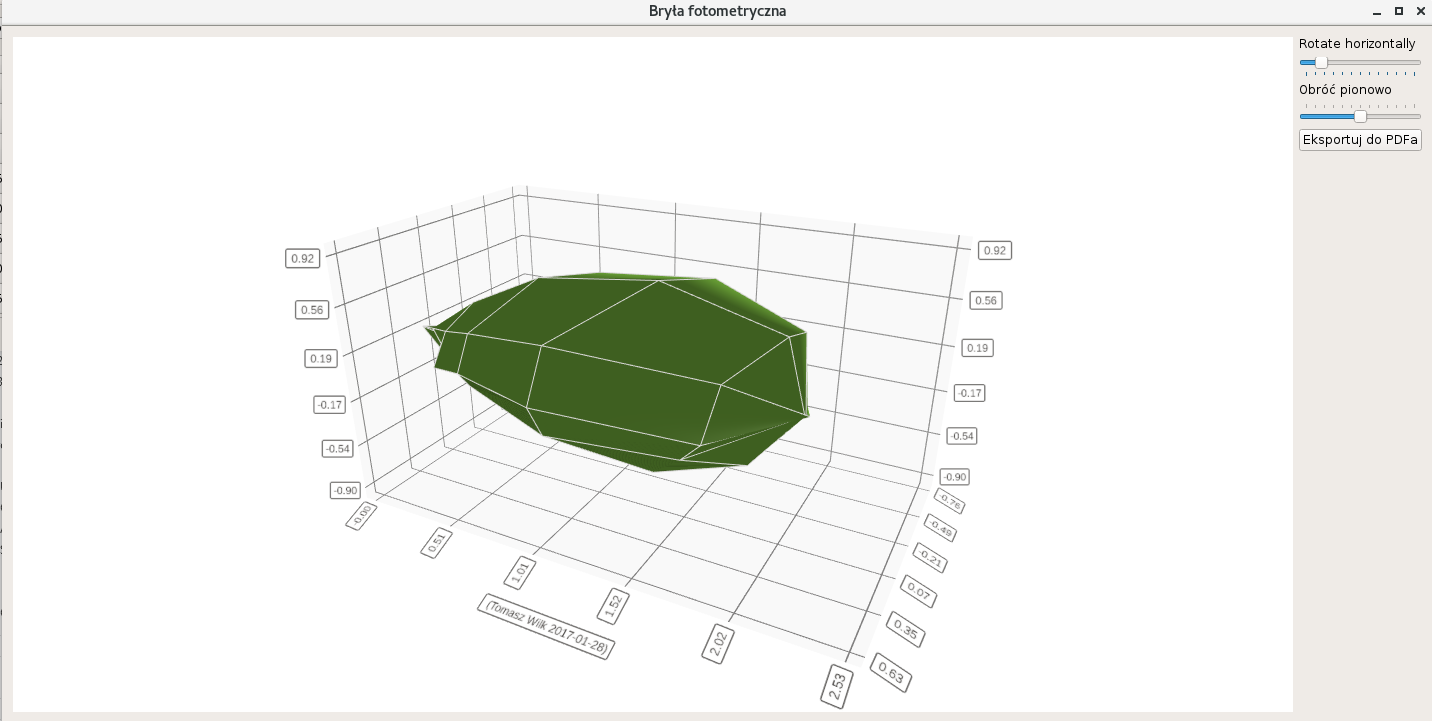


Rysunek Wartość całkowitego strumienia świetlnego wyliczona przez aplikację

Otrzymana wartość (33,93 lm) jest rozsądnego rzędu, a implementacja jej według wcześniej wspomnianej metody pozwala sądzić, że jest wartością poprawną.

**Moduł „Bryła fotometryczna”**

Moduł umożliwia wykreślenie bryły fotometrycznej z danych pomiarowych w trójwymiarowym, kartezjańskim układzie współrzędnych (Rysunek 15.) . Student za pomocą suwaków, myszy i przycisków klawiatury, ma możliwość obracania widoku bryły w osi zarówno równoległej jak i prostopadłej do płaszczyzny XY. Po odpowiednim ustawieniu, obraz można zapisać w formacie PDF.

Rysunek Przykłady brył fotometrycznych uzyskanych dla różnych opraw wykreślonych w aplikacji

## 4.2. Weryfikacja poprawności aplikacji narzędziami inżynierii oprogramowania

Weryfikacja poprawności wykonania projektu odbyła się poprzez:

* Kompilowanie projektu kompilatorem g++6.3 używanym przez Qt Creatora. Kompilator w finalnej fazie nie sygnalizował żadnych ostrzeżeń.
* Analizę statyczną kodu narzędziem Clang. Aplikacja ta nie zgłaszała żadnych ostrzeżeń w finalnej fazie projektu. Raport z Clang Static Analyzer znajduje się w *Załączniku 2.*
* Narzędziem do weryfikowana użycia pamięci Valgrind. Pełen raport znajduje się w *Załączniku 3*. Należy zauważyć, że aplikacja nie jest całkowicie wolna od błędów. Podczas typowego użytkowania (przeczytanie danych z pliku, wykreślenie kilku krzywych fotometrycznych, bryły oraz policzenie strumienia świetlnego), nie zwalnia kilkudziesięciu kilobajtów danych. Jednak nie jest to wartość, która mogłaby spowodować jakiekolwiek problemy w użytkowaniu prostego programu z interfejsem użytkownika, który nie jest wielokrotnie używaną biblioteką kumulującą każdy najmniejszy wyciek pamięci przez kilka miesięcy cyklu włączenia produkcyjnego serwera. Valgrind zgłasza również problemy z samą biblioteką Qt.
* Autor pracy starał się sumiennie obsłużyć każdą sytuacje wyjątkową którą mógł napotkać użytkownik.

# 4. Wnioski

## 4.1. Zalecenia do dalszego rozwoju aplikacji

* Producent spektrofotometru udostępnia bezpłatną bibliotekę do obsługi przyrządu pomiarowego[14], więc istnieje możliwość integracji urządzenia z aplikacją do pomiaru krzywych fotometrycznych.
* Znacznie dokładniejsze wyliczenie strumienia świetlnego możliwe byłoby poprzez dodanie możliwości interpolacji krzywej południka z punktów pomiarowych (metoda Lagrange’a wydaje się tu najodpowiedniejsza ze względu na jej prostotę, dokładność oraz dostępność bibliotek ją implementujących). Znacznie zwiększyłoby to ilość punktów na sferze z których liczony jest strumień.
* Plik z danymi pomiarowymi zapisywany jest w najprostszym z formatów CVS. Istnieje możliwość zapisywania pliku w formatach LDT lub IES w których zazwyczaj przechowuje się dane opisujące właściwości opraw świetlnych.

## 4.2. Podsumowanie

# Literatura

[1] E. Szwaba, Praca dyplomowa inżynierska „Stanowisko do pomiarów krzywych fotometrycznych oraz całkowitego strumienia świetlnego”, Wrocław 2015

[9] W. Żagań, Podstawy Techniki Świetlnej, Warszawa 2014, Oficyna Wydawnicza Politechniki Warszawskiej.

[10] Wyznaczanie Bryły Fotometrycznej Światłości – Instrukcja do ćwiczenia z Podstaw Techniki Świetlnej <http://lumen.iee.put.poznan.pl/studenci/niestacjonarne/Lab-2-bryla_fotom.pdf>, dostęp dnia 2016-12-06

[2] <http://www.ies.org/>; dostęp dnia 2016-12-04

[3] Talking Photometry: Understanding Photometric Data Formats; <http://www.photometrictesting.co.uk/File/understanding_photometric_data_files.php>; dostęp dnia 2016-12-04

[4] Strona domowa programu QLumEdit2 https://sourceforge.net/projects/qlumedit2/ , dostęp dnia 2016-12-04

[5] Strona domowa programu IESviewer <http://www.photometricviewer.com/> dostęp dnia 2016-12-04

[6] Strona domowa producenta Photometrics Pro - Luminaire Analysis Software <http://www.photometricspro.com/index.html>, dostęp dnia 2016-12-04

[7] Strona domowa programu Eulumdat Tools <http://eulumdat.blogspot.com/>, dostęp dnia 2016-12-06

[8] Strona domowa programu LDTediitor <http://ldteditor.software.informer.com/>, dostęp dnia 2016-12-06

[11] W. Golik i inni, Laboratorium z techniki świetlnej Skrypt nr 1792. Wydawnictwo Politechniki Poznańskiej, Poznań 1994.

[12] Clang Static Analyzer - http://clang-analyzer.llvm.org/ ; dostęp dnia 2017-01-07

[13] Google C++ Style Guide - <https://google.github.io/styleguide/cppguide.html> ; dostęp dnia 2017-01-08

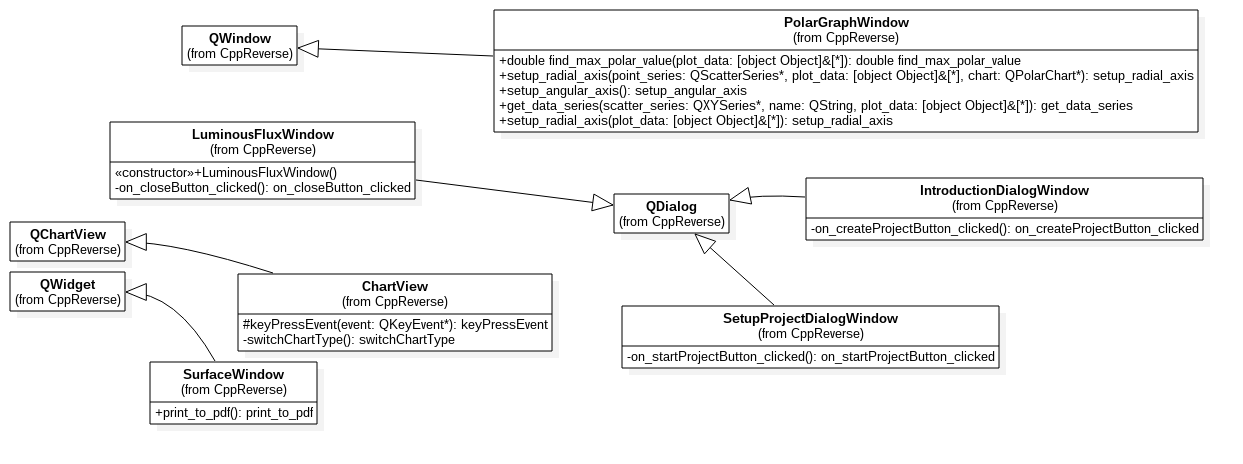
[14] Strona domowa Stellarnet – <http://www.stellarnet.us/software/#LINUX> ; dostęp dnia 2017-01-7

[15] Siatka, którą wykorzystano jako bazę do przygotowania obraz - <https://en.wikipedia.org/wiki/Sphere#/media/File:Sphere_wireframe_10deg_6r.svg>

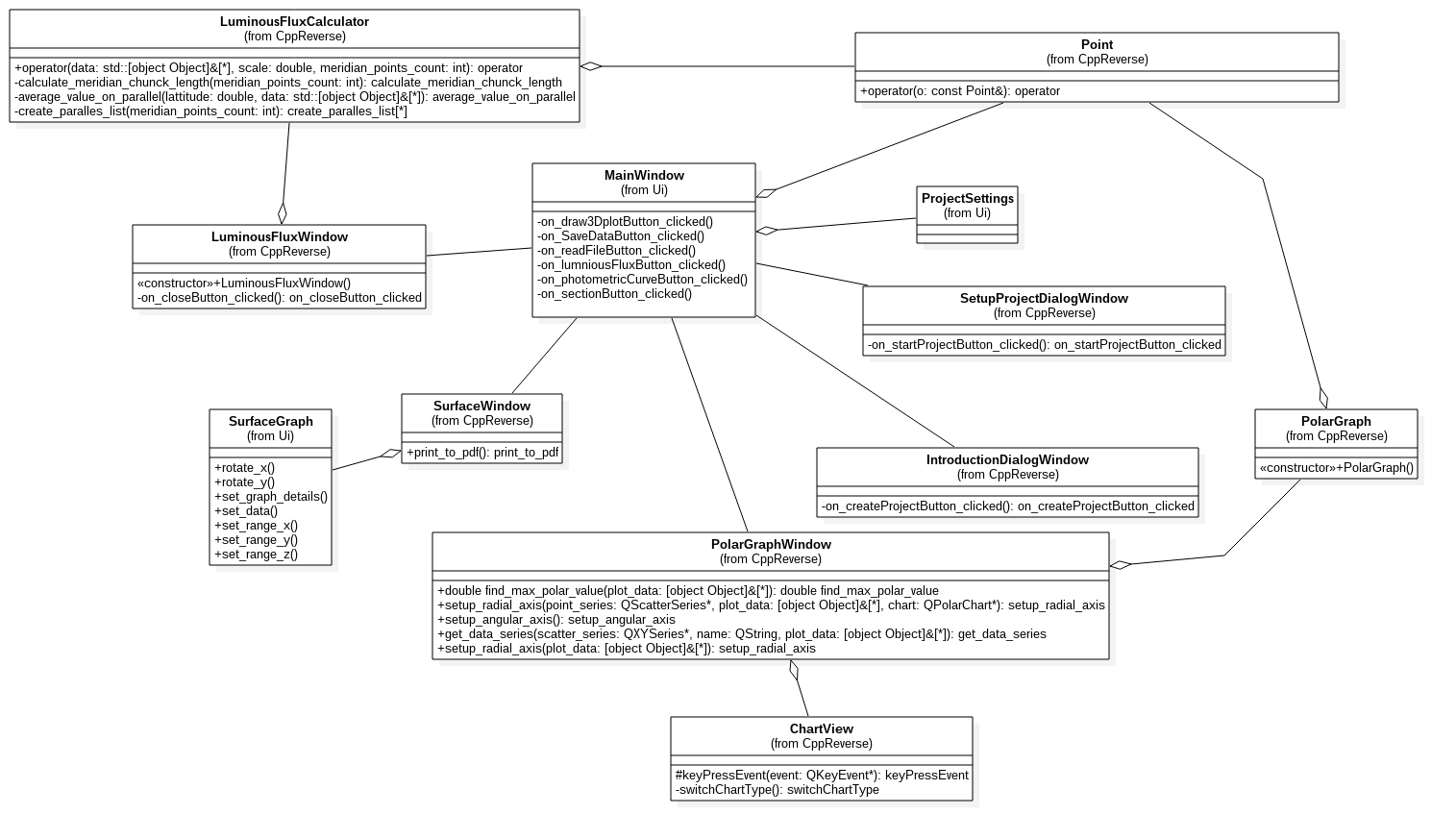
dostęp dnia 2016-01-05

# Załączniki

1. Hierarchia typów klas projektu



1. Diagram klas stworzonych w aplikacji



1. Skrócony wydruk z analizy statycznej kodu przeprowadzonej narzędziem Clang Static Analyzer

**#:~/devel/build-photometric-curves-a-Desktop\_Qt\_5\_7\_0\_GCC\_64bit-Debug$ scan-build-3.6 make**

scan-build: Using '/usr/lib/llvm-3.6/bin/clang' for static analysis

/home/tomasz/Qtsec/5.7/gcc\_64/bin/uic ../photometric-curves/mainwindow.ui -o ui\_mainwindow.h

/home/tomasz/Qtsec/5.7/gcc\_64/bin/uic ../photometric-curves/luminous-flux-window.ui -o ui\_luminous-flux-window.h

/home/tomasz/Qtsec/5.7/gcc\_64/bin/uic ../photometric-curves/introduction-dialog-window.ui -o ui\_introduction-dialog-window.h

/home/tomasz/Qtsec/5.7/gcc\_64/bin/uic ../photometric-curves/setup-project-dialog-window.ui -o ui\_setup-project-dialog-window.h

/usr/share/clang/scan-build-3.6/c++-analyzer -c -pipe -g -std=gnu++11 -Wall -W -D\_REENTRANT -fPIC -DQT\_QML\_DEBUG -DQT\_PRINTSUPPORT\_LIB -DQT\_CHARTS\_LIB -DQT\_WIDGETS\_LIB -DQT\_DATAVISUALIZATION\_LIB -DQT\_GUI\_LIB -DQT\_CORE\_LIB -I../photometric-curves -I. -I../../Qtsec/5.7/gcc\_64/include -I../../Qtsec/5.7/gcc\_64/include/QtPrintSupport -I../../Qtsec/5.7/gcc\_64/include/QtCharts -I../../Qtsec/5.7/gcc\_64/include/QtWidgets -I../../Qtsec/5.7/gcc\_64/include/QtDataVisualization -I../../Qtsec/5.7/gcc\_64/include/QtGui -I../../Qtsec/5.7/gcc\_64/include/QtCore -I. -I. -I../../Qtsec/5.7/gcc\_64/mkspecs/linux-g++ -o main.o ../photometric-curves/main.cpp

/usr/share/clang/scan-build-3.6/c++-analyzer -c -pipe -g -std=gnu++11 -Wall -W -D\_REENTRANT -fPIC -DQT\_QML\_DEBUG -DQT\_PRINTSUPPORT\_LIB -DQT\_CHARTS\_LIB -DQT\_WIDGETS\_LIB -DQT\_DATAVISUALIZATION\_LIB -DQT\_GUI\_LIB -DQT\_CORE\_LIB -I../photometric-curves -I. -I../../Qtsec/5.7/gcc\_64/include -I../../Qtsec/5.7/gcc\_64/include/QtPrintSupport -I../../Qtsec/5.7/gcc\_64/include/QtCharts -I../../Qtsec/5.7/gcc\_64/include/QtWidgets -I../../Qtsec/5.7/gcc\_64/include/QtDataVisualization -I../../Qtsec/5.7/gcc\_64/include/QtGui -I../../Qtsec/5.7/gcc\_64/include/QtCore -I. -I. -I../../Q

[.....]

g++ -Wl,-rpath,/home/tomasz/Qtsec/5.7/gcc\_64/lib -o photometric-curves main.o polar-graph-window.o point.o luminous-flux-calculator.o luminous-flux-window.o introduction-dialog-window.o setup-project-dialog-window.o surface-window.o surface-graph.o project-settings.o polar-graph.o polar-graph-view.o main-window.o moc\_polar-graph-window.o moc\_luminous-flux-window.o moc\_introduction-dialog-window.o moc\_setup-project-dialog-window.o moc\_surface-window.o moc\_surface-graph.o moc\_main-window.o -L/home/tomasz/Qtsec/5.7/gcc\_64/lib -lQt5PrintSupport -L/usr/lib64 -lQt5Charts -lQt5Widgets -lQt5DataVisualization -lQt5Gui -lQt5Core -lGL -lpthread

scan-build: Removing directory '/tmp/scan-build-2017-01-11-183337-4773-1' because it contains no reports.

**scan-build: No bugs found.**

1. Wydruk sposobu używania pamięci z programu Valgrind

**$ valgrind ./photometric-curves**

tomasz@wilkbook:~/devel/build-photometric-curves-a-Desktop\_Qt\_5\_7\_0\_GCC\_64bit-Debug$ valgrind ./photometric-curves | tee ~/log-valgrind

==5121== Memcheck, a memory error detector

==5121== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.

==5121== Using Valgrind-3.12.0 and LibVEX; rerun with -h for copyright info

==5121== Command: ./photometric-curves

==5121==

==5121== Conditional jump or move depends on uninitialised value(s)

==5121== at 0x4C31EA6: rawmemchr (vg\_replace\_strmem.c:1402)

==5121== by 0x7C013A1: \_IO\_str\_init\_static\_internal (strops.c:41)

==5121== by 0x7BF4B66: vsscanf (iovsscanf.c:40)

==5121== by 0x7BEF2D6: sscanf (sscanf.c:32)

==5121== by 0xCCB1E4A: ??? (in /usr/lib/x86\_64-linux-gnu/libdrm.so.2.4.0)

==5121== by 0xCCB2182: ??? (in /usr/lib/x86\_64-linux-gnu/libdrm.so.2.4.0)

==5121== by 0xCCB5BF8: drmGetDevice (in /usr/lib/x86\_64-linux-gnu/libdrm.so.2.4.0)

==5121== by 0x6EAA266: ??? (in /usr/lib/x86\_64-linux-gnu/libGL.so.1.2.0)

==5121== by 0x6EA9624: ??? (in /usr/lib/x86\_64-linux-gnu/libGL.so.1.2.0)

==5121== by 0x6E7D978: ??? (in /usr/lib/x86\_64-linux-gnu/libGL.so.1.2.0)

==5121== by 0x6E79541: glXGetFBConfigs (in /usr/lib/x86\_64-linux-gnu/libGL.so.1.2.0)

==5121== by 0x6E7A8D1: glXChooseFBConfig (in /usr/lib/x86\_64-linux-gnu/libGL.so.1.2.0)

==5121==

==5121== Syscall param writev(vector[...]) points to uninitialised byte(s)

==5121== at 0x7C6BD7D: ??? (syscall-template.S:84)

==5121== by 0xC46C41C: ??? (in /usr/lib/x86\_64-linux-gnu/libxcb.so.1.1.0)

==5121== by 0xC46C81C: ??? (in /usr/lib/x86\_64-linux-gnu/libxcb.so.1.1.0)

==5121== by 0xC46CAB6: ??? (in /usr/lib/x86\_64-linux-gnu/libxcb.so.1.1.0)

==5121== by 0xC46D2CB: xcb\_flush (in /usr/lib/x86\_64-linux-gnu/libxcb.so.1.1.0)

==5121== by 0xDB92C2F: QXcbWindow::hide() (in /home/tomasz/Qtsec/5.7/gcc\_64/lib/libQt5XcbQpa.so.5.7.0)

==5121== by 0x609EE64: QWindow::setVisible(bool) (in /home/tomasz/Qtsec/5.7/gcc\_64/lib/libQt5Gui.so.5.7.0)

==5121== by 0x557666E: QWidgetPrivate::hide\_sys() (in /home/tomasz/Qtsec/5.7/gcc\_64/lib/libQt5Widgets.so.5.7.0)

==5121== by 0x557C7AC: QWidgetPrivate::hide\_helper() (in /home/tomasz/Qtsec/5.7/gcc\_64/lib/libQt5Widgets.so.5.7.0)

==5121== by 0x55817E7: QWidget::setVisible(bool) (in /home/tomasz/Qtsec/5.7/gcc\_64/lib/libQt5Widgets.so.5.7.0)

==5121== by 0x571BB76: QDialog::setVisible(bool) (in /home/tomasz/Qtsec/5.7/gcc\_64/lib/libQt5Widgets.so.5.7.0)

==5121== by 0x571A3D1: QDialog::done(int) (in /home/tomasz/Qtsec/5.7/gcc\_64/lib/libQt5Widgets.so.5.7.0)

==5121== Address 0xd593759 is 4,537 bytes inside a block of size 21,152 alloc'd

==5121== at 0x4C2CBC5: calloc (vg\_replace\_malloc.c:711)

==5121== by 0xC46BDCB: xcb\_connect\_to\_fd (in /usr/lib/x86\_64-linux-gnu/libxcb.so.1.1.0)

==5121== by 0xC46FD28: xcb\_connect\_to\_display\_with\_auth\_info (in /usr/lib/x86\_64-linux-gnu/libxcb.so.1.1.0)

==5121== by 0xC161E89: \_XConnectXCB (in /usr/lib/x86\_64-linux-gnu/libX11.so.6.3.0)

==5121== by 0xC152BC1: XOpenDisplay (in /usr/lib/x86\_64-linux-gnu/libX11.so.6.3.0)

==5121== by 0xDB7E095: QXcbConnection::QXcbConnection(QXcbNativeInterface\*, bool, unsigned int, char const\*) (in /home/tomasz/Qtsec/5.7/gcc\_64/lib/libQt5XcbQpa.so.5.7.0)

==5121== by 0xDB81AEA: QXcbIntegration::QXcbIntegration(QStringList const&, int&, char\*\*) (in /home/tomasz/Qtsec/5.7/gcc\_64/lib/libQt5XcbQpa.so.5.7.0)

==5121== by 0xD93C3EA: ??? (in /home/tomasz/Qtsec/5.7/gcc\_64/plugins/platforms/libqxcb.so)

==5121== by 0x608341A: QPlatformIntegrationFactory::create(QString const&, QStringList const&, int&, char\*\*, QString const&) (in /home/tomasz/Qtsec/5.7/gcc\_64/lib/libQt5Gui.so.5.7.0)

==5121== by 0x6090483: QGuiApplicationPrivate::createPlatformIntegration() (in /home/tomasz/Qtsec/5.7/gcc\_64/lib/libQt5Gui.so.5.7.0)

==5121== by 0x6090F4C: QGuiApplicationPrivate::createEventDispatcher() (in /home/tomasz/Qtsec/5.7/gcc\_64/lib/libQt5Gui.so.5.7.0)

==5121== by 0x69BB9CC: QCoreApplicationPrivate::init() (in /home/tomasz/Qtsec/5.7/gcc\_64/lib/libQt5Core.so.5.7.0)

==5121==

Warning: Flat qualifier not supported on your platform's GLSL language. Requires at least GLSL version 1.2 with GL\_EXT\_gpu\_shader4 extension.

==5121== Warning: noted but unhandled ioctl 0x6458 with no size/direction hints.

==5121== This could cause spurious value errors to appear.

==5121== See README\_MISSING\_SYSCALL\_OR\_IOCTL for guidance on writing a proper wrapper.

Warning: Flat qualifier not supported on your platform's GLSL language. Requires at least GLSL version 1.2 with GL\_EXT\_gpu\_shader4 extension.

==5121==

==5121== HEAP SUMMARY:

==5121== in use at exit: 6,637,889 bytes in 38,509 blocks

==5121== total heap usage: 1,660,869 allocs, 1,622,360 frees, 402,301,359 bytes allocated

==5121==

**==5121== LEAK SUMMARY:**

**==5121== definitely lost: 7,011 bytes in 152 blocks**

**==5121== indirectly lost: 354,141 bytes in 4,275 blocks**

**==5121== possibly lost: 628 bytes in 4 blocks**

**==5121== still reachable: 6,276,109 bytes in 34,078 blocks**

**==5121== of which reachable via heuristic:**

**==5121== newarray : 4,584 bytes in 9 blocks**

**==5121== multipleinheritance: 528 bytes in 1 blocks**

==5121== suppressed: 0 bytes in 0 blocks

==5121== Rerun with --leak-check=full to see details of leaked memory

==5121==

==5121== For counts of detected and suppressed errors, rerun with: -v

==5121== Use --track-origins=yes to see where uninitialised values come from

**==5121== ERROR SUMMARY: 57 errors from 2 contexts (suppressed: 0 from 0)**