

Warszawa 2019

prof. nzw. dr hab. inż. Włodzimierz Smoliak  
Promotor

Adam Jędrzejowski  
nr albumu 277417

Wielopłatformowa przeglądarka obrazów DICOM w C++

w specjalności Elektronika i Informatyka w medycynie  
na kierunku Elektronika

# Praca dyplomowa

# Inżynierska

Zakład Elektroniki Jądrowej i Medycznej  
Instytut Radiowej Elektroniki i Technik Multimedialnych



Politechnika Warszawska  
WYDZIAŁ ELEKTRONIKI  
I TECHNIK INFORMATYCZNA

## **Wieloplatformowa przeglądarka obrazów DICOM w C++**

Praca składa się z sześciu rozdziałów: wstęp, obrazowanie diagnostyczne, biblioteki i narzędzia, implementacja, komplikacja oraz podsumowanie. Wstęp jest wprowadzeniem do tematu i celu pracy.

W drugim rozdziale jest opisane zagadnienie problemowe związane z obrazami w medycynie. Wymienione są techniki diagnostyczne oraz ich podstawowe różnice. Przedstawione są parametry cyfrowych obrazów w medycynie. Ponadto opisano prezentacje obrazów medycznych oraz wyjaśniono czym są przeglądarki obrazów. Omówione są posiadane przez nie funkcje. Opisano format zapisu cyfrowych obrazów medycznych, standard DICOM.

Trzeci rozdział opisuje biblioteki i narzędzia użyte w czasie pisania pracy inżynierskiej. Wyjaśnione są cele użycia narzędzia CMake i jego zalety. Opisano bibliotekę Qt, jej możliwości, drzewa obiektów implementowane przez nią i sposób konstrukcji programowania zdarzeniowego w niej zawartego. Przedstawiono i uzasadniono wybór biblioteki GDCM jako biblioteki do obsługi i wczytywania plików DICOM.

W czwartym rozdziale przedstawiono sposób implementacji pracy. Określono przewidywany zakres implementowanych funkcji oprogramowania. Opisano graficzny interfejs użytkownika i jego funkcje programu. Wyjaśniono projekt struktury obiektowej programu. Następnie szczegółowo opisano strukturę danych wraz z klasami C++. Tam gdzie była możliwość załączony jest diagram UML. Opisano wszystkie algorytmy przetwarzania danych w celu lepszej wizualizacji obrazu.

W piątym rozdziale opisano przebieg komplikacji kodu źródłowego.

**Słowa kluczowe:** medyczne diagnostyczne techniki obrazowe; standard DICOM; przeglądarka obrazów medycznych; wyświetlanie obrazów medycznych; C++; Qt; GDCM

## Multi-platform DICOM image viewer in C++

The work consists of six chapters: introduction, diagnostic imaging, libraries and tools, implementation, compilation and summary. The first chapter is an introduction to the subject and the purpose of the work.

The second chapter describes problems that are related to images in medicine. Diagnostic techniques and their basic differences are listed in this part. There are presented the static techniques and the dynamic ones. In addition the representations of the images are described and it is explained what viewers are. Their functions are depicted. The format for recording digital medical images, DICOM standard, is described.

The third chapter describes the libraries and tools that were used to write the engine. The purpose of using the CMake tool and its advantages are explained. The engine work. The library, its capabilities, object trees implemented by it and the way of programming Qt library, its capabilities, objects implemented by it and the choice of the DICOM construction have been described for the events contained in it. The choice of the DICM library was presented and justified as a library for handling and loading DICOM files.

The fourth chapter presents the way in which the work is implemented. The graphical range of the implemented functions of the software has been determined. The source code of the implementation of the software has been described. The interface user interface and its program functions are described. The design of the object structure of the program has been explained. The structure of the data is described for better visualization of the images.

The fifth chapter describes the process of compilation of the source code. All data processing algorithms are described for better visualization of the images. Classes is then described in details. Where it was possible a UML diagram was included. Of the program has been explained. The structure of the data, together with the C++ user interface and its program functions are described. The design of the object structure of the program has been explained. The structure of the data is described for better visualization of the images.

**Keywords:** medical diagnostic techniques; DICOM; medical image viewer; medical image viewer; C++; Qt; GDICM

## Bibliografia

- [1] Thomas M. Deserno Daniel Haak, Charles-E. Page. A survey of dicom viewer software to integrate clinical research and medical imaging. *J Digit Imaging*, 29:206–215, 2016.

Spis rysunków

Wydomy odpowiadaliśmy kartę za skradanej fakszywy zeznania oswiadczań, że nieznały dyplomu i dyplomowa została napisała prezesa mniej samodzielnie, pod opieką kierującego.

OSWADCZENIE

POLITECHNIKA WARSZAWSKA  
Wydział Elektroniki i Technik Informacyjnych Politechniki Warszawskiej  
Warszawa, 05 czerwca 2019

- |      |  |     |  |    |  |
|------|--|-----|--|----|--|
| 7.7  | Wyswietlanie informacji o aktualnych projektach w programie DICOM Viewer             | 2.2 | Wyświetlanie informacji o aktualnych projektach w programie DICOM Viewer                     | 9  | • informacja praca dyplomowa nie zawiera danych i informacji, które uzywane są w sposobu posiadania  |
| 2.3  | Generowanie obrazów 3D z wielu okienekDICOM Viewer                                   | 10  | DICOM Viewer 3D Pro. Wyświetlanie obrazów 3D z wielu okienekDICOM Viewer                     | 10 | • informacja praca dyplomowa nie zawiera danych i informacji, które uzywane są w sposobie posiadania |
| 2.4  | Rekonstrukcja wiłolipaszczynowa w programie DICOM Viewer                             | 11  | Szare DICOM Viewer 3D Pro. Wyświetlanie obrazów 3D z wielu okienekDICOM Viewer               | 11 | • informacja praca dyplomowa nie zawiera danych i informacji, które uzywane są w sposobie posiadania |
| 2.5  | ELEMENTY ZAŁOGI MEDICAL HARBOUR. Wyświetlanie obrazów 3D z wielu okienekDICOM Viewer | 11  | Zdjęcie złytye za złogą Medical Harbour. Wyświetlanie obrazów 3D z wielu okienekDICOM Viewer | 11 | • informacja praca dyplomowa nie zawiera danych i informacji, które uzywane są w sposobie posiadania |
| 3.1  | Prywatność odkrytego programu w Qt. Wyświetlanie                                     | 23  | Przykładowe odkrytego programu w Qt. Wyświetlanie  | 23 | • informacja praca dyplomowa nie zawiera danych i informacji, które uzywane są w sposobie posiadania |
| 4.1  | Okno przyglądarki tuz po uruchomieniu. Wyświetlanie                                  | 29  | Okno przyglądarki tuz po uruchomieniu. Wyświetlanie  | 29 | • informacja praca dyplomowa nie zawiera danych i informacji, które uzywane są w sposobie posiadania |
| 4.2  | Okno przyglądarki z wczytanymi kilkoma obrazami. Wyświetlanie                        | 30  | Okno przyglądarki z wczytanymi kilkoma obrazami. Wyświetlanie                                | 30 | • informacja praca dyplomowa nie zawiera danych i informacji, które uzywane są w sposobie posiadania |
| 4.3  | Prywatność obrazów monochromatycznych. Wyświetlanie                                  | 31  | Prywatność obrazów monochromatycznych. Wyświetlanie  | 31 | • informacja praca dyplomowa nie zawiera danych i informacji, które uzywane są w sposobie posiadania |
| 4.4  | DiagRAM klas UML przedstawiona struktury programu. Wyświetlanie                      | 32  | DiagRAM klas UML przedstawiona struktury programu. Wyświetlanie                              | 32 | • informacja praca dyplomowa nie zawiera danych i informacji, które uzywane są w sposobie posiadania |
| 4.5  | DiagRAM klas UML przedstawiona struktury programu. Wyświetlanie                      | 34  | DiagRAM klas UML przedstawiona struktury programu. Wyświetlanie                              | 34 | • informacja praca dyplomowa nie zawiera danych i informacji, które uzywane są w sposobie posiadania |
| 4.6  | Diagram klas UML przedstawiona struktury programu. Wyświetlanie                      | 35  | Diagram klas UML przedstawiona struktury programu. Wyświetlanie                              | 35 | • informacja praca dyplomowa nie zawiera danych i informacji, które uzywane są w sposobie posiadania |
| 4.7  | Diagram klas UML przedstawiona struktury programu. Wyświetlanie                      | 36  | Diagram klas UML przedstawiona struktury programu. Wyświetlanie                              | 36 | • informacja praca dyplomowa nie zawiera danych i informacji, które uzywane są w sposobie posiadania |
| 4.8  | Diagram klas UML przedstawiona struktury programu. Wyświetlanie                      | 40  | Diagram klas UML przedstawiona struktury programu. Wyświetlanie                              | 40 | • informacja praca dyplomowa nie zawiera danych i informacji, które uzywane są w sposobie posiadania |
| 4.9  | Zakładka wraz z numeracją elementów interfejsu. Wyświetlanie                         | 42  | Zakładka wraz z numeracją elementów interfejsu. Wyświetlanie                                 | 42 | • informacja praca dyplomowa nie zawiera danych i informacji, które uzywane są w sposobie posiadania |
| 4.10 | Wykres linijkowy określony dla C1720 W1208 z histogramem w tl. Wyświetlanie          | 50  | Wykres linijkowy określony dla C1720 W1208 z histogramem w tl. Wyświetlanie                  | 50 | • informacja praca dyplomowa nie zawiera danych i informacji, które uzywane są w sposobie posiadania |
| 4.11 | Porównanie jednego obrazu z trzech „ołącz”, odpowiadające C40 W80, C50               | 51  | Porównanie jednego obrazu z trzech „ołącz”, odpowiadające C40 W80, C50                       | 51 | • informacja praca dyplomowa nie zawiera danych i informacji, które uzywane są w sposobie posiadania |
| 4.12 | Paleta Hot Iron (na środku) i Hot Metal Blue (po prawej) w porównaniu                | 51  | Paleta Hot Iron (na środku) i Hot Metal Blue (po prawej) w porównaniu                        | 51 | • informacja praca dyplomowa nie zawiera danych i informacji, które uzywane są w sposobie posiadania |
| 4.13 | Wizualizacja układu osi współrzędnych kartezjańskich pacjenta. Wyświetlanie          | 51  | Wizualizacja układu osi współrzędnych kartezjańskich pacjenta. Wyświetlanie                  | 51 | • informacja praca dyplomowa nie zawiera danych i informacji, które uzywane są w sposobie posiadania |
| 4.14 | Podział płaszczyzny sceny. Wyświetlono osiem części. Wyświetlanie                    | 60  | Podział płaszczyzny sceny. Wyświetlono osiem części. Wyświetlanie                            | 60 | • informacja praca dyplomowa nie zawiera danych i informacji, które uzywane są w sposobie posiadania |
| 4.15 | Prywatność obrazu medycznego (przełożony przez MRF) z orientacją                     | 63  | Prywatność obrazu medycznego (przełożony przez MRF) z orientacją                             | 63 | • informacja praca dyplomowa nie zawiera danych i informacji, które uzywane są w sposobie posiadania |
| 4.16 | Przykładowy obraz medyczny (przełożony przez MRF) z orientacją                       | 63  | Przykładowy obraz medyczny (przełożony przez MRF) z orientacją                               | 63 | • informacja praca dyplomowa nie zawiera danych i informacji, które uzywane są w sposobie posiadania |

## Rozdział 6

### Podsumowanie

Celem pracy inżynierskiej było napisanie aplikacji do obsługi obrazów DICOM w C++ z możliwością komplikacji na wiele platform. Cel udało się osiągnąć. Użycie języka C++ umożliwiło wykorzystanie całego potencjału obliczeniowego maszyny. Zastosowano biblioteki dostępne na różnych platformach: Qt i GDCM, które również zostały napisane w C++, dzięki czemu uzyskano jednolity program napisany w jednym języku. Zapewniono jednolity sposób komplikacji na platformach przy użyciu narzędzia CMake. Aplikacja działa w ten sam sposób na wszystkich testowanych platformach: Linux, MacOS i Windows. Jednolity wygląd aplikacji zapewniła biblioteka Qt, co sprawia, że interfejs aplikacji jest prawie taki sam na każdym systemie.

Zaplanowano i dodano obsługę podstawowych operacji na obrazie ułatwiających jego oglądanie i ocenienie, takich jak: przenoszenie; skalowanie; obrót. Zaimplementowano pseudokolorowanie obrazów monochromatycznych z możliwością dodawania nowych palet. Wprowadzono obsługę serii obrazów jako całości, włączając w to przegląd obrazów w serii, animacje, wspólne okna w skali barwnej oraz wspólne przekształcenia macierzowe.

Napotkano problem z biblioteką GDCM w postaci braku możliwości używania plików binarnych dostarczonych przez twórców. Te pliki binarne zostały skompilowane za pomocą innego kompilatora niż pliki binarne Qt. Spowodowało to, że typ std::string z jednej biblioteki nie jest kompatybilny z std::string z drugiej biblioteki. Wynika to z użycia innych interfejsów binarnych aplikacji (ang. *application binary interface*) w różnych kompilatorach. Problem można rozwiązać komplikując bibliotekę GDCM własnoręcznie.

Spis třesčí

5.5 Kompilacija Sokar

- W przypadku Linuxa i MacOS, w których "make", "path/gdcm-bin/GDCM.sln" i kliniki przymykały się do bieżącego katalogu, należy zmodyfikować plik "path/gdcm-bin/GDCM.sln" w katalogu "Windowsa otwórz plik", aby zidentyfikować ścieżkę do katalogu "gdcm-bin".

Komplikacja zawału polowego, której nazwą jest **występować kolapsacją**:

- Uruchom CMakre z menu programu lub za pomocą poleceńia „cmake-gui”.

W Point," where is the source code: "wpisz," /path/sokrat-app/;

- w pollu "Where to build the binaries;" wplzz "path/sokar-app-bin/";

‘W

- ustawa parametr wartości „*OtDir*” na której do skompilowanej biblioteki *Ot*,

-q-a

- zas w przypadku Windowsa otworzy "path/gdcm-bin/Sokar.sln" i kliknij przycisk "Locally debugger Windows".

5.6 Uruchomienie

VW przypada na Linuxa i MACOS, plik binary zaznajduje się w katalogu „/path/sokar-app-bin”. Mozilla go uruchomisz go klikającą link z terminala za pomocą komendy „./Sokar”. W przy- padku Windows plik binary zaznajduje się w folderze „/path/sokar-app-bin/Debug”.

- |       |  |    |
|-------|--|----|
| 3     | Obrzowane diagnozyne w medycynie         |    |
| 2.2   | Parametry obrzow                         |    |
| 2.2.1 | Podstawa parametry obrazu cyfrowego      | 5  |
| 2.2.2 | Kontраст                                 | 5  |
| 2.2.3 | Rozdzielenie                             | 7  |
| 2.2.4 | Słosunek sygnału do szumu (SNR)          | 7  |
| 2.2.5 | Pozitom artefaktów                       | 8  |
| 2.2.6 | Pozitom zniekształceń przestrzennych     | 8  |
| 2.3   | Przestącająca obrzawy medycznych         | 8  |
| 2.3.1 | Przegądarki obrzaw                       | 8  |
| 2.3.2 | Funkcje przegądarki obrzaw               | 8  |
| 2.3.3 | Kryteria porównywania przegądarki obrzaw | 12 |
| 2.4   | Formaty cyfrowych obrazów medycznych     | 13 |
| 2.4.1 | Standard DICOM V3.0                      | 13 |
| 2.4.2 | Sposób zapisu danych w pliku DICOM       | 14 |
| 2.4.3 | DICOMDIR                                 | 17 |
| 2.4.4 | Forme formaty zapisu                     | 17 |
| 3.1   | CMakre                                   | 18 |
| 3.2   | QT                                       | 18 |
| 3.2.1 | Wymowa                                   | 19 |
| 3.2.2 | Licencja                                 | 19 |
| 3.2.3 | Normy i certyfikaty                      | 19 |
| 3.2.4 | Globalne typy struktur                   | 19 |
| 3.2.5 | Klasa Obiekt                             | 20 |
| 3.2.6 | Graficzny interfejs użytkownika          | 22 |
| 3.3   | GDCM                                     | 22 |
| 3.3.1 | Uzasadnienie wybór                       | 23 |
| 3.3.2 | Opis                                     | 24 |
| 3.3.3 | Licencja                                 | 24 |
| 3.3.4 | Podstawowe klasy                         | 24 |
| 3.4   | Git                                      | 25 |

<b>4 Implementacja</b>	<b>28</b>
4.1 Zakres implementacji . . . . .	28
4.2 Wieloplatformowość . . . . .	29
4.3 Graficzny interfejs użytkownika . . . . .	29
4.4 Projekt struktury obiektowej programu . . . . .	31
4.5 Struktury danych . . . . .	32
4.5.1 Konwertowanie danych ze znaczników . . . . .	32
4.5.2 Scena . . . . .	33
4.5.3 Kolekcje scen . . . . .	40
4.5.4 Zakładka . . . . .	42
4.5.5 Obiekt zakładek . . . . .	45
4.5.6 Okno główne programu . . . . .	46
4.6 Algorytmy . . . . .	47
4.6.1 Cykl generowania obrazów . . . . .	47
4.6.2 Generowania obrazu monochromatycznego . . . . .	49
4.6.3 Tworzenie transformat i ich użycie na obrazie . . . . .	57
4.6.4 Ustalanie pozycji pacjenta względem sceny . . . . .	59
<b>5 Kompilacja</b>	<b>64</b>
5.1 Narzędzia potrzebne do komplikacji . . . . .	64
5.2 Biblioteki potrzebne do komplikacji . . . . .	64
5.2.1 Instalacja Qt . . . . .	64
5.2.2 Pobranie kodu źródłowego GDCM . . . . .	65
5.2.3 Pobranie kodu źródłowego Sokar . . . . .	65
5.3 Przygotowanie katalogów . . . . .	65
5.4 Kompilacja GDCM . . . . .	65
5.5 Kompilacja Sokar . . . . .	66
5.6 Uruchomienie . . . . .	66
<b>6 Podsumowanie</b>	<b>67</b>

automatycznie pobrany plik instalacyjny. Po pobraniu należy go otworzyć i postępować zgodnie z instalacją.

W pewnym momencie użytkownik może zostać poproszony o dane kontaktowe. Nie jest to wymagane i można kliknąć przycisk „Skip”.

Następnie należy wybrać komponenty do zainstalowania. W przypadku Windowsa należy zainstalować wersje „Qt 5.12.3 MSVC 2017 64-bit”. Z kolei na MacOS należy zainstalować „Qt 5.12.3 clang\_x64”. Można wyłączyć wszystkie inne opcje, nie są one wymagane do komplikacji programu. Dalej należy postępować zgodnie z instrukcjami pojawiającymi się na ekranie.

### 5.2.2 Pobranie kodu źródłowego GDCM

Program był testowany na wersji 2.8.9, z tego powodu zalecanym jest używanie tej wersji.

Należy udać się na stronę <https://github.com/malaterre/GDCM/releases/tag/v2.8.9> i pobrać plik „Source code (zip)”, a następnie go rozpakować.

### 5.2.3 Pobranie kodu źródłowego Sokar

Kod źródłowy aplikacji można pobrać repozytorium git znajdującego się pod adresem <https://gl.ire.edu.pl/ajedrzejowski/sokar-app>.

## 5.3 Przygotowanie katalogów

Należy utworzyć folder, w którym będą znajdowały się wszystkie foldery z plikami, dalej ten folder będzie nazywany „/path/”. Kod źródłowy GDCM należy umieścić w katalogu „/path/gdcm/”. Kod źródłowy Sokar należy umieścić w katalogu „/path/sokar-app/”. Powinno się również utworzyć foldery „/path/gdcm-bin/” i „/path/sokar-app-bin/”.

## 5.4 Kompilacja GDCM

Kompilacja zawiera polecenia, które należy wykonać w następującej kolejności:

- uruchom CMake z menu programów lub za pomocą polecenia „cmake-gui”,
- w polu „Where is the source code?” wpisz „/path/gdcm/”,
- w polu „Where to build the binaries?” wpisz „/path/gdcm-bin/”,
- kliknij przycisk „Configure” znajdujący się w dolnej lewej części okna,
- w oknie zatytułowanym „CMakeSetup” wybierz opcje: „Unix Makefiles” i „Use default native compilers” dla Linuxa i MacOS; „Visual Studio 15 2017”, „x64” i „Use default native compilers” dla Windowsa,
- zaznacz checkbox przy wartości „GDCM\_BUILD\_SHARED\_LIBS”,
- kliknij przycisk „Finish”,
- następnie kliknij przycisk „Generate”,

## Rozdział 1

## Wstęp

Medyczna diagnostyka obrazowa lub obrazowanie modyfikowane to działy diagnostyki medycznej rożnego rodzaju oddziałujące fizycznie. Obrazowe techniki diagnostyczne w szczególnosci dyżeniu zjawiską są pozytywami i zbiornikiem odrzutów ludzkiego ciała za pomocą rozwiązań modyfikowanych aby uzyskać lepsze rezultaty. W tym samym czasie techniki diagnostyczne rozwiązań modyfikowanych są stosowane w dziedzinie radiologii, radioterapii, radiobiologii, radiometrii, radiorezonansu, tomografii, tomografii SPET oraz tomografii PET. Wybrane techniki są szerzej opisane w sekcji 2.1.

Zajęcia związane z rozwiązań modyfikowanych zapisywane w formacie zdefiniowanym przez producenta. Najczęściej istnieje możliwość zapisu danych dotyczących DICOM (Digital Imaging and Communications in Medicine). Oba odrzutów w pliku danych zapisywane są wszystkie parametry badania takie jak warunki akwizycji, nastawny wzorzec, pozycja pacjenta w uzadzeniu pojazdu, model i producent uzadzienia oraz unikalny identyfikator uzadzenia.

Zapisywanie sekcji administracyjnej pacjenta powinno mieć dane identyfikacyjne, takie jako imię, nazwisko, data urodzenia, wiek pacjenta, data badania i inne dane związane z pacjentem. Zapisywanie sekcji technicznej pacjenta powinno mieć dane dotyczące uzadzienia, takie jak typ modelu, producent uzadzienia, nastawny wzorzec, pozycja pacjenta w uzadzeniu pojazdu, model i producent uzadzienia, oraz unikalny identyfikator uzadzenia.

Uzadzenie jest zapisywane w formacie zdefiniowanym przez producenta. Najczęściej istnieje możliwość zapisu danych dotyczących DICOM (Digital Imaging and Communications in Medicine). Oba odrzutów w pliku danych zapisywane są wszystkie parametry badania takie jak warunki akwizycji, nastawny wzorzec, pozycja pacjenta w uzadzeniu pojazdu, model i producent uzadzienia, oraz unikalny identyfikator uzadzenia.

Do komplikacji są portfelowe biblioteki Qt i GDCM.

## 5.1 Narzędzia potrzenne do komplikacji

- Windows — Visual Studio w wersji 2017 lub nowszej,
  - Linux — Platformę zarządzającą naszą aplikację komendy: make (w wersji 3.10 lub nowszej), g++ (w wersji 8 lub nowszej),
  - MacOS — Xcode w wersji 10 lub nowszej.
- Kod zródłowy został skompilowany za pomocą powiązanych narzędzi. W kodzie programu nie występują żadne elementy odlegające od standardu C++17, więc nie powinno być problemów z użyciem niszczących wsparcia standardu C++17.

## 5.2 Biblioteki potrzenne do komplikacji

- Linux
  - Qt jest dostępną w katalogu standardowym dyskryptujej Linuxa. Dla tego instalacja bibliotek Qt może pomóc w rozwoju aplikacji.
  - Program był testowany na wersji 5.12, z tego powodu zalecamy jąst użycwanie tej wersji.
- Windows i MacOs
  - Qt instalacji Qt najazdy użycie sie na oficjalną stronę biblioteki Qt. W prawym górnym rogu powinno się kliknąć przycisk "Download. Try". Na dolnej kolumnie znajdują się linki do pobrania z repozytorium za pomocą menedżera pakietów.

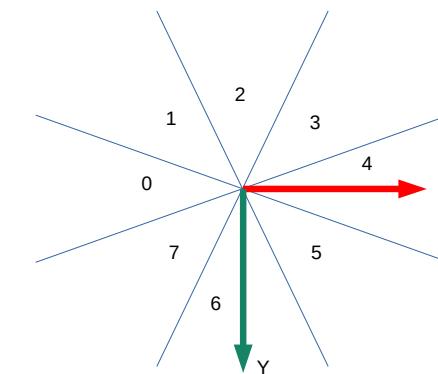
W celu instalacji Qt najazdy użycie sie na oficjalną stronę biblioteki Qt. W prawym górnym rogu powinno się kliknąć przycisk "Download. Try". Na dolnej kolumnie znajdują się linki do pobrania z repozytorium za pomocą menedżera pakietów.

danych obrazowych zawartych w pliku. Głównym aspektem tego procesu jest tak zwane pseudokolorowanie danych numerycznych.

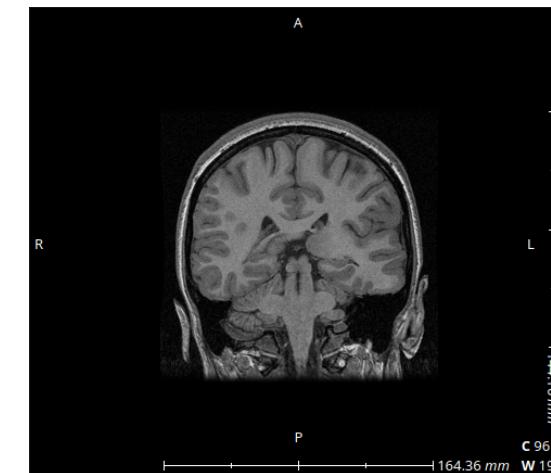
Rozwój obrazowych technik diagnostycznych w medycynie oraz zwiększena dostępność aparatury spowodowała, że badania obrazowe są coraz bardziej powszechnne. Badania obrazowe pomagają lekarzom w diagnostyce i terapii w codziennej praktyce lekarskiej. Przekazywanie badań obrazowych pomiędzy lekarzami różnych specjalności zostało rozwiązane poprzez rozwój standardu DICOM, który przewiduje wymianę danych zarówno poprzez komunikację klient-serwer urządzeń medycznych jak i wymianę plików cyfrowych. Istnieje wiele narzędzi, komercyjnych i otwarto-źródłowych, do wizualizacji i analizy obrazów medycznych. Najczęściej jest to oprogramowanie dedykowane na jedną platformę systemową (system operacyjny). Innym rozwiązaniem jest zastosowanie środowiska, które pozwala na uruchomienie programu na wielu platformach. Takim środowiskiem jest Java firmy Oracle, która umożliwia uruchamianie programów napisanych w języku Java i skompilowanych do „kodu bajtowego” na dowolnej platformie, na której działa maszyna wirtualna Java. Jednakże takie rozwiązanie sprawia, że nie jesteśmy w stanie osiągnąć pełnego potencjału obliczeniowego maszyny przez pewien dodatkowy poziom wirtualizacji.

Celem niniejszej pracy inżynierskiej było opracowanie przeglądarki obrazów medycznych działającej na różnych platformach i zapewniającej szybkość działania, która nie jest ograniczona wirtualizacją kodu. Założono, że cel ten zostanie zrealizowany poprzez opracowanie jednolitego kodu w języku C++ dla wizualizacji i przetwarzania obrazów, komplikowanego do kodu maszynowego na każdą z docelowych platform. Język C++ pozwala uzyskać kod maszynowy, który charakteryzuje się wysoką wydajnością z bezpośrednim dostępem do zasobów sprzętowych i funkcji systemowych. Przyjęto, że do obsługi zagadnień specyficznych dla danego systemu operacyjnego, w tym graficznego interfejsu użytkownika będzie wykorzystana biblioteka Qt. Biblioteka Qt jest wielo-platformowym zestawem narzędzi rozwijania oprogramowania. Zapewnia ona nie tylko obsługę interfejsu użytkownika ale również bogatą bibliotekę programowania aplikacji. Dodatkową zaletą wyboru biblioteki Qt w kontekście obrazowania medycznego jest to, że posiada ona certyfikaty zgodności z normą IEC 62304:2015 ułatwiający wprowadzanie przeglądarki obrazów na rynek Unii Europejskiej jako wyrobu medycznego klasy I z funkcją pomiarową, klasy II lub III.

W opracowanym kodzie przeglądarki obrazów do obsługi plików w formacie DICOM wykorzystano bibliotekę Grassroots (Grassroots DICOM library — GDCM).



Rysunek 4.14: Podział płaszczyzny sceny. Wyróżniono osiem części. Zdjęcie własne.



Rysunek 4.15: Przykładowy obraz medyczny (przekrój głowy MR) z oznaczeniem orientacji obrazu za pomocą liter A, P, R, L, F, H. Zdjęcie własne.

# Rozdział 2

## Obrazowanie diagnostyczne w medycynie



Ostateczne wyznaczenie pozycji punktów Pacjenta na obrąźce odbywa się przy użyciu funkcji `rotateTransform` i `ScenePosition`.

- pozycja z **Image Position** (0x0020, 0x0032) zosłata zapisana do punktu zerowego, - dzięciu temu, wylilk też będzie względem punktu zero. Wyznaczenie macierzy *imGMatrix* jest jednorazowe.

różnym kątem. W tomografii komputerowej podobnie jak w radiografii wykorzystuje się promieniowanie X do pomiaru projekcji (stąd inna nazwa tomografia rentgenowska). W wybranej płaszczyźnie dokonuje się pomiarów projekcji po liniach biegnących pod różnym kątem i w różnych odległościach od badanego obiektu. Przekrój obiektu jest rekonstruowany numerycznie na podstawie zmierzonych projekcji.

Obrazowany jest współczynnik natężenia promieniowania X przez obiekt. Wielkość obrazu może być różna i jest zależna od ustawień tomografu, najczęściej jest to 512 na 512 wokseli. Piksel obrazu jest uzyskiwany podczas rekonstrukcji obrazu i reprezentuje przenikalność promieniowania X. Kontrast i rozdzielcość zależy od tych samych parametrów co w klasycznej radiografii.

W standardzie DICOM technika jest oznaczana skrótwcem „CT”.

- Obrazowanie metodą rezonansu magnetycznego — MRI

Sposób tworzenie obrazu MRI jest wysoce skomplikowanym procesem, którego szczegółowy opis przekracza zakres niniejszego opracowania. Obrazowana jest sumaryczna gęstość atomów wodoru (protonów) w badanym obiekcie. W zależności od sekwencji pobudzeń polem elektromagnetycznym, wyróżniamy trzy typy obrazów: PD, T1 i T2. Kontrast zależy od gęstości protonów, czasu relaksacji podłużnej i poprzecznej, prędkości przepływu płynu. Rozdzielcość zależy od parametrów skanera (rozmiar woksela).

W standardzie DICOM modalność rezonansu magnetycznego jest oznaczana jako „MR”.

- Ultrasonografia

Podczas badania ultrasonograficznego generujemy fale akustyczne o wysokich częstotliwościach, które kierowane są w stronę obiektu, a następnie rejestrowane są fale odbite. Obrazowana jest różnica gęstości poszczególnych warstw znajdujących się w obiekcie.

Zbieranie danych odbywa się przez cyklicznie wysyłanie i odbieranie fal ultradźwiękowej pod różnymi kątami. Z każdego cyklu jest tworzona jedna linia, obraz jest tworzony z wielu linii, które następnie są układane pod różnymi kątami, odpowiadającymi ich rzeczywistemu ułożeniu na głowicy. Wielkość obrazu jest zależna od algorytmu rekonstrukcji i jest z góry ustawniona przez producenta aparatu. Różnice pomiędzy pikselami definiują umowną różnicę gęstości zależną od aparatu. Kontrast zależy od częstotliwości fali, głębokości badanego obiektu, liczby piezoelektryków w głowicy, obrazowanej struktury. Rozdzielcość zależy od czasu trwania impulsu zaburzenia oraz od szerokości wiązki ultradźwiękowej (powierzchnia czynna przetworników).

W standardzie DICOM obraz ultrasonograficzny jest oznaczano jako „US”. Obrazy dopplerowskie „Color flow Doppler(CD)” i „Duplex Doppler(DD)” były kiedyś w standardzie, ale zdecydowano się je wycofać.

- Scyntygrafia

Obrazowa technika diagnostyczna z gałęzi medycyny nuklearnej. Polega na wprowadzeniu do organizmu radiofarmaceutycyku, czyli związkowi chemicznego zawierającego izotop promieniotwórczy. Charakteryzuje się on krótkim czasem rozpadu i powinowactwem chemicznym z badanymi organami. Wykrywa się rozpad zachodzący w ciele

- $\Delta_i$  i  $\Delta_j$  — rzeczywista wielkość piksela obrazu wyrażona w milimetrach. W algorytmie wyznaczania strony pacjenta ta wartość może wynosić 1, ponieważ odpowiada za skale.

Praktycznie rzeczą biorąc, pierwsza macierz to wektor reprezentujący pozycję pacjenta, druga jest to przekształcenie macierzowe we współrzędnych jednorodnych, trzecia to pozycja na obrazie.

### Wyznaczanie pozycji pacjenta

Interesuje nas wyznaczenie pozycji sześciu punktów na płaszczyźnie obrazu. Założymy, że pacjent znajduje się w środku układu współrzędnych i jest nieskończoność mały. Możemy więc zdefiniować sześć punktów o następujących współrzędnych, dalej używanych pod nazwą *PatientPosition*, które będą odpowiadały stronom pacjenta:

- „R” —  $[-1, 0, 0, 1]$ ,
- „L” —  $[+1, 0, 0, 1]$ ,
- „A” —  $[0, -1, 0, 1]$ ,
- „P” —  $[0, +1, 0, 1]$ ,
- „F” —  $[0, 0, -1, 1]$ ,
- „H” —  $[0, 0, +1, 1]$ .

Punkty *PatientPosition* odpowiadają punktom  $P_{xyz}$  z równania ze standardu DICOM.

**UWAGA:** Wszystkie obliczenia odbywają się we współrzędnych jednorodnych.

Na równaniu z poprzedniego punktu wykonuje takie przekształcenie:

$$PatientPosition = imgMatrix * ScenePosition$$

$$imgMatrix^{-1} * PatientPosition = imgMatrix^{-1} * imgMatrix * ScenePosition$$

$$imgMatrix^{-1} * PatientPosition = ScenePosition$$

$$ScenePosition = imgMatrix^{-1} * PatientPosition$$

gdzie:

- *imgMatrix* — macierz przekształcenia obrazu, o której będzie później opisana,
- *ScenePosition* — pozycja na obrazie, która nas interesuje,
- *PatientPosition* — jeden z punktów względem pacjenta.

Wygląd macierzy *imgMatrix*:

$$\begin{bmatrix} X_x & Y_x & 0 & 0 \\ X_y & Y_y & 0 & 0 \\ X_z & Y_z & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Powyższa macierz różni się od macierzy definiowanej w standardzie. Po pierwsze wartości z Tag Pixel Spacing (0x0028, 0x0030) zostały pominięte, a nadano im wartość 1. Po drugie

## 2.2.1 Podstawowe parametry obrazu cyfrowego

Standard DICOM nazwywa techniką obrazowania modalnościową (ang. modality).

- PET-MRI — połączenie PET z rezonansem magnetycznym.

- PET-CT — połączenie PET z wielorzędową tomografią komputerową,

Lstmitieja badamia. Ileczkaćce w sobie rozne techniki, takie jak:

Lecznika orzozwania z gałązka medyczny nuklearnej, w której rejestruje się po- mienionowane powstające podczas amiliacki pozycjony (antydetektorów). Zrobić to można również (pozycjony) jest podana pacjentowi substancja promieniotwórcza, a następnie rozpadowej beta plus. Rejestruje się ją detektorami, o którym mowało się wyżej. Wysyłając do nich sygnały z detektora, odczytuje się informację o tym, ile ją dostrzegły detektory oraz iliczba detektorów.

- #### • Tomographic PET

W standardzie DICOM obraz jest oznaczany jako „PT”.

detectrow.

- ## • Tomografia SPECT

W standardzie DICOM obraz cyfryzacyjny jest ozaczany jako „NM”.

merami, gamma kamerami lub kamerali Angera.

W czasie trwania pomiaru, oraz od aktywności w strefy kliniczego radiofarmaceutycza.

przedstawia się go w formie graficznej.

## 2.2 Parametry obrázku

сии. (8и) низкоинтенсивные излучения шириной в 60 мрадиан распространяются

- PET-MRI — połączenie PET z rezonansem magnetycznym.

- PET-CT — połączenie PET z wielorzędowym tomografem

W standardzie DICOM obraz jest oznaczany jako „PT”.

Detectja odbywa się za pomocą kolimatora, scyntylatora, fotopowielacza i układu

poprzec rejestracji promieniowania wytworzaneego podczas tego rozpadu, a nastepnie

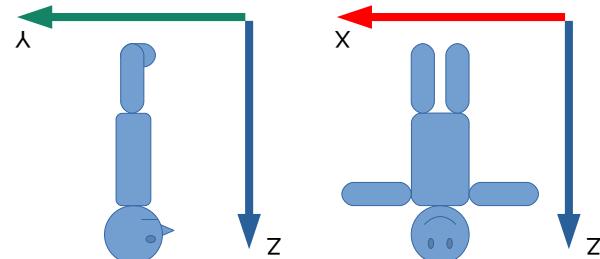
Format zapisu informacji o orientacji obrazu

- $P_{xyz}^{xyz}$  — koordynaty wokseli (i,j) we wsposłudźinych obrazu wyrazione w milimetrach,
- $S_{xyz}^{xyz}$  — trzy wartości wokseli z elementu ze znacznika  $\text{Image Tag}$   $\text{Dicom Position}$  (0x0020, 0x0032),
- $X_{xyz}^{xyz}$  — trzy pierwsze wartości wokseli ze znacznika  $\text{Image Tag}$   $\text{Dicom Orientation}$  (0x0020, 0x0037),
- $Y_{xyz}^{xyz}$  — trzy ostatnie wartości wokseli ze znacznika  $\text{Image Tag}$   $\text{Dicom Orientation}$  (0x0020, 0x0037),
- $Z_{ij}$  — oznaczają wsposłudźihe na macierzy obrazu, odpowiednio kolumnie i wiersz.

$$\begin{bmatrix} 1 \\ 0 \\ \ell \\ i \end{bmatrix} W = \begin{bmatrix} 1 \\ 0 \\ \ell \\ i \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ z_S & 0 & \ell \nabla_X \lambda & \ell \nabla^z X \\ \ell \hat{n}_S & 0 & \ell \nabla \hat{n}_X & \ell \nabla^z \hat{n}_X \\ i x_S & 0 & \ell \nabla^x \lambda & \ell \nabla^x X \end{bmatrix} = \begin{bmatrix} 1 \\ z_d \\ \ell \hat{n}_d \\ i d \end{bmatrix}$$

Wartość **Dicom Image Orientation** (0x0020, 0x0037) składa się z sześciu liczb, odpowiadających pozycji interpretowanych danyach:

Rysunek 4.13: Wizualizacja układu osi współrzędnych kartezjańskich pacjenta. Zdjęcie własne.



- „Y” — os przekształca do przednia do tylni pacjenta, „A” — oznacza zwrot głowy z osią,
- „A” — oznacza zwrot głowy,
- „Y” — os przekształca do przednia do tylni pacjenta, „F” — oznacza zwrot głowy z osią,
- „Z” — os przekształca od dolu do góry pacjenta, „H” — oznacza zwrot głowy z osią,
- „F” — oznacza zwrot głowy.

- „x” — os przełożona do prawej do lewej stroną pągęta, „L” — oznacza zwrot zgody na jąco: Standard DICOM definicja małego obrazu pozycji względem pacjenta zasadniczo sie w odniesieniu do taga Image Orientation (0x0020, 0x0037) i taga Image Position (0x0020, 0x0032).

W dokumencie są wielokrotnie zawarte odniesienia do znaczników DICOM. Dlatego, aby zwiększyć czytelność pracy, została zastosowana konwencja poprzedzania znaczników przedrostkiem  $\text{Dicom}$  składającym się z numeru grupy i elementu grupy zapisanych heksadecymalnie. Przykład poniżej:

$\text{Dicom Tag PatientID} (0x0010, 0x0020)$

Oznacza to, że jest to znacznik o słowie kluczowym „PatientID”, numerze grupy  $10_{16}$  i numerze elementu  $20_{16}$ .

Wyrażenie „informacja ta zawarta w znaczniku ...” będzie oznaczało, że ta informacja znajduje się w elemencie danych o znaczniku.

Dodatkowo w dokumencie PDF można kliknąć na nazwę klasy i użytkownik zostanie przekierowany do strony <https://dicom.innolitics.com/ciods> poprzez wyszukiwarkę DuckDuckGo, na której znajduje się przeglądarka znaczników DICOM.

Każdy obraz cyfrowy jest matrycą pikseli o ustalonych rozmiarach. W przypadku standardu DICOM obrazy są matrycami wokseli, posiadającymi wysokość (zapisaną w  $\text{Dicom Tag Rows}$  ( $0x0028, 0x0010$ )) oraz szerokość (zapisaną w  $\text{Dicom Tag Columns}$  ( $0x0028, 0x0011$ )). Do poprawnej interpretacji znaczenia macierzy służy znacznik  $\text{Dicom Tag Photometric Interpretation}$  ( $0x0028, 0x0004$ ), informujący o fotometrycznym znaczeniu wokseli. Standard DICOM definiuje następujące wartości tego tagu (wraz z wyjaśnieniem):

- „MONOCHROME1” i „MONOCHROME2” — ta wartość woksela odwzorowuje skale monochromatyczną, odpowiednio od jasnego do ciemnego i od ciemnego do jasnego.
- „PALETTE COLOR” — ta wartość woksela jest używana jako indeks w każdej z tabel wyszukiwania kolorów palety czerwonej, niebieskiej i zielonej. Palety mają swoje własne tagi. Wartość raczej rzadka i nie spotykana.
- „RGB” — oznacza, że woksel jest trzy-kanałowym pikselem RGB (kanały: czerwony, zielony i niebieski).
- „HSV” (ang. *Hue Saturation Value*) — woksel reprezentuje piksel w modelu przestrzeni barw zaproponowany w 1978 roku przez Alveya Raya Smitha. Model ten nawiązuje do sposobu w jakim widzi oko człowieka. Wartość wycofana.
- „ARGB” — ta wartość woksela to piksel RGB z dodatkowym kanałem przezroczystości. Wartość wycofana.
- „CMYK” — ten woksel to piksel w modelu czterech podstawowych kolorów farb drukarskich stosowanych powszechnie w druku wielobarwnym w poligrafii: cyjan, magenta, żółty, czarny. Wartość wycofana.
- „YBR\_FULL” — ten woksel to piksel w modelu przestrzeni barw nazwanej YCbCr. Dodatkowo standard zdefiniował pochodne tej wartości: „YBR\_RCT”, „YBR\_FULL\_422”, „YBR\_PARTIAL\_422”, „YBR\_PARTIAL\_420”, „YBR\_ICT”, ale wszystkie są już wycofane.

Wiele elementów danych lub wartości zostały wycofane ze standardu DICOM w wersji 3.0. Oznaczane są jako wycofane (ang. *retired*). Można dalej wspierać ich obsługę w celach wstępnej kompatybilności, ale nie jest to wymagane.

- **Zoom** — stan skalowania, obsługiwany przez *Sokar::DicomScene*

Na macierzy skalowania wywoływana jest funkcja skalowania *Qt::QTransform::scale()* z parametrem *scale* wyliczonym podanym wzorem:

$$\begin{aligned} scale &= 1 \\ scale &= scale - \Delta y * 0.01 \\ scale &= scale - \Delta x * 0.001 \end{aligned}$$

Sprawia to, że ruch pionowy jest bardziej czuły na zmianę niż ruch poziomy. Teoretycznie jest możliwość implementacji odrębnego skalowania w dwóch osiach, jednakże jest to nieintuicyjne.

- **Rotate** — stan rotacji, obsługiwany przez *Sokar::DicomScene*

Na macierzy rotacji wywoływana jest funkcja rotacji *Qt::QTransform::rotate()* z parametrem *rotate* wyliczonym podanym wzorem:

$$\begin{aligned} rotate &= 0 \\ rotate &= rotate + \Delta y * 0.5; \\ rotate &= rotate + \Delta x * 0.1; \end{aligned}$$

Sprawia to, że ruch pionowy jest bardziej czuły na zmianę niż ruch poziomy.

- **Windowing** — stan okienkowania, obsługiwany przez *Sokar::Monochrome::Scene*

Do obiektu okienka są wysyłane zmiany poprzez funkcje: *Sokar::Window::mvVertical()* z parametrem  $\Delta y$  i *Sokar::Window::mvHorizontal()* z parametrem  $\Delta x$ . Następnie ponownie jest generowany obraz z uwzględnieniem zmiany okienka.

## Połączenie macierzy

Ostatnim krokiem jest połączenie macierzy w jedną. Dlatego cztery macierze są mnożone za pomocą wirtualnej funkcji *Sokar::DicomScene::getPixmapTransformation()*. Kod funkcji:

```
1 QTTransform DicomScene::getPixmapTransformation() {
2     QTTransform transform;
3     transform *= centerTransform;
4     transform *= scaleTransform;
5     transform *= rotateTransform;
6     transform *= panTransform;
7     return transform;
8 }
```

*Qt::QTransform* posiada operator mnożenia, dlatego można mnożyć obiekty tej klasy jak liczby. Realizuje to następujące równanie:

$$\text{panTransform} * \text{rotateTransform} * \text{scaleTransform} * \text{centerTransform}$$

### 4.6.4 Ustalanie pozycji pacjenta względem sceny

W obrazie DICOM jest pośrednio zapisana informacja o ułożeniu obrazu względem pacjenta. Celem algorytmu jest określenie jaką pozycję przyjmuje pacjent w stosunku do sceny tak, aby można było wyświetlić tą pozycję na scenie.

Każdy pojazd posiada określony numer identyfikacyjny, który jest zapisany na tabliczce umieszczonej na przodzie pojazdu. Numer ten jest unikalny dla danego pojazdu i służy do identyfikacji pojazdu w systemie zarządzającym. Wszystkie pojazdy są organizowane w klasie pojazdów.

### Czasowa

Rozdziały czasowe przedstawiają kolejno kolejne etapy rozwoju pojazdu. Rozdział czasowy przedstawiający kolejny etap rozwoju pojazdu jest zapisany na tabliczce umieszczonej na przodzie pojazdu. Wszystkie pojazdy są organizowane w klasie pojazdów.

### Przedrena

## 2.2.3 Rozdziały czasowe

gdzie  $I_{max}$  i  $I_{min}$  to najwyższa i najniższa wartość luminancji.

$$\frac{I_{max} - I_{min}}{I_{max} + I_{min}}$$

Jedna z wielu definicji kontrastu jest kontrast Miechelsona wyrażony wzorem:

## 2.2.2 Kontrast

Obran DICOM zawiera wiele informacji o problemie o problemieDICOM, z których większość ma charakter techniczny. Problematiczne pojęcia takie jak kontrast, szacunki dotyczące kontrastu, rozróżnienia i rozpoznawania obrazu.

czyż bez.

- Pixel Representation (0x0028, 0x0103) — informacje o pozycji w zakresie

- High Bit (0x0028, 0x0102) — informacje o zakresie bitów,

- Bits Stored (0x0028, 0x0101) — informacje o zakresie bitów z założonymi warstwami pikseli,

- Bits Allocated (0x0028, 0x0100) — informacje o zakresie bitów dostępu do zapisu.

każdego:

Kwantyzacja obrazu, czyli liczba pozycji w zakresie, na której zmieni-

je zakres sygnału do określonej wartości klawisza, której po wcisnięciu wykonywana jest operacja.

Należy pamiętać, że zakres sygnału jest określony przez klawisz, który po wcisnięciu wykonywany jest określona operacja.

Na przykład przesyłanie wywoływanego przez funkcję `Qt::IconScene::paint()` — stan przesyłania, obslugiwany przez funkcję `Sokar::IconScene::paint()`.

Liczby określające pozycję stawów paska narzędziowe.

Jeżeli jest wątpliwość, która masyzująca w zakresie przyciskiem myszy, to w zakresie jaka funkcja działa?

Wywoływanie funkcji `Qt::GraphicsScene::mousePressEvent()`, Działanie menu obsługujące zarówno odcenę przesztywną klasy dziedzinę po tej klasie. Dodał bowiem funkcja ta dostarcza obiekt `QGraphicsScene::mousePressEvent()`, który menu obsługujące myszy, aby rozszerzała:

`Qt::GraphicsScene::mousePressEvent(QGraphicsScene::mousePressEvent)`

Zmiany poprzez obsługę myszy

Po jakiegokolwiek zmianie macierzy jest wywoływaną funkcją `Sokar::IconScene::updatePe-`

• **ClearRotate** — przyciśnac macierz rotacji do stanu zerowego.

z parametrami  $-1 \ 1$ ,

• **FlipVertical** — na macierzy rotacji zastosować funkcję `Qt::Transform::scale()`

z parametrem  $-1 \ 1$ ,

• **FlipHorizontal** — na macierzy rotacji zastosować funkcję `Qt::Transform::scale()`

z parametrem  $-90^\circ$ ,

• **RotateLeft90** — na macierzy rotacji zastosować funkcję `Qt::Transform::rotate()`

z parametrem  $90^\circ$ ,

• **RotateRight90** — na macierzy rotacji zastosować funkcję `Qt::Transform::rotate()`

z parametrem  $-90^\circ$ ,

• **DrillHolesOutline** — przyciśnac macierz skalę do stanu zerowego,

skale w zakresie od wyiniarowania obrazu i sceny,

• **Fit2Screen** — przyciśnac macierz skalę do stanu zerowego,

listą opisów reakcji na sygnały (stan zerowy macierzy, to stan w którym macierz nie wykonyuje żadnych operacji):

• **ToolBarActivationSlot()**, która jest slotem,

wszystkich sygnałów, które mają implementowane przez funkcję `Sokar::IconScene::`

po otrzymaniu odpowiedniego sygnału jest wykonywana operacja. Odpior

sygnałów jest zapisany w sekci 4.5.4.

Należy pamiętać, że zakres sygnału, który po wcisnięciu wykonywany jest określonej operacją, to stan w którym macierz nie

z parametrem  $0 \ 1$ ,

z parametrem  $1 \ 0$ ,

z parametrem  $0 \ 0$ ,

z parametrem  $1 \ 1$ ,

z parametrem  $-1 \ -1$ ,

z parametrem  $-90^\circ$ ,

z parametrem  $-180^\circ$ ,

z parametrem  $-270^\circ$ ,

z parametrem  $-360^\circ$ ,

z parametrem  $-45^\circ$ ,

z parametrem  $-135^\circ$ ,

z parametrem  $-225^\circ$ ,

z parametrem  $-315^\circ$ ,

z parametrem  $-405^\circ$ ,

z parametrem  $-495^\circ$ ,

z parametrem  $-585^\circ$ ,

z parametrem  $-675^\circ$ ,

z parametrem  $-765^\circ$ ,

z parametrem  $-855^\circ$ ,

z parametrem  $-945^\circ$ ,

z parametrem  $-1035^\circ$ ,

z parametrem  $-1125^\circ$ ,

z parametrem  $-1215^\circ$ ,

z parametrem  $-1305^\circ$ ,

z parametrem  $-1395^\circ$ ,

z parametrem  $-1485^\circ$ ,

z parametrem  $-1575^\circ$ ,

z parametrem  $-1665^\circ$ ,

z parametrem  $-1755^\circ$ ,

z parametrem  $-1845^\circ$ ,

z parametrem  $-1935^\circ$ ,

z parametrem  $-2025^\circ$ ,

z parametrem  $-2115^\circ$ ,

z parametrem  $-2205^\circ$ ,

z parametrem  $-2295^\circ$ ,

z parametrem  $-2385^\circ$ ,

z parametrem  $-2475^\circ$ ,

z parametrem  $-2565^\circ$ ,

z parametrem  $-2655^\circ$ ,

z parametrem  $-2745^\circ$ ,

z parametrem  $-2835^\circ$ ,

z parametrem  $-2925^\circ$ ,

z parametrem  $-3015^\circ$ ,

z parametrem  $-3105^\circ$ ,

z parametrem  $-3195^\circ$ ,

z parametrem  $-3285^\circ$ ,

z parametrem  $-3375^\circ$ ,

z parametrem  $-3465^\circ$ ,

z parametrem  $-3555^\circ$ ,

z parametrem  $-3645^\circ$ ,

z parametrem  $-3735^\circ$ ,

z parametrem  $-3825^\circ$ ,

z parametrem  $-3915^\circ$ ,

z parametrem  $-4005^\circ$ ,

z parametrem  $-4095^\circ$ ,

z parametrem  $-4185^\circ$ ,

z parametrem  $-4275^\circ$ ,

z parametrem  $-4365^\circ$ ,

z parametrem  $-4455^\circ$ ,

z parametrem  $-4545^\circ$ ,

z parametrem  $-4635^\circ$ ,

z parametrem  $-4725^\circ$ ,

z parametrem  $-4815^\circ$ ,

z parametrem  $-4905^\circ$ ,

z parametrem  $-4995^\circ$ ,

z parametrem  $-5085^\circ$ ,

z parametrem  $-5175^\circ$ ,

z parametrem  $-5265^\circ$ ,

z parametrem  $-5355^\circ$ ,

z parametrem  $-5445^\circ$ ,

z parametrem  $-5535^\circ$ ,

z parametrem  $-5625^\circ$ ,

z parametrem  $-5715^\circ$ ,

z parametrem  $-5805^\circ$ ,

z parametrem  $-5895^\circ$ ,

z parametrem  $-5985^\circ$ ,

z parametrem  $-6075^\circ$ ,

z parametrem  $-6165^\circ$ ,

z parametrem  $-6255^\circ$ ,

z parametrem  $-6345^\circ$ ,

z parametrem  $-6435^\circ$ ,

z parametrem  $-6525^\circ$ ,

z parametrem  $-6615^\circ$ ,

z parametrem  $-6705^\circ$ ,

z parametrem  $-6795^\circ$ ,

z parametrem  $-6885^\circ$ ,

z parametrem  $-6975^\circ$ ,

z parametrem  $-7065^\circ$ ,

z parametrem  $-7155^\circ$ ,

z parametrem  $-7245^\circ$ ,

z parametrem  $-7335^\circ$ ,

z parametrem  $-7425^\circ$ ,

z parametrem  $-7515^\circ$ ,

z parametrem  $-7605^\circ$ ,

z parametrem  $-7695^\circ$ ,

z parametrem  $-7785^\circ$ ,

z parametrem  $-7875^\circ$ ,

z parametrem  $-7965^\circ$ ,

z parametrem  $-8055^\circ$ ,

z parametrem  $-8145^\circ$ ,

z parametrem  $-8235^\circ$ ,

z parametrem  $-8325^\circ$ ,

z parametrem  $-8415^\circ$ ,

z parametrem  $-8505^\circ$ ,

z parametrem  $-8595^\circ$ ,

z parametrem  $-8685^\circ$ ,

z parametrem  $-8775^\circ$ ,

z parametrem  $-8865^\circ$ ,

z parametrem  $-8955^\circ$ ,

z parametrem  $-9045^\circ$ ,

z parametrem  $-9135^\circ$ ,

z parametrem  $-9225^\circ$ ,

z parametrem  $-9315^\circ$ ,

z parametrem  $-9405^\circ$ ,

z parametrem  $-9495^\circ$ ,

z parametrem  $-9585^\circ$ ,

z parametrem  $-9675^\circ$ ,

z parametrem  $-9765^\circ$ ,

z parametrem  $-9855^\circ$ ,

z parametrem  $-9945^\circ$ ,

z parametrem  $-10035^\circ$ ,

z parametrem  $-10125^\circ$ ,

z parametrem  $-10215^\circ$ ,

z parametrem  $-10305^\circ$ ,

z parametrem  $-10395^\circ$ ,

z parametrem  $-10485^\circ$ ,

z parametrem  $-10575^\circ$ ,

z parametrem  $-10665^\circ$ ,

z parametrem  $-10755^\circ$ ,

z parametrem  $-10845^\circ$ ,

z parametrem  $-10935^\circ$ ,

z parametrem  $-11025^\circ$ ,

z parametrem  $-11115^\circ$ ,

z parametrem  $-11205^\circ$ ,

z parametrem  $-11295^\circ$ ,

z parametrem  $-11385^\circ$ ,

z parametrem  $-11475^\circ$ ,

z parametrem  $-11565^\circ$ ,

z parametrem  $-11655^\circ$ ,

z parametrem  $-11745^\circ$ ,

z parametrem  $-11835^\circ$ ,

z parametrem  $-11925^\circ$ ,

z parametrem  $-12015^\circ$ ,

z parametrem  $-12105^\circ$ ,

z parametrem  $-12195^\circ$ ,

z parametrem  $-12285^\circ$ ,

z parametrem  $-12375^\circ$ ,

z parametrem  $-12465^\circ$ ,

z parametrem  $-12555^\circ$ ,

z parametrem  $-12645^\circ$ ,

z parametrem  $-12735^\circ$ ,

z parametrem  $-12825^\circ$ ,

z parametrem  $-12915^\circ$ ,

z parametrem  $-13005^\circ$ ,

z parametrem  $-13095^\circ$ ,

z parametrem  $-13185^\circ$ ,

z parametrem  $-13275^\circ$ ,

z parametrem  $-13365^\circ$ ,

z parametrem  $-13455^\circ$ ,

z parametrem  $-13545^\circ$ ,

z parametrem  $-13635^\circ$ ,

z parametrem  $-13725^\circ$ ,

## 2.2.4 Stosunek sygnału do szumu (SNR)

Rodzaj i poziom szumu zależy od techniki obrazowania. Stosunek sygnału do szumu ma decydujący wpływ na widoczność obiektów, kontrast oraz percepcję szczegółów w obrazie.

## 2.2.5 Poziom artefaktów

Artefakty są zjawiska falszujące obraz poprzez tworzenie struktur w obrazie, nieistniejących w rzeczywistości. Jest to problem występujący w różnych technikach obrazowania. Najbardziej znanymi artefaktami są np. w badaniu USG tak zwany warkocz komety w przypadku obiektów o wysokiej różnicy impedancji w stosunku do otoczenia.

## 2.2.6 Poziom zniekształceń przestrzennych

Zniekształcenia przestrzenne powstają w wyniku geometrycznego ułożenia i kształtu obiektu badanego oraz aparatu pomiarowego. Przykładem takiego zniekształcenia mogą być różne powiększenia obiektów zależne od głębokości ich ułożenia w USG, zmiana pozycji pacjenta (przez ruchy klatki piersiowej w czasie badania), czy deformacja obrazu spowodowana zmianami rozkładu pola magnetycznego przez metalowe obiekty w znajdujące się w tym samym pomieszczeniu w przypadku badań MRI.

## 2.3 Prezentacja obrazów medycznych

W celu przeglądania i porównywania należy posiadać narzędzie do wyświetlenia w sposób poprawny, najlepiej jednym i tym samym programem.

### 2.3.1 Przeglądarki obrazów

Przeglądarki obrazów to programy należące do kategorii przeglądarki plików. Zwykle przeglądarki obrazów takich jak jpg, png lub gif wyświetlają obraz w takiej postaci jakiej jest zapisany, najpierw przeprowadzając dekompresję tego obrazu. W przypadku obrazów medycznych najczęściej nie mamy do czynienia z danymi reprezentującymi kolory w spektrum światła widzialnego. Przeglądarka obrazów DICOM musi wygenerować kolorowy obraz z danych na podstawie parametrów obrazu.

### 2.3.2 Funkcje przeglądarki obrazów

#### Obsługa wielu formatów danych

W standardzie DICOM przewidziano możliwość zapisania wielu typów danych w różnych formatach, nie tylko obrazów, ale też nagrań nagrań audio i tekstów. Przeglądarka obrazów DICOM może mieć możliwość odczytania, wyświetlenia lub odsłuchania danych.

#### Podstawowe operacje na obrazie

- Skalowanie lub powiększenie, czyli możliwość powiększenia lub zmniejszenia wyświetlanego obrazu o pewien współczynnik skalujący.

## Palety

Klasa *Sokar::Palette* reprezentuje palety kolorów używanych do pseudokolorowania obrazu. Paleta przerabia liczbę zmiennoprzecinkową od zera do jedynki na kolor RGB, zwracając *Sokar::Pixel*. Paleta nie ma zdefiniowanej długości minimalnej i maksymalnej.

Palety są wczytywane z plików XML w czasie uruchamiania programu i można definiować własne palety nie będące częścią standardu. Przykładowy wygląd pliku palety HotIron:

```
1 <palette display="Hot_Iron" name="HOT_IRON">
2
3   <entry red="0" green="0" blue="0"/>
4   <entry red="2" green="0" blue="0"/>
5   <entry red="4" green="0" blue="0"/>
6
7   ...
8
9   <entry red="254" green="0" blue="0"/>
10  <entry red="255" green="0" blue="0"/>
11  <entry red="255" green="2" blue="0"/>
12
13  ...
14
15  <entry red="255" green="250" blue="248"/>
16  <entry red="255" green="252" blue="252"/>
17  <entry red="255" green="255" blue="255"/>
18 </palette>
```

### 4.6.3 Tworzenie transformat i ich użycie na obrazie

#### Współrzędne jednorodne

Współrzędne jednorodne definiuje się jako sposób reprezentacji punktów n-wymiarowej przestrzeni rzutowej za pomocą układu  $n + 1$  współrzędnych. W bibliotece Qt jedną z implementacji współrzędnych jednorodnych jest klasa *Qt::QTransform*. Implementuje ona podstawowe zachowanie macierzy 3 na 3, jak również wbudowane operacje takie jak: przesuwanie implementowane przez *Qt::QTransform::translate()*, obrót implementowany przez funkcję *Qt::QTransform::rotate()* i skalowanie implementowane przez *Qt::QTransform::scale()*.

Przykład działania:

```
1 QTransform transform;
2 transform.translate(50, 50);
3 transform.rotate(45);
4 transform.scale(0.5, 1.0);
```

Powyższe przekształcenie macierzowe skaluje obiekt na 50% szerokości, obraca o 45 stopni, przesuwa o 50 punktów na osi *x* i *y*.

Taką macierz można nałożyć na obiekty klasy *Qt::QGraphicsPixmapItem*.

#### Interakcja z użytkownikiem

Trzy macierze (bez środkowującej) są zmieniane w trakcie interakcji z użytkownikiem. Są zmieniane w dwóch przypadkach: po odebraniu sygnału od paska zadań, obiektu klasy *Sokar::DicomToolbar* lub podczas ruchu myszki, gdy wciśnięty jest prawy przycisk myszy.

• Wczytanie wileń pliku w taki samie sezon, ułożenia ich według pozycji geometrycznej i wyświetlenia

zbiór elementów darych, bez obrzawy.

• Obsługa DICOMDIR. Jest to rozszerzenie wczytania pliku DICOMDIR i wyświetlenie struktury sezonu bieżącego. Plik DICOMDIR to wiele zindeksowanej plikuDICOMDIR zawierającej

### Odsługa wileń pliku

jeden obrázek.

szczególnych obiektów, np. tła. Standard DICOM umozliwia ułożenie wielu mask na przystanek fragment obrazu w celu lepszej wizualizacji bieżącej mafii warstwy

• Maska (ang. overlay). Jest to rozszerzenie nazwana maską, elementu, który będzie

tycznego.

jest szczególnego opisane w sekcji 4.6.2 wraz z generowaniem obrazu monochromatycznego do wyświetlenia. Odsłonowanie obrazu dany na obrázku monochromatycznego pozwalać

• Odsłonowanie. Termin odnosi się do uzyskania funkcji okna wyświetlonego w celu zaznaczyć obrázku dany na obrázku monochromatycznego pozwalać

### Anotacja parametrow w celu lepszej informacji

możliwości użyczenia odlicia i ustalenie odrzaru w dwuściach X i Y.

• Rotacja i odcięcia iustrzane, czyli możliwość obliczenia obrazu o zadany kąt oraz

zgodą Software UAB.

Rysunek 2.1: Narzędzie Lupa w programie MedDream DICOM Viewer. Zdjęcie uzyte za



Przykład użycia takiego narzędzia znajduje się na rysunku 2.1.

• Lupa, służące mierzenie. Jest to rozszerzenie mierzenie powiększenia obrazu

na ekranie lub w okienku programu.

to przydatne, gdy powiększyć obrzą do takiego stopnia, że nie będzie mieszczyć się w tym okienku.

• Przesuwanie (ang. pan), czyli możliwość przesuwania obrzą o dowolny wektor. Jest

```

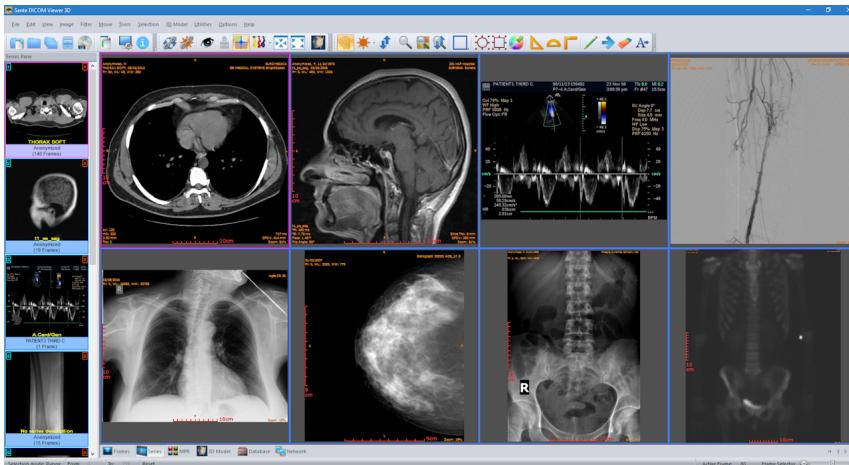
1 template<typename IntType>
2 void Monochrome::Scene::generatePixmap()
3 {
4     Monochrome::Scene::generatePixmap();
5     switch (getCurturementWindow() ->type) {
6         case Window::IntStatic:
7             genQPixmapOfTypeWindow<IntType, WindowIntDynamic>();
8         break;
9         case Window::IntStatc:
10            genQPixmapOfTypeWindow<IntType, WindowIntStatc>();
11        break;
12        default:
13            throw WrongScopeException<"FILE", "LINE"!>;
14    }
15 }
16
17 void Monochrome::Scene::generatePixmap()
18 {
19     Monochrome::Scene::generatePixmap();
20     switch (getLastXmapChange() ->genLT) {
21         case Window::Monochrome:
22             genQPixmapOfTypeWindow<IntType>();
23         break;
24         case Window::IntFormat:
25             genQPixmapOfTypeWindow<IntType>();
26         break;
27         case Window::PixelFormat:
28             genQPixmapOfTypeWindow<IntType>();
29         break;
30         case Window::PixelFormat:
31             genQPixmapOfTypeWindow<IntType>();
32         break;
33         default:
34             throw SoKakr::InnychJestZwacanyWyjatek();
35     }
36 }
37
38 pixmap::convertFromImage(qImage);
39
40 return true;

```

ich jako film. Innymi słowy jest periodyczna podmiana obrazu na obraz następny w serii.

- Wyświetlanie wielu obrazów jednocześnie. Jest to możliwość wyświetlania kilku obrazów w postaci tabelki, w której każda komórka była by innym obrazem.

Przykład wyświetlania wielu obrazów na raz w jednym oknie znajduje się na rysunku 2.2



**Rysunek 2.2:** Wyświetlenie wielu obrazów na raz w jednym oknie w przeglądarce Sante DICOM Viewer 3D Pro. Zdjęcie użyte za zgodą Santesoft.

### Generowanie obrazów wolumetycznych

Jeżeli mamy do dyspozycji wiele obrazów tomograficznych o znanych parametrach to możemy wczytać je, poszgregować a następnie wygenerować trójwymiarowy obiekt, który wyświetlany jest ekranie komputera za pomocą trójwymiarowej grafiki komputerowej.

Przykład takiego obrazu znajduje się na rysunku 2.3.

### Analiza i przetwarzanie danych

- Histogram, czyli możliwość wygenerowania histogramu obrazu.

Histogram to wykres przedstawiający dystrybucję wartości numerycznych obrazu.

- Mierzenie i wykonywanie pomiarów. Pozwala na określenie odległości pomiędzy dwoma punktami przez lekarza lub zmierzenie wielkości/pola zadanego kształtu.

- Rekonstrukcja wielopłaszczyznowa. Obrazy tomograficzne przedstawiają przekroje. Jeżeli parametry wielkości woksela są dostępne to istnieje możliwość wygenerowania nowego obrazu, który byłby przekrojem poprzecznym.

Przykład generowania rekonstrukcji wielopłaszczyznowej jest pokazany na rysunku 2.4

### Iterowanie po obrazie

Po wygenerowaniu obrazu należy przeiterować go przez funkcję „okna”. Do zokienkowania jednego piksela nie potrzeba innego piksela, dlatego w celu przyspieszenia procesu okienkowania iteracja po obrazie odbywa się w wielu wątkach.

W C++ typy zmiennych muszą być zdefiniowane przed komplikacją, co jest bardzo problematyczne. Mając dwa typy okienek, każde odsługujące 4 typy liczb całkowitych, musiałoby zostać zaimplementowanych 8 prawie identycznych funkcji. Dlatego podział ten został zaimplementowany za pomocą 4 funkcji z szablonami:

- *Sokar::Monochrome::Scene::genQPixmapOfTypeWidthWindowThread()*

Jest funkcją jednego wątku, który iteruje po obrazie. Jego parametrami są zakresy podane w indeksach wokseli po których ma iterować. *IntType* to jest typ zmiennej wokselu obrazu. *WinClass* klasa okienka. Nazewnictwo będzie kontynuowane w następnych punktach. Kod funkcji:

```
1 template<typename IntType, typename WinClass>
2 void Monochrome::Scene::genQPixmapOfTypeWidthWindowThread(quint64 from, quint64 to)
3 {
4     auto buffer = &targetBuffer[from];
5     auto origin = (IntType *) &originBuffer[0];
6     auto windowPtr = (WinClass *) getCurrentWindow();
7
8     origin += from;
9
10    for (quint64 i = from; i < to; i++, origin++) {
11        /* W tym miejscu jest dokonywana zamiana liczby na kolor */
12        *buffer++ = windowPtr->getPixel(*origin);
13    }
14 }
```

- *Sokar::Monochrome::Scene::genQPixmapOfTypeWidthWindow()*

Jest to funkcja, która dzieli obraz na wątki, tworzy je i uruchamia. Ilość wątków jest ustalana za pomocą funkcji *Qt::QThread::idealThreadCount()*. Wątki działają na zakresach o długości wokseli podzielonej przez ilość wątków. Kod funkcji:

```
1 template<typename IntType, typename WinClass>
2 void Monochrome::Scene::genQPixmapOfTypeWidthWindow() {
3
4     /* Tworzenie wektora wątków */
5     std::vector<std::thread> threads;
6
7     quint64 max = imgDimX * imgDimY;
8     quint64 step = max / QThread::idealThreadCount();
9
10    for (int i = 1; i < QThread::idealThreadCount(); i++) {
11        std::thread t(&Scene::genQPixmapOfTypeWidthWindowThread<IntType, WinClass>,
12                      this,
13                      i * step,
14                      std::min((i + 1) * step, max));
15
16        threads.push_back(std::move(t));
17    }
18
19    /* W celu zmniejszenia ilości wątków, wątek obecny też zostanie wykorzystany */
20    genQPixmapOfTypeWidthWindowThread<IntType, WinClass>(0, step);
21
22    /* Czekanie na wszystkie wątki */
23    for (auto &t: threads) t.join();
24 }
```

- *Sokar::Monochrome::Scene::genQPixmapOfType()*



## Edycja danych

- Dodawanie nowych obiektów. Pozwala na rysowanie, dodawanie figur geometrycznych lub tekstu przez lekarza i zapis tych informacji w pliku DICOM. Chodzi tu głównie o szkice i notatki tworzone podczas analizy obrazu przez personel medyczny.
- Edycja parametrów oraz anonimizacja danych. Jest to możliwość edycji parametrów w pliku DICOM w różnych celach. Funkcja jest używana do usuwania danych osobowych pacjenta w celu późniejszej publikacji obrazu.

### 2.3.3 Kryteria porównywania przeglądarek obrazów

Porównanie aplikacji posiadających tak wiele parametrów jak przeglądarki DICOM jest bardzo skomplikowanym procesem. Dlatego wyróżniono 26 kryteriów do ich porównywania w postaci logicznej: „tak” lub „nie”, podzielonych na 5 grup, platformy, interfejsu, wsparcia, obrazowania dwu i trójwymiarowego [1]. Kryteria te w jasny sposób pozwalają na ocenę praktycznych aspektów użytkowania przeglądarki.

## Platforma

Grupa platforma zawiera kryterium samodzielności. Aplikacje samodzielne są zaprojektowane tak, aby nie wymagały żadnego dodatkowego sprzętu fizycznego bądź infrastruktury do poprawnego działania. Rozwiązań sieciowych określają, czy aplikacja jest usługą sieciową i czy można z aplikacji korzystać jak ze strony WWW. Aplikacje są wieloplatformowe, czyli mają możliwość uruchomienia ich na różnych systemach operacyjnych Linux/MacOS/Windows oraz możliwość używania ich na urządzeniach mobilnych takich jak telefon.

## Interfejs

Przeglądarka powinna mieć możliwość komunikacji z interfejsami innych systemów. Podstawowe interfejsy sieciowe to: C-STORE SCP DICOM C-STORE, C-STORE SCU, Query-Retrieve, WADO, Parameter Transfer.

## Wsparcie techniczne

Aplikacja powinna mieć dostępną pisemną dokumentację oprogramowania (np. podręczniki lub strony internetowe), wsparcie przez pocztę internetową, możliwość porozumienia się z twórcą lub opiekunem oprogramowania, forum (możliwość pytania się społeczności o opinie i ich wymiana) oraz rodzaj wikipedii (strona internetowa w formacie Wikipedii dostępna dla użytkownika).

## Obrazowanie dwuwymiarowe

Przewijanie (ang. *scroll*), jako forma procesu wyświetlanego obrazów, można poprawić dzięki zmniejszeniu interakcji z klawiaturą oraz myszką. Można to osiągnąć na przykład oferując możliwość przejścia do następnego lub poprzedniego obrazu przez przesunięcie kółkiem myszy lub używając przycisków góra/dół na klawiaturze. Wyświetlane elementy powinny obejmować analizowanie i wyświetlanie elementów danych DICOM, wyświetlanie rozdzielczości obrazu, badanie (np. identyfikator podmiotu) oraz znaczniki DICOM specyficzne dla dostawcy (np. specjalne ustawienie urządzenia rejestrującego). Najważniejsze

Przeglądarka pozwala na inwersje okienka. Dlatego kiedy użytkownik zażyczy sobie inwersji, zmienne  $y_0$  i  $y_1$  zamienią się wartościami.

Standard DICOM przewiduje, że wszystkie dane powinny być wyskalowane za pomocą wzoru:

$$OutputUnits = m * SV + b$$

gdzie:

- $m$  — wartość z  $\frac{\text{Dicom Tag}}{\text{RescaleSlope}}$  (0x0028, 0x1053),
- $b$  — wartość z  $\frac{\text{Dicom Tag}}{\text{RescaleIntercept}}$  (0x0028, 0x1052),
- $SV$  — stored values — wartość wokselu z pliku,
- $OutputUnits$  — wartość wynikowa.

Wartości okienka odnoszą się do wartości już wyskalowanej, a ponieważ skalowanie całego obrazu jest czasochronne, przeskalowanie okienka da taki sam efekt:

$$(OutputUnits - b)/m = SV$$

więc:

$$\begin{aligned}x_0 &= \text{rescaleIntercept} \\x_1 &= \text{rescaleIntercept} \\x_0 / &= \text{rescaleSlope} \\x_1 / &= \text{rescaleSlope}\end{aligned}$$

Posiadamy teraz dwa punkty okienka odnoszące się do wartości obrazu. Wyznaczono parametry prostej przechodzącej przez dwa punkty:

$$a = (y_1 - y_0)/(x_1 - x_0)$$

$$b = y_1 - a * x_1$$

Teraz algorytm się rozdwaja. Pobieranie wartości z okienka odbywa się za pomocą funkcji *Sokar::MonochromeWindow::getPixel()*.

## Implementacja dynamiczna bez tablicy LUT

W tej wersji funkcja *Sokar::Monochrome::Window::getPixel()* wygląda następująco:

```
1 inline const Pixel &getPixel(quint64 value) override {
2     if (value < x0) {
3         return background;
4     } else if (value > x1) {
5         return foreground;
6     } else {
7         return palette->getPixel(a * value + b);
8     }
9 }
```

Widzimy tutaj, że funkcja najpierw sprawdza czy zakres okienka został przekroczyony, następnie wylicza wartość obrazu i pobiera kolor z palety.

UWAGA: ponieważ nie istnieją rzeczywiste obrazy o wokselu 32-bitowym lub 64-bitowym, implementacja dynamiczna nie była testowana w warunkach rzeczywistych.

Standard DCOM jest opowiadziek specjalizujący się w tworzeniu rozwiązań systemów komputerowych, który z kolei medycznych na potrzeby wymiany danych pomiędzy aplikacjami. DCOM pozwala na tworzenie rozwiązań, które wykorzystują obrazów, stacji do przetwarzania i analizowania obrazów medycznych.

2.4.1 Standard DICOM v3.0

Pierwsze tomografy komputerowe przedstawiały swoje rozkwięt w latech siedemdziesiątych ubiegłego wieku. Oraz myśląc o tym, że technologia rozwoju się nieustannie, zdecydowanie wykorzystać możliwości obrotowej darymki pomiarowej przesz komputer. Wyczajne piki grawitacyjne (jak np. jpg, gif, bmp, gif), nie nadawały się do zapisu takich obrazów, ponieważ zapisywali obraz w postaci sładkawego słodka. Kazdy producent stosował

#### 2.4 Format cyfrowych obrazów medycznych

Pięczętnikowa zasada opartej rekonsztynacji wielopłaszczowniów. Wykroje dane doyczane do gromadzonej wzdłuż jednostki celowej. Wśród jednostek medycznych jest promieniowanie i strzalkowymach (np. poprzecznym). Wśród jednostek medycznych jest promieniowanie strzałkowe i strzałkowymach, aby poprawić wizualizację niekotorych struktur. W tym celu należy zapewnić unikalne rekonsztynki osią pomocniczych na podstawie kierunku pierwotnego. Główne zadanie rekonsztynki jest przegladanie daryczej w linii kierunku pierwotnego. Główne zadanie rekonsztynki jest przegladanie daryczej w linii kierunku pierwotnego. Główne zadanie rekonsztynki jest przegladanie daryczej w linii kierunku pierwotnego. Główne zadanie rekonsztynki jest przegladanie daryczej w linii kierunku pierwotnego.

## Obrzowanie trojwy miarowe

Informacjy powinny byc mizuznione w oknie wyswietlacza jefo nakkadka na obraz, np. aktualna pozycja lub nazwa podmiotu wykonyujacego badanie. Okienkoowane jest to sposob zamiany danych na skale zarościi, okienkoowane jest opisane w sekci 4.6.2. Poszukajcie wizualizacji wstępienia i rozkład warstoci kolorów na obrazach, powalać się na istotne cechy organizu. Przeglądajka powinna mieć mizuznosc wyswietlajacą opisywane kolorów, mianych kształtow, mizuznosc sałazy i wykazanego dlegetosci w jednostkach dzugosci aż do rozmiaru. Jest to możliwe, gdyż dzugowita plik DICOM zawierała juz parametry spłetowe mizuzdzenia (np. lose pikseli na milimetr). Aduataje (opisy), które byly wtywzone przez personal medyczny powinny byc zapisane w odpowiedni sposob w pliku.

- *x1 i y1* — współrzędne drutiego punktu,
- *x0 i y0* — współrzędne pierwszego punktu,
- *width* — szerokość okienka,
- *center* — środek okienka,

$$0.1 = 0.0$$

0.0 =  $\text{if}$

$$c_1 = center + width/2$$

$$x_0 = center - width/2$$

Najpierw wyznaczane jest okienko, które zmienia wartości obranych na skale od zero do

#### Wyznaczenie parametrow okna

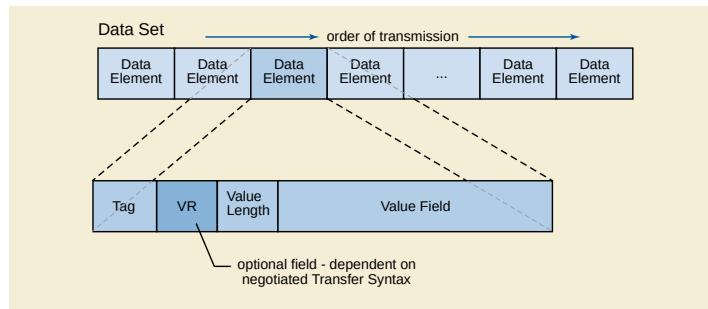
UWAGA: Standard DICOM zakłada, że dany mili moge być liczyty całkowite (int) oraz zmiennoprzecinkowe (float lub double), ale Praktyczne, nie ma takich apartów medycznych, które zapisywają takie obrazy, gdzie dane to liczby zmiennoprzecinkowe. Dlatego w pracy zazwyczaj, że taki obrazu nie będzie odbitym przekształceniem.

Implementacija algoritma

UWAGA: Za każdym razem kiedy jest odniesienie do obecnego standardu DICOM, w domyśle jest to odsłona numer 2019a.

## 2.4.2 Sposób zapisu danych w pliku DICOM

Plik w formacie DICOM przypomina zbiór elementów danych z rekordami. Zbiór nazywa się „Data Set” i składa się z rekordów, które nazywają się „Data Element”. Elementy danych są ułożone w postaci listy. Element danych może zawierać w sobie listę elementów danych.



Rysunek 2.5: Elementy danych w zbiorze elementów danych. Zdjęcie ze standardu DICOM dostępne pod adresem [http://dicom.nema.org/medical/dicom/2019a/output/chtml/part05/chapter\\_7.html](http://dicom.nema.org/medical/dicom/2019a/output/chtml/part05/chapter_7.html).

### Element danych

Element danych, zwany przez standard DICOM „Data Element” jest rekordem, który przechowuje pojedynczą informację o obiekcie. Składa się z czterem elementów:

- „Tag” — to unikalny identyfikator, dalej zwany znacznikiem, jest złożony z dwóch liczb: numer grupy (`uint16`) i numer elementu (`uint16`) grupy. Informuje o tym co dany rekord w sobie zawiera. W jednym zbiorze elementów nie mogą się pojawić dwa elementy posiadających ten sam znacznik.

Na przykład: jeżeli liczby znaczniku przyjmą wartości odpowiednio wartość  $0010_{16}$  i  $0010_{16}$  to oznacza, że jest to znacznik <sup>Dicom</sup><sub>Tag</sub> PatientName (0x0010, 0x0010), czyli zawiera w sobie parametr zawierający nazwę pacjenta.

Dokładne omówienie znaczników znajduje się w sekcji 2.4.2.

- „Value Representation”, w skrócie „VR” – to dwa bajty w postaci tekstu, informujące o formacie w jakim parametr został zapisany.

Dokładne omówienie „VR”-ów znajduje się w dalszej części sekcji.

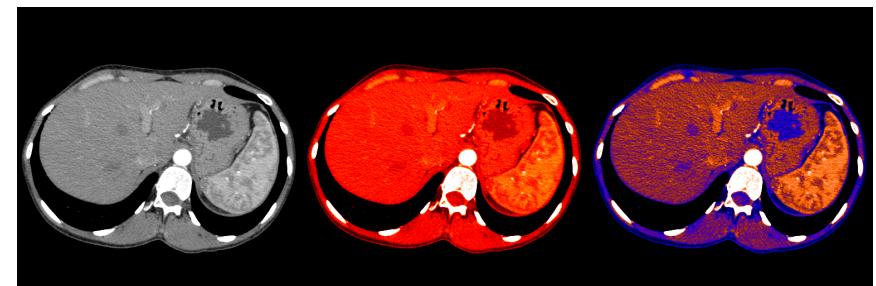
- „Value Length”, w skrócie „VL” — 32-bitowa lub 16-bitowa liczba nieoznaczona, która informuje o długości pola danych („Value Field”).

Wartość „VL” zwykle jest liczbą parzystą. Standard DICOM zakłada, że wszystkie dane powinny być dopełniane do parzystej liczby bajtów.

- „Value Field” (opcjonalne) — pole z parametrem o długości VL.



Rysunek 4.11: Porównanie jednego obrazu w trzech „oknach”, odpowiednio C40 W80, C50 W350 i C600 W2800. Zdjęcie własne.



Rysunek 4.12: Paleta Hot Iron (na środku) i Hot Metal Blue (po prawej) w porównaniu do palety w skali szarości (po lewej). Zdjęcie własne.



- AS — Age String — wiek lub długość życia

Długość danych wynosi 4 bajty. Pierwsze trzy bajty to liczba całkowita zapisana za pomocą tekstu. Czwarty bajt to znak określający jednostkę czasu. Standard definiuje cztery możliwe jednostki czasu: „D” jako dzień, „W” jako tydzień, „M” jako miesiąc, oraz „Y” jako jeden rok.

Przykład: „018M” oznacza 18 miesięcy, „123D” oznacza 123 dni.

- AT — Attribute Tag — inny znacznik

Długość danych to zawsze 32 bity, są to dwie 16-bitowe liczby, odpowiednio grupa i element grupy. Ten VR jest używany kiedy wskazujemy na inny znacznik. Wartość nie jest nigdy pokazywana użytkownikowi, a jedynie używana w interpretacji przez inne algorytmy do analizy obrazu.

Przykład: znacznik  $\frac{\text{Dicom}}{\text{Tag}}$  FrameIncrementPointer (0x0028, 0x0009) jest używany kiedy w pliku jest zapisana sekwencja kilku obrazów. Wskazuje on na inny znacznik zawierający informacje, w jaki sposób ta sekwencja ma być wyświetlona.

- DA — Date — data lub dzień

Długość danych zawsze wynosi 8 bajtów. Data zapisana w formacie „YYYYMMDD”, gdzie: „YYYY” cztery cyfry roku, „MM” dwie cyfry miesiąca, „DD” dwie cyfry dnia w kalendarzu Gregoriańskim.

Przykład: „19800716” oznacza 16 lipca 1980

UWAGA: Standard „ACR-NEMA Standard 300”, czyli poprzednik DICOM definiował datę w sposób „YYYY.MM.DD”, według standardu DICOM, taki zapis jest nie poprawny, ale zdarzają się stare obrazy z takimi datami i *Sokar::DataConverter* obsługuje taki format.

- DS — Decimal String — liczba zmiennoprzecinkowa lub ciąg kilku liczb zmiennoprzecinkowych zapisanych za pomocą tekstu w notacji wykładniczej

Długość jednej liczby powinna maksymalnie wynosić 16 bajtów. Dostępne znaki to „0”-„9”, „+”, „-”, „E”, „e”, „.”. Biblioteka QT posiada wbudowany konwerter liczb zapisanych w formacie wykładniczym.

Przykład: „426\468 ” oznacza dwie liczby 426 i 468. Proszę zwrócić uwagę na spacje na końcu.

- IS — Integer String — liczba całkowita

Długość jednej liczby powinna maksymalnie wynosić 12 bajtów. Dostępne znaki to „0”-„9”, „+”, „-”. Biblioteka QT posiada wbudowany konwerter liczb całkowitych.

Przykład: „426 ” oznacza liczbę 426.

- PN — Person Name — nazwa osoby

Ponieważ pacjenta, bądź obiekt badany można nazwać w sposób dowolny i odiegający od polskiego standardu nazewnictwa, standard DICOM nie przewiduje rozdzielenia poszczególnych składowych nazwy na oznaczone fragmenty. „Person Name” dzieli nazwę na podane fragmenty, rozdzielony znakiem „^” (94 znak kodu ASCII):

## YBR

YBR albo  $YC_bC_r$  to model przestrzeni kolorów do przechowywania obrazów i wideo. Wykorzystuje do tego trzy typy danych: Y – składową luminancji, B lub  $C_b$  – składową różnicową chrominancji Y-B, stanowiącą różnicę między luminancją a niebieskim, oraz R lub  $C_r$  – składową chrominancji Y-R, stanowiącą różnicę między luminancją a czerwonym. Kolor zielony jest uzyskiwany na podstawie tych trzech wartości. YBR nie pokrywa w całości RGB, tak jak RGB nie pokrywa YBR. Posiadały one część wspólną, a część która nie jest wspólna ulega zniekształceniu.

Wartości w pliku DICOM są ulożone w taki sposób:

$$Y_1, B_1, R_1, Y_2, B_2, R_2, Y_3, B_3, R_3, Y_4, B_4, R_4, \dots$$

Ponieważ wartości te reprezentują kolory, są już formą obrazu, ale nie można ich jeszcze wyświetlić na monitorze RGB. Dlatego należy przekonwertować kolor YBR na kolor RGB, iterując po wszystkich wartościach obrazu.

Poniżej przedstawiono kod źródłowy funkcji zamiany kolorów YBR na RGB.

```

1 Sokar::Pixel ybr2Pixel(quint8 y, quint8 b, quint8 r) {
2     qreal red, green, blue;
3
4     red = green = blue = (255.0 / 219.0) * (y - 16.0);
5
6     red += 255.0 / 224 * 1.402 * (r - 128);
7     green -= 255.0 / 224 * 1.772 * (b - 128) * (0.114 / 0.587);
8     green -= 255.0 / 224 * 1.402 * (r - 128) * (0.299 / 0.587);
9     blue += 255.0 / 224 * 1.772 * (b - 128);
10
11    /* W tym miejscu jest dokonywana utrata danych */
12    red = qBound(0.0, red, 255.0);
13    green = qBound(0.0, green, 255.0);
14    blue = qBound(0.0, blue, 255.0);
15
16    return Sokar::Pixel(quint8(red), quint8(green), quint8(blue));
17 }
```

### 4.6.2 Generowanie obrazu monochromatycznego

Obraz monochromatyczny to obraz w odcieniach szarości, od białego do czarnego lub od czarnego do białego. Dane są zapisane w sposób ciągły wartość po wartości.

#### Pseudokolorowanie obrazu

Mamy obraz, którego piksele to n-bitowe liczby, na przykład 16-bitowa liczba całkowita. W takiej postaci wyświetlenie obrazu na monitorze RGB lub nawet na profesjonalnym 10-bitowym jest niemożliwe. Należy taką liczbę przekonwertować na trzy liczby, reprezentujące 3 kanały RGB, czerwony, zielony i niebieski. Dlatego do wyświetlania obrazów monochromatycznych o dużym kontraście stosuje się twór zwany okienkiem. Jest to funkcja, która mapuje n-bitowy obraz na 8-bitowy obraz w skali szarości. Wykorzystuje się 8 bitów ponieważ monitor RGB jest w stanie wyświetlić 256 odcieni szarości.

#### Zwiększenie kontrastu za pomocą „funkcji okna”

Jest przyjęte, że „okno” definiuje się dwoma liczbami: środkiem, oznaczanym jako  $center$  i długością, oznaczaną jako  $width$ . Wyznaczamy zakres okienka  $x_0$  i  $x_1$  ze środka

W tomografii komputerowej wytłkiem rekonstrukcji jest macierz hęzq opisującej rozkład przestrzenny współczynnika ostabilizacji promieniowania. Ze względem na aspekty praktyczne i medyczne, niezwykle istotna rzeczą jest zapisy dotyczące numerycznych parametrów typu akwizycji itp. W przeszukiwaniu wybierane są parametry warunkowe powodzące spadek informatywnego zakresu, który jest zapisywany w postaci plików cyfrowych. W celu tego powodu produkenci stworzyli wersje różnych programów, które dostarczają informacje o charakterystykach danego skanera.

## 2.4.3 DICOMDIR

- SS — Signed Short — 16-bitowa liczba całkowita bez znaku
  - US — Unsigned Short — 16-bitowa liczba całkowita bez znaku
  - UT — Unlimited Text — tekst o nieograniczonej długości.
  - Zwykły tekst o długości maksymalnie  $2^{32} - 2$  bajtów.

- given name complex — imię, np. Adam,
- middle name — środkowe imię — środkowe imię, brak odpowiednika,
- name prefix — przekleś imię, np: inż.,
- name suffix — sufixy — sufixy po imieniu, brak odpowiednika.

Dlhosce jadnega flagmantu powinnia maksymalne wyznoscie 64 znaki. W przypadku miniszesz isolscy segmentow, many zafazy, ze się pustce.

Przykład: „prof. dr. hab. inż. Jan Nowak Pracownik ZEJIM” były zapisany w sposob mestefujacy: „Nowak Jan prof. dr. hab. inż. Pracownik ZEJIM”.

#### 2.4.4 Line formaty zapisu

W tomogramie komputerowej wykryte m rekonstrukcji jest mniej więcej taka sama jak w tomografii w przestrzeni wspolczesnej ostatecznie oznacza, iż rozkład przestrzenny wypłaszczyliśmy do płaszczyzny, ale zapis informacji o rozkładzie przestrzennym pozostaje niezmieniony. Zatem w przestrzeni ośrodkowej przekroju powodem powstania niezwykłej istotnej izoczą jest zapis informacji o rozkładzie przestrzennym, który jest dany w postaci parametrycznej, a nie numerycznej.

Wartości obrazu są przepisywane do **targetBuffer** dla biblioteki QT.

- $R_n$  — war tość czernownego Kamahitu,
  - $G_n$  — war tość zielonego Kamahitu,
  - $B_n$  — war tość niebieskiego Kamahitu.

gdzie:

$$B_1, B_2, R_3, R_4, \dots, G_1, G_2, G_3, G_4, \dots, B_1, B_2, B_3, B_4, \dots$$

- 0 — oznacza to, że wszystkie płytki są ulokowane w taki sposób:
  - 1 — oznacza to, że wszystkie płytki są ulokowane w taki sposób:

Ubrzow zapasnych w HGR nie tycza wazan sposob odrabiac, aleze ghotowe do wyswietlania. Nalezy je odchowac do posortowania. Szczegolne zera i brak koloru powinny zostac usuniety przed ukladaniem na pasek.

Ubrz monochromatyczny to obraz w określonym zakresie, od bieli do czerni, który pozwala na rozpoznanie i rozróżnianie obiektów na tle tła. Czarno-białe zdjęcia charakteryzuje brak barw, natomiast kolorowe charakteryzuje obecność barw. Czarno-białe zdjęcia charakteryzuje brak barw, natomiast kolorowe zdjęcia charakteryzuje obecność barw.

## Ubrzą monocromatyczny

jest na nowo generowany dla bazy **image**.  
Całe odświeżanie odrzuń jest implementowane w funkcji **Sokar::DicomScene::reloadP-  
xmaps()**. Funkcja wywołuje **Sokar::DicomScene::generateP-  
xmap()** i odświeża **PixmaPItem**  
kiedy zjadzie taka potreba.

- **IconPixmap** — obiekt odrzużany ikoną, klasę `Qt::QPixmap`, docelowo powiniene mieć 128 pikseli na 128 pikseli.

do wyswietlania ekranie, ktora moze byc uzywana jako urzadzenie do malaowania w bibliotece Q.

## Rozdział 3

# Biblioteki i narzędzia

### 3.1 CMake

CMake to wieloplatformowe narzędzie do automatycznego zarządzania procesem komplikacji programu. Jest to niezależne od kompilatora narzędzie pozwalające napisać jeden plik, z którego można wygenerować odpowiednie pliki budowania dla dowolnej platformy.

Z uwagi na to, że projekt musi mieć możliwość komplikacji na 3 platformy CMake jest idealnym rozwiązańiem. Dodatkowo w pracy tej starano się wybrać biblioteki, które komplikują się za pomocą CMake.

### Licencja

CMake został opublikowany na licencji BSD, zgodnej z zasadami wolnego oprogramowania. Powstał początkowo na Uniwersytecie Kalifornijskim w Berkeley. Licencje BSD skupiają się na prawach użytkownika. Są bardzo liberalne, zezwalają nie tylko na modyfikacje kodu źródłowego i jego rozpowszechnianie w takiej postaci, ale także na rozprowadzanie produktu bez postaci źródłowej czy włączenia do zamkniętego oprogramowania, pod warunkiem załączenia do produktu informacji o autorach oryginalnego kodu i treści licencji. W programie została załączona informacja o użyciu CMake, więc jest możliwość użycia jej w pracy.

### 3.2 QT

Biblioteka Qt, rozwijana przez organizację Qt Project, jest zbiorem bibliotek i narzędzi programistycznych dedykowanych dla języków C++, QML i Java.

Qt jest głównie znana jako biblioteka do tworzenia interfejsu graficznego, jednakże posiada ona wiele innych rozwiązań ułatwiających programowanie obiektowe i zdarzeniowe.

W tej pracy wybrano biblioteki Qt z uwagi na to, że posiada interfejs w C++. Komplikacja oprogramowania używającego Qt może odbywać się za pomocą dwóch narzędzi: CMake oraz dedykowanego narzędzia „qmake”, zrobionego specjalnie na potrzeby biblioteki Qt. Dzięki czemu cały projekt przeglądarki używa tego samego języka oraz tego samego narzędzia zarządzania komplikacją.

- Help:

- About Qt — otwiera okno informacji o bibliotece Qt, Biblioteka Qt ma wbudowane takie okno w postaci `Qt::QMessageBox::aboutQt()`,
- About GDCM — otwiera okno z informacjami o bibliotece GDCM, implementowane przez funkcje `Sokar::About::GDCM()`,
- About Sokar — otwiera okno z informacjami o aplikacji, implementowane przez funkcje `Sokar::About::Sokar()`.

## 4.6 Algorytmy

### 4.6.1 Cykl generowania obrazów

Klasa `Sokar::DicomScene` dostarcza następujące obiekty do generowania obrazu:

- `processing` — obiekt klasy `Qt::QMutex`, zamek do zablokowania podczas generowania obrazu, aby parametry obrazu nie mogły być zmieniane podczas jego generowania,
- `imgDimX` — zmienna typu `uint`, oznacza szerokość obrazu w pikselach,
- `imgDimY` — zmienna typu `uint`, oznacza wysokość obrazu w pikselach,
- `targetBuffer` — wektor docelowego obrazu RGB o długości `imgDimX * imgDimY`, typu `std::vector<Pixel>`.

`Sokar::Pixel` to struktura reprezentująca piksel. Nie jest to w żadnym wypadku obiekt, a jedynie twór ułatwiający zarządzanie kodem.

```
1 struct Pixel {  
2     quint8 red = 0;  
3     quint8 green = 0;  
4     quint8 blue = 0;  
5 }
```

C++ od standardu C++03 przewiduje, że elementy znajdujące się w `std::vector` są ulozone ciągiem, jeden za drugim. Dlatego odwołując się do wskaźnika pierwszego elementu w ten sposób `&targetBuffer[0]`, można potraktować to jako tablicę.

- `originBuffer` — wektor danych wypełniona danymi z jednej ramki o długości iloczynu `imgDimX * imgDimY` i ilości bajtów jednego piksela obrazu,
- `qImage` — obiekt obrazu klasy `Qt::QImage`.

`Qt::QImage` można utworzyć z bufora, w tym przypadku jest to `targetBuffer`. Format obrazu to `Qt::Format_RGB888`, czyli trzy bajty, każdy na jeden kanał. Proszę zwrócić uwagę, że struktura `Sokar::Pixel` odpowiada temu formatowi. Według dokumentacji Qt obiekt ten po utworzeniu z istniejącego bufora powinien z niego dalej korzystać, dlatego zmiany `targetBuffer` nie wymagają odświeżania `qImage`.

- `pixmap` — obiekt obrazu do wyświetlania, klasy `Qt::QPixmap`. Obiektów klasy `Qt::QImage` nie da się wyświetlić, nie jest on przystosowany do wyświetlania. Natomiast klasa `Qt::QPixmap` to reprezentacja obrazu dostosowana

W roznych systemach operacyjnych sie rozuze komplikatory i wskad troszadno-  
scie poswiaita sie problem dotyczacy zmieniny fundamantalitycz. Przykjad jest zaga-  
dnie: ile bitow ma zmienina [int](#)? Udzaje sie do dokumentacji C++, dostepna pod  
adresem <https://pl.cppreference.com/w/cpp/language/types>. Natomiast w dokumencie Msvc, komplikatora firmy  
Microsoft, ze [int](#) ma minimum 16 bitow. Natomiast w dokumencie <https://docs.microsoft.com/pl-pl/cpp/int32--int64>,  
mische pewnosc o dzugosci liczby calkowitej nalezy uzyc takich typow: [int8](#), [int16](#),  
[int64](#). Dla tego dzugosc zmieniila sie do systemu komplikatora ora zapewniaj pewnosc podczas deklaracji,  
ktore dostosowuja sie do systemu komplikatora ora zapewniaj pewnosc podczas deklaracji,  
ze dana zmienina bedzie zakladane dzugosc. Doda to wie tyty literakow sa dostepne w  
manglowku [Getglobals](https://getglobals), dokumenciega dosteplna pod adresem <https://doc.qt.io/qt-5/qglobal.h.html>.  
Dla tego w pracy zosatly uzyte typy fundamentalne dostarczane przez bibliotekę Qt.

### B.2.4 Globálne typy struktúr

- ISO 9001:2015.
  - IEC 62304:2015 (2006 + AI),
  - IEC 61508:2010-3 7.A.4 (SIL 3),
  - IEC 60068-2-1015 (2006 + AI),

The QT Company posiada serię certyfików od FDA i UE, które ułatwiają wprowadzenie produktów do rynku medyczny.

### 3.2.3 Normy i certyfikaty

Biblioteka Gt jest dstrybutora w dwoch wersjach: komercyjne i otwarta zdrojowes. Wersja komercyjna jest dstrybutora nie poszada wielu modułów, ale jest dstrybutora na licencji GNU General Public License w wersji 3, co pozwala na użycie tej biblioteki dla celów pracy.

### 3.2.2 Licencja

- <https://www.qtcentre.org/threads/11347-How-do-you-pronounce-Qt>
  - <https://ubuntuforums.org/showthread.php?t=1605716>

Wedgek autorów, (t) powinni się czyste jak angielskie słowo „cute”, po Polsku „kutu”. Według autorki, (t) powinni się czyste jak angielskie słwo „cool”, po Polsku „fajne”.

### 3.2.1 Wyrmwa

- **File:**
  - Open — otwiera okienko wyboru
  - Open File Name(), nastepnie ::getOpenFileName()
  - Open Recent — program zapamiętuje ostatnie z tego menu
  - Export as — zapisać obrazu w
  - Exit — wyjście z aplikacji.

Otworzene nowelego pliku moze odbyc sie z nastepujacych zrodow: obiekty dzewia ze struktura plikow w systemie (opisaneego w § 5.4), menu programu (opisaneego w § 5.6), lub porzadku przeciazanego i uproszczonego pliku. Z dwoczt piramidowymi roztagami tylko po jednym pliku, automatycznie spodobem mowia roztagiem zapisowanym jak i wiele plikow. Wyslyshane prosby opliwa sie za pomoca dwoch funkcji: `Sokar::DicomTabs::addDicomFile()` i `Sokar::DicomTabs::addDicomFiles()`. Kazda z tych funkcji ma dwa parametry: nazwe pliku i adres do obiektu, pliki zostaja wczystane za pomoca funkcji `gdem::` `Wczytywania plikow`.

- `qint8` — liczba całkowita, 8-bitowa, ze znakiem,
- `qint16` — liczba całkowita, 16-bitowa, ze znakiem,
- `qint32` — liczba całkowita, 32-bitowa, ze znakiem,
- `qint64` — liczba całkowita, 64-bitowa, ze znakiem,
- `quint8` — liczba całkowita, 8-bitowa, bez znaku,
- `quint16` — liczba całkowita, 16-bitowa, bez znaku,
- `quint32` — liczba całkowita, 32-bitowa, bez znaku,
- `quint64` — liczba całkowita, 64-bitowa, bez znaku,
- `qreal` — największa dostępna liczba zmiennoprzecinkowa.

### 3.2.5 Klasa QObject

W dokumencie są wielokrotnie zawarte odniesienia do klas z biblioteki Qt. Dlatego, aby zwiększyć czytelność pracy, została zastosowana konwencja poprzedzania klas z biblioteki Qt przedrostkiem `Qt::`, który jest za razem przestrzenią nazw. Przykład poniżej:

`Qt::QObject`

Wszystkie funkcje wewnętrz klas są oznaczone następująco:

`Qt::QObject::connect()`

Dodatkowo w dokumencie PDF klikając na nazwę klasy użytkownik zostanie przekierowany do oficjalnej dokumentacji Qt znajdującej się pod adresem <https://doc.qt.io/qt-5>.

Biblioteka Qt dostarcza klasę `Qt::QObject`, która jest bazą dla wszystkich obiektów Qt i wszystkie klasy współpracujące z biblioteką Qt powinny po niej dziedziczyć. `Qt::QObject` implementuje 2 podstawowe rzeczy: system drzewa obiektów (opisany w sekcji 3.2.5), system sygnałów (opisany w sekcji 3.2.5).

### Drzewa obiektów

W C++ jednym z największych problemów jest wyciek pamięci, który pojawia się wtedy, gdy zaalokujemy na stercie obiekt za pomocą operatora `new` i nie usuniemy go gdy ten będzie niepotrzebny.

`Qt::QObject` zakłada, że obiekty mogą mieć jednego rodzica, a rodzic może mieć wiele dzieci. Rodzica można przypisać podczas tworzenia obiektu oraz zmieniać go dowolnie w trakcie działania programu. Przypisanie rodzica dziecku oznacza to, że gdy wywołamy destruktor rodzica, ten wywoła destruktory dzieci i w ten sposób całe drzewo obiektów zostanie zniszczone.

### Pasek filmu

Pasek filmu znajduje się w dolnej części zakładki i jest implementowany przez klasę `Sokar::MovieBar`. Ma dostęp do sekwencji scen i ukrywa swoją obecność przed użytkownikiem, kiedy w sekwencji jest tylko jedna scena.

Pasek jest podzielony na trzy części: trzy przyciski znajdujące się po lewej, pasek pokazujący postęp sekwencji na środku i prządko z trzema przyciskami po prawej.

Trzy lewe przyciski odpowiadają za poruszanie się po sekwencji. Wciśnięcie pierwszego przycisku (z indeksem 8 na rysunku 4.9) powoduje zatrzymanie upływu sekwencji i wysłanie sygnału `Sokar::SceneSequence::stepBackward()` do sekwencji. Wciśnięcie drugiego przycisku (9) powoduje wyłączenie lub wyłączenie upływu sekwencji. Wciśnięcie trzeciego przycisku (10) powoduje zatrzymanie upływu sekwencji i wysłanie sygnału `Sokar::SceneSequence::stepForward()` do sekwencji.

Pasek (11) pokazujący postęp sekwencji jest obiektem klasy `Qt::QSlider`. Odświeżanie paska jest wrażliwe na sygnał `Sokar::SceneSequence::stepped()` od sekwencji.

Elementy po prawej stronie definiują parametry trybu filmowego. Prządka (12) jest elementem do wprowadzania liczb zmiennoprzecinkowej klasy `Qt::QDoubleSpinBox`. Im większa wartość liczby, tym klatki filmu są dłużej wyświetlane. Drugi (13) przycisk pozwala zmienić sposób przemiatania. Trzeci (14) przycisk wymusza tryb jednego okienkowania dla wszystkich klatek filmu. Jeżeli mamy załadowanych wiele obrazów tego samego badania, to nie koniecznie muszą mieć to samo okno. Dodatkowo ten tryb pozwala wprowadzić jednolite okienko dla wszystkich klatek po zmianie parametrów tego okienka na jednej klatce. Czwarty (15) i ostatni przycisk służy do użycia jednej macierzy transformaty na wszystkich klatkach.

### Tryb filmowy

Tryb filmowy można aktywować jedynie wtedy, gdy w sekwencji scen jest więcej niż jedna scena. Włączenie trybu filmowego polega na stworzeniu obiektu klasy `Sokar::MovieMode`. Obiekt ten zapisuje wskaźnik do obecnie wyświetlonej sceny, a także czy powinno być użyte to samo okno, oraz czy powinna być używana ta sama macierz przekształcenia. Następnie obiekt ten jest wysyłany do wszystkich scen w sekwencji. Uruchamiany jest timer, czyli obiekt klasy `Qt::QTimer`, na czas równy czasowi trwania sceny zapisanego w kroku przemnożonego przez liczbę z prządki. Po upływie timera, wstawiana jest nowa scena za pomocą sygnału `Sokar::MovieBar::setStep()`, a timer jest ustawiany na nowo.

### Podgląd miniaturek

Ten element to wybór scen za pomocą ikon, implementowany przez klasę `Sokar::FrameChooser`. Element, podobnie jak pasek filmu, ma dostęp do sekwencji scen i ukrywa swoją obecność przed użytkownikiem, kiedy w sekwencji jest tylko jedna scena. Po wciśnięciu ikony scena jest zmieniana.

### 4.5.5 Obiekt zakładek

Obiekt zakładek, implementowany za pomocą klasy `Sokar::DicomTabs`, odpowiada za wyświetlanie wielu obiektów zakładek w jednym obiekcie interfejsu. Obsługuje również wczytanie nowych plików.

Sygnalny i sloty

```

int main() {
    // Tworzymy obiekt przycisku
    auto *pustek = new GłówneDziałanie("Pustek");
    // Tworzymy obiekt okna
    auto *win = new Okno(pustek);
    // Tworzymy obiekt przycisków
    auto *quit = new Przycisk("Wyjdź");
    // Przypisujemy dodatkowa przycisków
    quit->SetParent(windows);
    // W tym momencie przycisk wrz z oknem zostaje usunięty
    delete windows;
}

```

Mieczam zatem pozwala nam tworzyć nowe obiekty na stercie i inne maturalne siegi o których mówiące czytelnicy mogłyby znać. Przykładowe uzycie:

- Rotate Left — Obrotę w lewo  
Akcja: **RotatLeft90**.
  - Rotate Left — Obrotę w lewo  
Po otwyzmaniu sygnału obraz na scenie powinienn obroć się o 90 stopni w lewo.
  - Flip Horizontal — Odwróć lustrzaną pozycję  
Akcja: **FlipHorizontal**.
  - Flip Vertical — Odwróć lustrzaną pozycję  
Po otwyzmaniu sygnału obraz na scenie powinienn odwrócić się lustrzaną pozycję.
  - Flip Vertical — Odwróć lustrzaną pozycję  
Akcja: **FlipVertical**.
  - Clear Transformation — Wyczyść przekształcenia obrotu  
Akcja: **ClearRotate**.
  - Clear Transformation — Wyczyść przekształcenia obrotu  
Po otwyzmaniu sygnału obraz na scenie powinienn odwrócić się lustrzaną pozycję.

System szygarnia w slotow jest implementacj<sup>a</sup> programowa na dzierzazie. Szygarnia jest zrodolem dzierzazni, a slot jest odwrotnikiem programowania dzierzazni. Szygarnia slotu obiektu dynamiczne w czasie dzierzaznia. Szygarnia dodatkowe wele slotow, jak i do jednego slotu moza w programowaniu. Zasada dzierzazne zostanie wewnetrowane, to wszytskie sloty podklzone do szygarni. Szygarnia 1 sloty sa implementowane prez funkcje definiowanymi. System szygarnia w slotow sa implementowane w systemie SIGTERM. Dodaekowo szygarnia w slotach poszukiwana jest algorytmem dzierzazni. Takie implementacja mozliwia programowanie dzierzazni.

- Ten element połaci wyjazdowy zwiastował niekotorych elementów na scenie. Kilkunie-

- Po oztřímyaní sýgmažuh obiectk klasý *Sokar::HospitalDataIndicator* znamená významnou sečení použitím polozkazé hbu ukrýte siže w zalednoscí od stanin pozycí.

```

        Counters, b:, * Twarzamy da obiekty klasycz Counter (definičia s nastrepnym sekcijí)
        /* Leczymy sygnal Countere: valueChanged obiectu "a",
           do slotu myCounter::setValue obiectu "b" */
           do slotu moza tez odpilne myczene lambda *
           /* Do slotu moza tez odpilne myczene lambda */
           /* Dlaczego? zazwyczaj konieczne jest w dzialej czesci dokumentu*/
           /* leczmy sygnal Countere: valueChanged obiectu "a",
           /* Leczymy sygnal Countere: valueChanged obiectu "a",
           /* W caelsie ustawiania zostat myczany sygnal z "a" do "b", wicze:
           a.value() == 12 b.value() == 12 */
           a.setValue(12);
           /* Ustawiamy wartosc licznika obiectu "a" na 12 */
           a.setValue(12);
           /* Sygnal Counter: valueChanged obiectu "b" na 48 */
           a.setValue(48);
           /* Zmieniamy sloto, wicze:
           a.value() == 12 b.value() == 48 */
           a.setValue(48);
           /* Dlaczego? zazwyczaj konieczne jest w dzialej czesci dokumentu*/
           /* Leczymy sygnal Countere: valueChanged obiectu "a",
           /* Leczymy sygnal Countere: valueChanged obiectu "a",
           /* Twarzamy da obiekty klasycz Counter (definičia s nastrepnym sekcijí)

```

- Tagi (5)Akcja: [OpenDataSet](#).

Kliknięcie tego przycisku wysele prostę o otworzenie okna ze zbiorem elementów danych. Pliku o nazwie klasa Skarby:DicomGraphics, dziedziczącej po [Qt::QGraphicsScene](#).

## Przykładowa klasa dziedzicząca po QObject

```
1 #include <QObject>
2
3 class Counter : public QObject {
4     /* Każda klasa dziedzicząca po QObject musi na samym
5      * początku swojej definicji mieć makro "Q_OBJECT". */
6     Q_OBJECT
7
8 public:
9     Counter() { m_value = 0; }
10    int value() const { return m_value; }
11
12    /* Sloty powinny być poprzedzone makrem "slots".
13     * Widoczność slotów można zmieniać. */
14 public slots:
15    void setValue(int value){
16        if (value != m_value) {
17            m_value = value;
18
19            /* Podczas wywoływania sygnału należy
20             * poprzedzić to makrem "emit". */
21            emit valueChanged(value);
22        }
23    }
24
25
26    /* Sygnały powinny być poprzedzone makrem "signals".
27     * Wszystkie sygnały są publiczne. */
28 signals:
29    void valueChanged(int newValue);
30
31 private:
32    int m_value;
33};
```

### 3.2.6 Graficzny interfejs użytkownika

Graficzny interfejs użytkownika został zaimplementowany za pomocą klasy *Qt::QWidget*. Klasa ta dziedziczy po *Qt::QObject* i po *Qt::PaintDevice*, obiekcie służącym do rysowania. *Qt::QWidget* reprezentuje element graficzny interfejsu użytkownika, ma zaimplementowany mechanizm renderowania, wyświetlania na ekranie użytkownika, obsługu myszki klawiatury, przeciągnięcia i upuszczenia (ang. *drag and drop*), itp. Wszystkie elementy takie jak przyciski i pola tekstowe muszą dziedziczyć po niej.

Interfejs klasy jest niezależny od platformy, na której się znajduje. Nawet tworzenie własnej, niestandardowej kontrolki nie wymaga uwzględniania systemu operacyjnego, a przynajmniej w kwestii użytkowej.

Kilka przykładowych klas obiektów graficznych i ich cechy:

- *Qt::.QLabel* — klasa służąca do wyświetlania tekstu bez możliwości interakcji z nim. Dziedziczy po klasie *Qt::QFrame*, która dziedziczy po *Qt::QWidget*.
- *Qt::QPushButton* — klasa do tworzenia zwykłego przycisku. Dziedziczy po klasie *Qt::AbstractButton*, która dziedziczy po *Qt::QWidget*. Obsługa zdarzenia wciśnięcia przycisku jest przez obsługę sygnału *Qt::AbstractButton::clicked()*. Przykład można zobaczyć na rysunku 3.1.
- *Qt::QTabWidget* — implementuje zakładki, takie jak w przeglądarce internetowej. Dziedziczy bezpośrednio po klasie *Qt::QWidget*. Zawartości zakładek mogą być

## Pasek narzędzi

Pasek narzędzi znajdujący się na górze, implementowany jest przez klasę *Sokar::DicomToolBar*, dziedziczącą po klasie *Qt::ToolBar*. Posiada on zespół ikonek z rozwijalnymi menu kontekstowymi.

Kliknięcie odpowiedniej ikony spowoduje wysłanie sygnału do obecnie wyświetlanej sceny. Są dwa sygnały możliwe do wysłania *Sokar::DicomToolBar::stateToggleSignal()* lub *Sokar::DicomToolBar::actionTriggerSignal()*. Pierwszy sygnał oznacza zmianę stanu paska, czyli sposób obsługi myszki i zawiera jeden argument: stan (typu *enum*). Sygnał ten okazał się bezużyteczny i nie jest obecnie wykorzystywany przez scenę. Drugi oznacza akcję, która powinna być wykonana przez scenę. Zawiera dwa argumenty: typ akcji (typu *enum*) i stan akcji (typu *bool* z domyślną wartością *false*).

Ikonę na pasku:

- Okienkowanie (1)

Stan *Windowing* oznacza, że horyzontalny ruch myszki powinien zmieniać szerokość okna, a wertykalny środek okna. Przycisk jest aktywny tylko wtedy, gdy obecna scena posiada obraz monochromatyczny.

- Przesuwanie (2)

Stan *Pan* oznacza, że ruch myszki powinien przesuwać obraz na scenie w prawo, lewo, górę lub dół, kiedy jest wciśnięty lewy klawisz myszy.

Rozwijalne menu zawiera tylko jedne element „Move To Center” wysyłający sygnał akcji z argumentem *ClearPan*.

- Skalowanie (3)

Stan *Zoom* oznacza, że ruch myszki powinien skalować obraz kiedy jest wciśnięty lewy klawisz myszy.

Menu rozwijalne:

- Fit To Screen — Dopasuj do ekranu

Akcja: *Fit2Screen*.

Po otrzymaniu sygnału obraz na scenie powinien dopasować swoją wielkość do wielkości sceny

- Original Resolution — Skala jeden do jednego

Akcja: *OriginalResolution*.

Po otrzymaniu sygnału obraz na scenie powinien dopasować swoją wielkość jeden do jednego w stosunku do piksela na ekranie.

- Rotacja (4)

Stan *Rotate* oznacza, że ruch myszki powinien obracać obrazem znajdującym się na scenie.

Menu rozwijalne:

- Rotate Right — Obróć w prawo

Akcja: *RotateRight90*.

Po otrzymaniu sygnału obraz na scenie powinien obrócić się o 90 stopni w prawo.

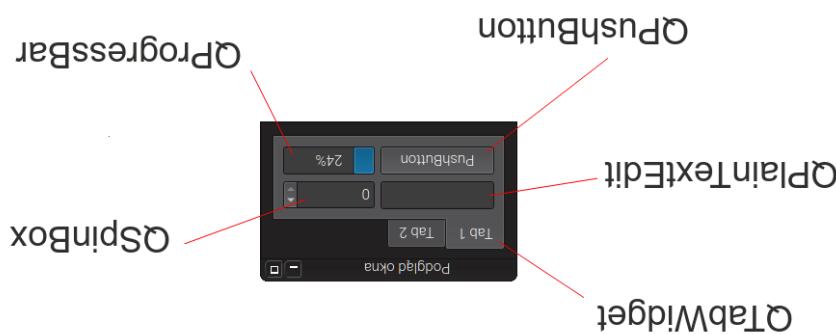
- współpracowanie z językiem C++,
- mniej licencji powallać się jeźwiąc w potrzebny zakresie,
- być darmowa, najlepiej otwarto źródłowa,
- aby aktywnie rozwiązała — znaczna większość bibliotek charakteryzuowała się tym,
- aby porzuciona i ostatecznie zmiana była wprowadzona wiele lat temu, a proces jeli rozwoju trwał od 2 do 5 miesięcy,

Magin", dostrej pod adresem <https://idiotmagining.com/programs>. Biobletek, ktore poszukiwane w tej pracy powinna:

### 3.3.1 Uzasadnienie wyboru

3.3 GDCM

**Wykuszek 3.1:** Izometryczne określone przez jedną w gęsi. Zadanie wiejskie.



The screenshot shows a window titled "QPlainTextEdit". Inside the window, there is a "QTabWidget" containing two tabs labeled "Tab 1" and "Tab 2". Below the tabs, there is a "QSpinBox" with the value set to 0. To the right of the spin box is a "QPushButton" with the text "Pushbutton". The status bar at the bottom displays the message "Podg醕ek okna".

**Rysunek 4.9:** Zakkadka wraz z numeracją elementów interfejsu. Zdjécie własne.



Dodatakowo postada obiekt kolekcji scen opisana w sekci 4.5.3.

- Podgląd minimalistyczny dla zasobów w prawie czeków — implementowany za pomocą klasy *Sokar::FrameChooser*, opisany w sekci 4.4.

- *Sokar*: *Diagrammatic*, *optical* w *sketchy* 4.5.A.
- *swak*: *flimy* w *dolhei* *cegei* — *implementowany za pomoca kasy* *Sokar*: *MonteBar*.

• DicomToolBar, opisany w sekcií 4.5.4, poskytuje si trojúhelník implementaci za použití rady `Scalable`.

Każąda zakładała z obrazem lub obrazami jest implementowana przesyłka klasy *Sokar*:  
*komView*.  
Interfejsy graficzny *Sokar*: Docom View wyświetla nastepująco elementy:

Seeggħaż-ja abbywa s-Solar: `DicomFileSet:create()`. Do finkjekk jest es-safha kollha. Ma nistgħix minnha minn `DicomFileSet` u minn `Solar`.

- być dostępna na Linuxa, MacOS i Microsoft Windows.

Ostatecznie podjęto decyzję o wyborze biblioteki o nazwie Grassroots DICOM (GDCM), dostępną pod adresem <http://gdcm.sourceforge.net/>.

### 3.3.2 Opis

Przetłumaczony opis biblioteki z oficjalnej strony prezentuje się następująco: Grassroots DICOM (GDCM) to implementacja standardu DICOM zaprojektowanego jako open source, dzięki czemu naukowcy mogą uzyskać bezpośredni dostęp do danych klinicznych. GDCM zawiera definicję formatu pliku i protokołów komunikacji sieciowej, z których oba powinny zostać rozszerzone dla zapewnienia pełnego zestawu narzędzi badaczowi lub małemu dostawcy obrazowania medycznego w celu połączenia z istniejącą bazą danych medycznych.

GDCM jest biblioteką posiadającą możliwość wczytywania, edycji i zapisu plików w formacie DICOM. Obsługuje ona wiele kodowań obrazów jak i protokoły sieciowe. Jest w całości napisana w C++, a do komplikacji używa CMake. Dzięki temu w całym programie jest używany język C++ wraz z CMake, co ułatwia zarządzanie procesem kompilacji do jednego pliku.

Główna zaletą biblioteki jest dobra dokumentacja wraz z przykładami jej użycia, które okazały się kluczowe przy wyborze. Biblioteka została napisana w sposób obiektowy z usprawnieniami zawartymi w C++, takimi jak referencje i obiekty stałe, co ułatwia jej użycie.

### 3.3.3 Licencja

GDCM jest wydana na licencji BSD License, Apache License V2.0, która jest kompatybilna z GPLv3. Licencja ta dopuszcza użycie kodu źródłowego zarówno na potrzeby wolnego oprogramowania, jak i własnościowego oprogramowania.

### 3.3.4 Podstawowe klasy

W dokumencie są wielokrotnie zawarte odniesienia do klas z biblioteki GDCM. Dlatego, aby zwiększyć czytelność pracy, została zastosowana konwencja poprzedzania klas z biblioteki Qt przedrostkiem `gdcm::`, który razem jest przestrzenią nazw biblioteki. Przykład poniżej:

```
gdcm::ImageReader
```

Wszystkie funkcje wewnętrz klas są oznaczone następująco:

```
gdcm::ImageReader::GetImage()
```

Dodatkowo w dokumencie PDF można kliknąć na nazwę klasy i użytkownik zostanie przekierowany do oficjalnej dokumentacji GDCM znajdującej się pod adresem <http://gdcm.sourceforge.net/html>.

Przedstawiono kilka podstawowych klas:

- `gdcm::Reader` — klasa służąca do wczytywania pliku DICOM,

- `Sokar::SceneSequence::stepBackward()` — krok do tyłu — zmniejsza indeks tym samym wykonując krok w stronę początku sekwencji,
- `Sokar::SceneSequence::step()` — wykonuje krok w tył lub przód w zależności od kierunku sekwencji.

Wszystkie powyższe funkcje są zarazem slotami dla sygnałów oraz emitują sygnał `Sokar::SceneSequence::stepped()`.

### Kolekcja ramek DICOM

Zbiory ramek są implementowane przez `Sokar::DicomFrameSet` i są tworzone z jednego wczytanego pliku DICOM. Klasa tworzy obiekt konwertera i pobiera liczbę ramek w obrazie. Tworzy jeden bufor na wszystkie ramki obrazów, a następnie dzieli go na ilość ramek. Biblioteka GDCM nie daje dostępu do oryginalnego bufora, dlatego wymagany jest bufor pośredni. Następnie jest tworzonych tyle obiektów scen ile jest ramek.

Kolejność sekwencji scen jest taka sama jak kolejność ramek. Natomiast czas wyświetlania ramki może być zapisany w różnych znacznikach. To, w którym znaczniku został zapisany, informuje element o znaczniku `Dicom Tag Frame Increment Pointer` (0x0028, 0x0009). Zawiera on wskaźnik do elementu o zadanym znaczniku.

Została zaimplementowana obsługa ponizszych znaczników:

- `Dicom Tag Frame Time` (0x0018, 0x1063) — element z tym znacznikiem zawiera czas trwania jednej ramki w milisekundach. Każdemu krokowi jest przypisywana ta wartość trwania.
- `Dicom Tag Frame Time Vector` (0x0018, 0x1065) — zawiera tablice z przyrostami czasu w milisekundach między n-tą ramką a poprzednią klatką. Pierwsza ramka ma zawsze przyrost czasu równy 0.
- `Dicom Tag Cine Rate` (0x0018, 0x0040) — zawiera ilość klatek wyświetlnych na sekundę. Każdemu krokowi jest przypisywana wartość do niej odwrotna.

W przypadku braku znacznika lub gdy zostaje wskazany znacznik nieznany, czas trwania ramki wynosi 83.3 milisekundy, co odpowiada 12 klatkom na sekundę.

### Kolekcja plików DICOM

Zbiory plików są implementowane przez `Sokar::DicomFileSet` i służą do przechowywania wielu wczytanych plików DICOM. Na początku pliki są sortowane na podstawie liczby zawartej w elemencie o znaczniku `Dicom Tag Instance Number` (0x0020, 0x0013). Dla każdego pliku jest tworzony obiekt `Sokar::DicomFrameSet`.

Sekwencja jest tworzona poprzez połączenie sekwencji poszczególnych obrazów.

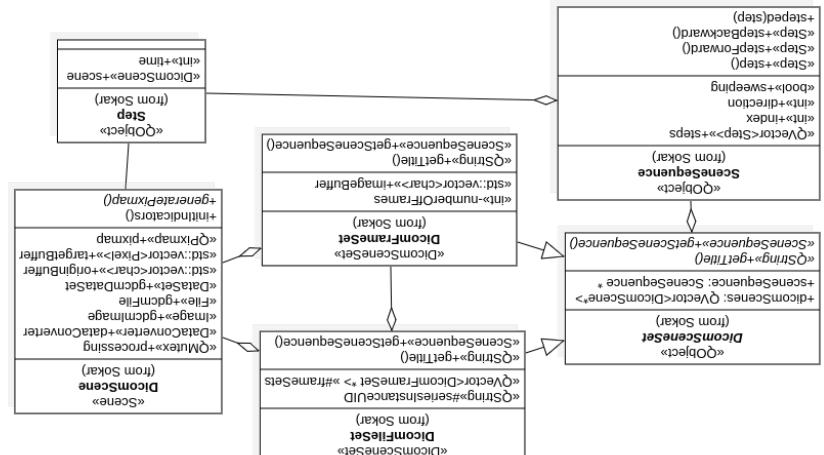
### Segregowanie obrazów

W przypadku kiedy mamy do czynienia z wieloma plikami, należy jest rozdzielić na serie i uporządkować w odpowiedniej kolejności. Unikalny identyfikator serii jest zawarty w elemencie danych o znaczniku `Dicom Tag Series Instance UID` (0x0020, 0x000E). Kolejności obrazów w serii to liczba zawarta w elemencie danych o znaczniku `Dicom Tag Instance Number` (0x0020, 0x0013).

- indeks, który m obecnie zna jduje się sekwencja,
  - indeks, który m wykonywać krok w stronę końca sekwencji,
  - indeks, który do przodu — zwiększa indeks tym razy;
  - Słownica ma świadomość funkcje zapewniające przesuwanie się po indeksie na wskazany indeks i zmieniające przekazane parametry klasy Słownika. Step zapisuje informacje: wskazany indeks, który do tego momentu warunek przekazany przez indeks, aktualna scena.
  - Słownica ma świadomość funkcje zapewniające przesuwanie się po indeksie na wskazany indeks i zmieniające przekazane parametry klasy Słownika. Step zapisuje informacje: wskazany indeks, który do tego momentu warunek przekazany przez indeks, aktualna scena.

Sekwencja scen

Rysunek 4.8: Diagram klas UML dziedziczenia klas *Sokar* i *DicomSceneSet*.



Abstractyjna klasa **Sokar**: **LichomScen**set mplemenyje wektor scen za pomocj klasy **Q::Vector**. Jest to obiek, ktory przechowuje sceny i tworzy sekwencje scen, ktora jest zezwyistym urozneiem ramk obrazow. Skad dwie implementacje klasa: klasa **Image** i klasa **Image** z jednego pliku. Dlatego klasa **Image** zada juz wiele mniej sliznych operacji.

#### 4.5.3 Kolekcje scen

- *gdcn::Image* — obiekt opierający na obrazu wyciągnięty poprzednim elementem informacji;
  - *gdcn::File* — obiekt pliku DICOM;
  - *gdcn::DataSet* — obiekt zbioru elementów;
  - *gdcn::DataElement* — obiekt elementu danych;
  - *gdcn::Tag* — obiekt znacznika;
  - *gdcn::StringFilter* — pomocnicza klasa służąca do konwersji na obiekt tekstu.

Poniżej zaprezentowane są kilka przykładowych użycia biblioteki GDCM.

### 3.3.5 Przykład użycia

## Przykład wczytania pliku

W poniższym przykładzie mamy do czynienia z wczytaniem pliku oraz pobraniem kilku wartości z elementów o danych znacznikach.

```
1 #include ...
2
3 int main() {
4
5     /* Tworzymy obiekt czytającego i wczytujemy plik */
6     gdcm::Reader reader;
7     reader.SetFileName("/path/to/file");
8     if (!reader.Read()) {
9         /* W przypadku wystąpienia błędu możemy go obsłużyć */
10        return 1;
11    }
12
13    /* Pobieramy obiekt pliku */
14    const gdcm::File &file = reader.GetFile();
15
16    /* Pobieramy obiekt zbioru danych */
17    const gdcm::DataSet &dataset = file.GetDataSet();
18
19    /* Tworzymy pomocniczą klasę do konwertowania danych na std::string */
20    gdcm::StringFilter stringFilter;
21    stringFilter.SetFile(file);
22
23    /* Tworzymy pomocnicze obiekty znaczników */
24    const static gdcm::Tag
25        TagPatientName(0x0010, 0x0010),
26        TagWindowCenter(0x0028, 0x1050),
27        TagWindowSize(0x0028, 0x1051);
28
29    /* Pobieramy tekst, jeżeli się znajduje w zbiorze */
30    if (dataset->FindDataElement(TagPatientName))
31        std::string name = stringFilter.GetString(TagPatientName);
32
33
34    if (dataset->FindDataElement(TagWindowCenter)){
35        /* Pobieramy element ze zbioru danych */
36        const DataElement& ele = dataset->GetElement(tag);
37        /* Pobieramy 16-bitowego inta */
38        quint16 center = ele.GetByteValue()->GetPointer();
39    }
40
41    if (dataset->FindDataElement(TagWindowSize)){
42        const DataElement& ele = dataset->GetElement(tag);
43        quint16 width = ele.GetByteValue()->GetPointer();
44    }
45
46 }
```

– „KVP” — szczytowe napięcie wyjściowe generatora promieniowania rentgenowskiego wyrażone w kilovoltach, pobierane z <sup>Dicom</sup> Tag KVP (0x0018, 0x0060)

- MR — rezonans magnetyczny:

– „Repetition time” — czas repetycji — pobierany ze znacznika <sup>Dicom</sup> Tag Repetition Time (0x0018, 0x0080).

– „Echo time” — czas echa — pobierany ze znacznika <sup>Dicom</sup> Tag Echo Time (0x0018, 0x0081).

– „Magnetic field” — pole magnetyczne — nominalna wartość pola magnetycznego wyrażona w teslach pobierana ze znacznika <sup>Dicom</sup> Tag Magnetic Field Strength (0x0018, 0x0087).

– „SAR” — swoiste tempo pochłaniania energii — pobierane ze znacznika <sup>Dicom</sup> SAR (0x0018, 0x1316).

## Generowanie obrazów z danych

Klasa *Sokar::DicomScene* jest klasą abstrakcyjną i nie generuje obrazu. Pozostawia to klasom dziedziczącym po niej. Dokładna analiza cyklu generowania obrazów jest opisana w sekcji 4.6.1.

## Przekształcenia macierzowe obrazu

Wyświetlanie obrazu na scenie odbywa się za pomocą obiektu klasy *Qt::QGraphicPixmapItem*, który dziedziczy po *Qt:: QGraphicsItem*. Ta ostatnia klasa ma w sobie zaimplementowaną funkcję pozwalającą na nałożenie przekształcenia macierzowego na obraz. W Qt przekształcenia macierzowe są implementowane za pomocą klasy *Qt::QTransform*, która jest macierzą 3 na 3.

Zostały zdefiniowane 4 macierze, które działają na obiekt obrazu wyświetlanego na scenie:

- **centerTransform** — macierz wyśrodkowująca (zadaniem tego przekształcenia jest przeniesienie obrazu na środek sceny),
- **panTransform** — macierz przesunięcia,
- **scaleTransform** — macierz skali,
- **rotateTransform** — macierz rotacji.

Podczas interakcji z użytkownikiem macierze mogą ulegać zmianom na dwa sposoby. Pierwszym sposobem jest odebranie sygnału od przycisków z paska zadań, szerzej opisanego w sekcji 4.5.4, znajdującego się nad sceną. Drugi sposób to przechwycenie ruchów myszki, gdy wciśnięty jest lewy przycisk myszy.

P pełny algorytm tworzenia macierzy i ich zmian poprzez interakcje z użytkownikiem, znajduje się w sekcji 4.6.3.

## Prykład wczytania obrazu

W tej pizzeriaźwiżny usiądzieć wczty walcu za pionocą kasy przy stolówce! do tego.

W tym fizyki zadej wizualny sprawdzenie czy w danej obrazu za pomocą kresy Pizy - stosowanej do tego

Pětý opis implementací algoritmu wyznačení stroužadidlo je situace v sekci 4.6.A.

- „H“ — head — czéšc górná pacienta.

- „F“ — feet — cześć dla pacjenta,

- „P“ — posterior — tył pacjenta,

- „A” — anterior — prizod pacienta,

Podzialeka

Podziękka jest implementowana przez *Sokar:PixelSpaceConfigIndicator*. Obiekt wysyłający informacje o rozmiarach obiektu na obrazie. Podziękka dostosowuje swoje wielkość do obiektu sceny, jak i do innego elementów sceny. Wartością wyświetlającą będzie pod uwagę transformata skali i rotacji obrazu.

```

3 int main()
4 {
5     /* Tworzymy obiekt czystej akcjięgo i masytuje my plik */
6     gdcm::SefFileReader litr;
7     litr.SefFileReader("path/to/file");
8     if (litr.Read()) {
9         /* Wypakujemy wszystkie bieżące moźwmy go odszukajęc.
10        Klasa gdcm::Mastrapader zwróci błąd gdy w pliku nie
11        będzie obrazu, błąd zwróci błąd gdy w pliku nie
12        return 1;
13    }
14    /* Pobieramy obiekt obrazu */
15    const gdcm::Image &image = litr.GetImage();
16    /* Pobieramy rozmiary jego wielkości. */
17    std::vector<char> imgbuffer;
18    imgbuffer.resize(dimensions[0]);
19    std::vector<char> imgbuffer.resize(dimensions[1]);
20    imgbuffer.resize(dimensions[2]);
21    /* Podzieliemy myślnik na dwuwymiarową tablicę */
22    const Podzielnik &dimensions = gdcm::GetDimensions();
23    /* Podzieliemy obraz na dwuwymiarowe interpretacje */
24    const dwytna dmy = dimensions[1];
25    dwytna dmy = dimensions[0];
26    /* Tworzymy obiekt i zmieniamy jego wielkość. */
27    if (gdcm::GetPhotometricInterpretation() ==
28        gdcm::PhotometricInterpretation::RGB)
29    {
30        std::cout << "Jest to obraz monochromatyczny typu druków
31        itogi: count << endl;
32        gdcm::PhotometricInterpretation::MONO2B);
33        std::cout << "Jest to obraz monochromatyczny typu druków
34        itogi: count << endl;
35        itogi: count << endl;
36        std::cout << "Jest to obraz monochromatyczny typu druków
37        itogi: count << endl;
38        const gdcm::File file = litr.GetFile();
39        const gdcm::Database &database = litr.GetDatabase();
40        file.GetPixelFormat() == gdcm::PixelFormat::UNITS);
41        cout << endl;
42    }

```

- „Modality“ — modalnosc — pobierana ze znaczki Dicom Modality (0x0008, 0x0060),
- „Series“ — numer serii — pobierany ze znaczki Dicom Series Number (0x0020, 0x0011),
- „Instances number“ — numer instancji w serii — pobierany ze znaczki Dicom Instance Number (0x0020, 0x0013).

Zawiera jàdlastepujàdce linie:

```
Te informacije sej impljemennitovane prez Sora: ModalityIndication. Objekt vysvetla informacije o akviziciji obrazu. Dame rozumje sije w zaleznosti od modalnosti obrazu. Domyslime, ze modalnost obrazu je implementowana przez Sora.
```

Dodatakowe informacje o modalności

- Wartosci odnosz±ce siê do w±asciwo¶ci plastra obrazu: „Slice thickness” — grubosze plastra — pobierana ze znaçznika `Decm` Slice Thickness (`0x0018, 0x00050`), „Slice location” — pozycja plastra — pobierana ze znaçznika `Decm` Slice Location (`0x0020, 0x1041`).

- „KVP” — szczytowe napięcie wyjściowe generatora promieniotwórczościennego
- skiego — wyjazne kilowotach, pobierane z Dicom KVP (0x0018, 0x0060)
- „Exposure time” — czas ekspozycji — pobierany ze znacznika Dicom Exposure Time (0x0018, 0x1150).
- „Exposure” — ekspozycja — wyrażona w mAs, pobierana ze znacznika Dicom Exposure Tag (0x0018, 0x1152).

### 3.4 Git

Chт to system kontroll wersji. Cтata praca zostatka wykonaana przy aywscie tego nazzрedzia, a repozitoryum z programem znajdzie się pod adresem https://gl.itr.pw.edu.pl/jagdziejowski/LaTeX/mozina\_znalezionej adreszjejowkski/sokar-4pp. Znajduje się przy pismieni na pisaniej w LaTeX mozina znalezionej pod adresem https://gl.itr.pw.edu.pl/jagdziejowski/sokar-writing.

- RT/CR — radiologia analogowa i cyfrowa:
  - „Exposure time” — czas ekspozycji — poniejszy ze znacznika Dicom Exposure Time (0x0018, 0x1150).

- RT/CR — radiologia analogowa i cyfrowa:

# Rozdział 4

## Implementacja

Najbardziej rozpoznawalne dwie przeglądarki to Osirix i Horus. Ich nazwy zaczerpnięto od nazw egipskich bogów: odpowiednio od Ozyrysa, boga śmierci i Horusa, boga nieba. Nazwa przeglądarki omawianej w pracy będzie miała nazwę: Sokar.

Sokar w mitologii egipskiej to bóstwo dokonujące przyjęcia i oczyszczenia zmarłego władcę oraz przenoszący go na swej barce do niebios, patron metalurgów, rzemieślników i tragarzy (nosicieli lektyk) oraz wszelkich przewoźników.

### 4.1 Zakres implementacji

Po analizie możliwości przeglądarek plików DICOM dostępnych na rynku postanowiono zaimplementować następujące komponenty w opracowywanej przeglądarce:

- Obsługa obrazów bez względu na ich modalność, ale z ograniczeniem do następujących interpretacji fotometrycznej:
  - „MONOCHROME1”,
  - „MONOCHROME2”,
  - „RGB”,
  - „YBR”.
- Przesuwanie (ang. *pan*).
- Skalowanie lub powiększenie poprzez decymacje i interpolacje liniowe.
- Rotacja i odbicia lustrzane.
- Okienkowanie i pseudokolorowanie, zarówno w skali szarości jak i z użyciem wielokolorowych palet.
- Obsługa serii obrazów jako całości
  - przegląd obrazów w serii,
  - animacje,
  - wspólne okna w skali barwnej,
  - wspólne przekształcenia macierzowe.

- Opis lub klasyfikacja badania dokonana przez instytucję  
Tekst brany z <sup>Dicom</sup> Tag Study Description (0x0008, 0x1030) i wyświetlany bez ingerencji.  
UWAGA: Ta wartość jest wpisywana przez technika, operatora lub lekarza wykonującego badanie, więc wartość ta może być nieprzewidywalna.

- Opis serii  
Tekst brany z <sup>Dicom</sup> Tag Series Description (0x0008, 0x103E) i wyświetlany bez ingerencji.  
UWAGA: Ta wartość jest wpisywana przez technika, operatora lub lekarza wykonującego badanie, więc wartość ta może być nieprzewidywalna.

Przykład pełnego tekstu:

**Jan Nowak** ♂  
HIS/123456  
born 1996-01-01, 19 years  
Kregosłup ledzwiowy a-p + boczne  
AP

### Dane jednostki organizacyjnej

Dane jednostki organizacyjnej są implementowane przez *Sokar::HospitalDataIndicator*. Pojawiają się zawsze na scenie w prawym górnym rogu i zawierają następujące linie:

- Nazwa instytucji  
Tekst jest obierany z <sup>Dicom</sup> Tag Institutional Department Name (0x0008, 0x1040) i wyświetlany bez ingerencji.
- Producent wyposażenia wraz z modelem urządzenia  
Tekst jest obierany z <sup>Dicom</sup> Tag Manufacturer (0x0008, 0x0070) i <sup>Dicom</sup> Tag Model Name (0x0008, 0x1070), oddzielony spacją i wyświetlany bez ingerencji.
- Nazwisko lekarza wykonującego badanie  
Tekst jest obierany z <sup>Dicom</sup> Tag Referring Physician Name (0x0008, 0x0090) i wyświetlany bez ingerencji.
- Nazwisko operatora wspierającego badanie  
Tekst jest obierany z <sup>Dicom</sup> Tag Operators Name (0x0008, 0x1070) i wyświetlany bez ingerencji.

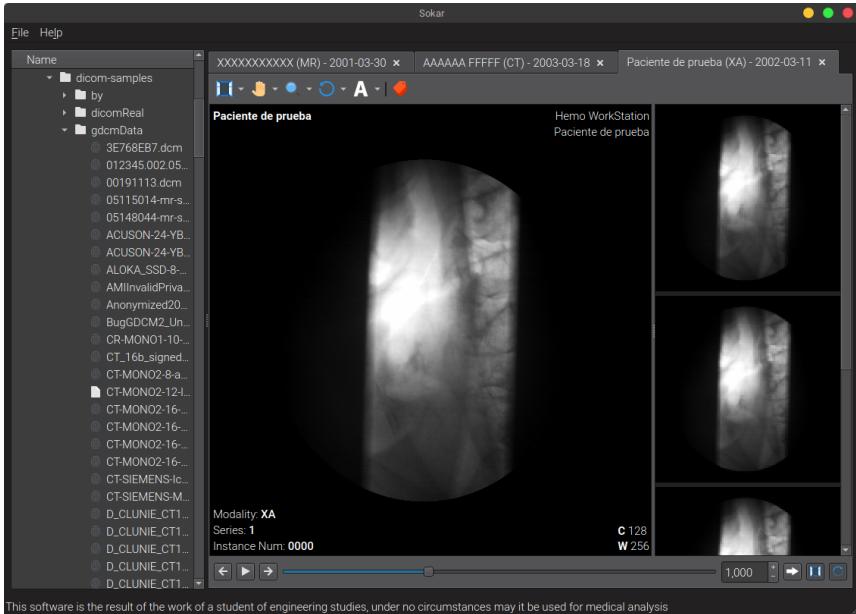
### Orientacja obrazu

Orientacja obrazu jest implementowana przez *Sokar::ImageOrientationIndicator*. Obiekt wyświetla cztery litery oznaczające orientację obrazu w stosunku do pacjenta. Obiekt posiada cztery pola: lewe, górne, prawe i dolne.

Każda z sześciu możliwych liter oznacza kierunek oraz zwrot w jakim jest ulożony pacjent:

- „R” — right — część prawa pacjenta,
- „L” — left — część lewa pacjenta,





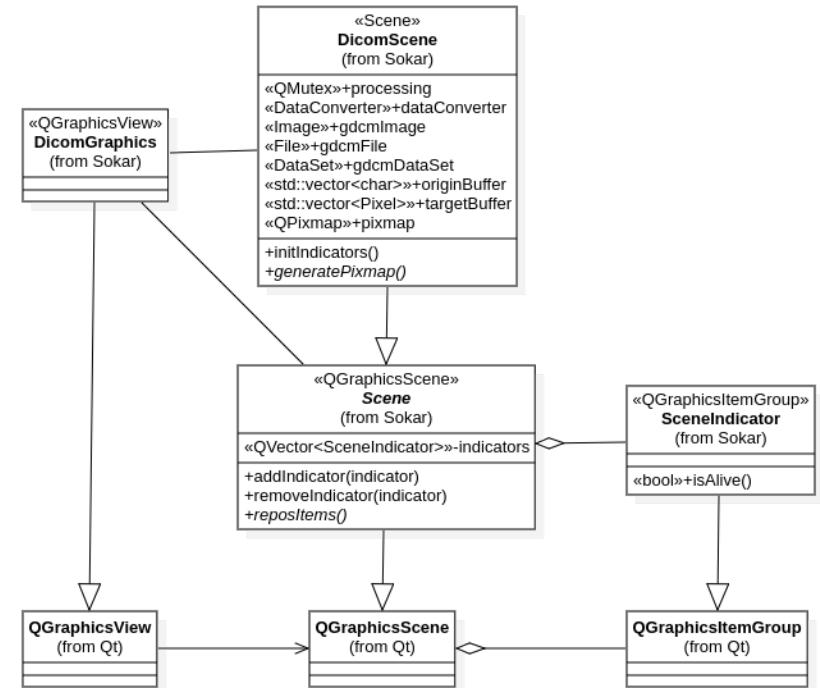
Rysunek 4.2: Okno przeglądarki z wczytanymi kilkoma obrazami. Zdjęcie własne.

Obiekt wewnętrzny zakładek odpowiada za wyświetlanie wszystkich elementów umożliwiających interakcję użytkownika z obrazem. Jest on implementowany przez klasę *Sokar::DicomView*. Jeden taki obiekt może posiadać wiele obrazów wyświetlanych w formie animacji. Obrazy są wyświetlane na scenie implementowanej przez *Sokar::DicomScene*. Pod sceną znajduje się pasek filmu. Z jego pomocą użytkownik może zatrzymać lub wznowić animację. Na prawo od sceny znajdują się ikony i z wszystkimi ramkami filmu. Pasek filmu i ikony obrazów ukrywają się, gdy jest wczytyany tylko jeden obraz.

Scena to obiekt wyświetlający i generujący obraz na ekranie. Dodatkowo na scenie znajdują się pięć zestawów informacji z pliku DICOM:

- dane pacjenta — w lewym górnym rogu,
- dane szpitala lub jednostki w, której obraz został wykonany — w prawym górnym rogu,
- dane akwizycji obrazów w lewym dolnym rogu, mogących się różnić dla każdej modalności,
- podziałka informująca o rzeczywistym rozmiarze obiektu znajdującego się na obrazie znajdująca się w dolnej i prawej części obrazu,
- cztery litery z sześciu (H, F, A, P, R, L) informujących o ułożeniu obrazu względem pacjenta.

Przykładowa scena z obrazem monochromatycznym znajduje się na rysunku 4.3.



Rysunek 4.6: Diagram klas UML dziedziczenia klasy *Sokar::DicomScene*.

#### 4.4 Projekt struktury obiektowej programu

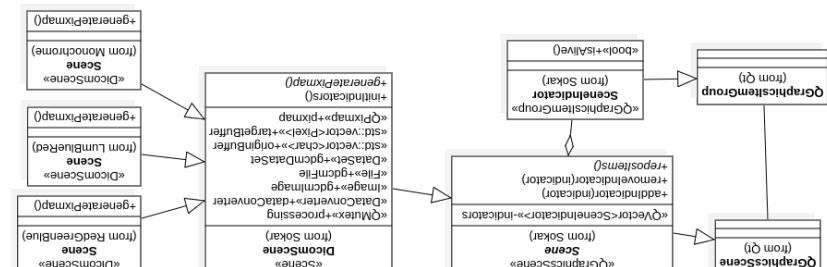
Moziwosć wyzwolona ammacyjna poszukana jest wtedy, gdy w edycji zakładek będzie zaznaczona sile typacji nizi jedna ramka obrazu. Móżna to osiągnąć wczystującą wiedzą o strukturze menu programu zaznajomionego z jego jasnym opisem w sekcji 4.5.6.

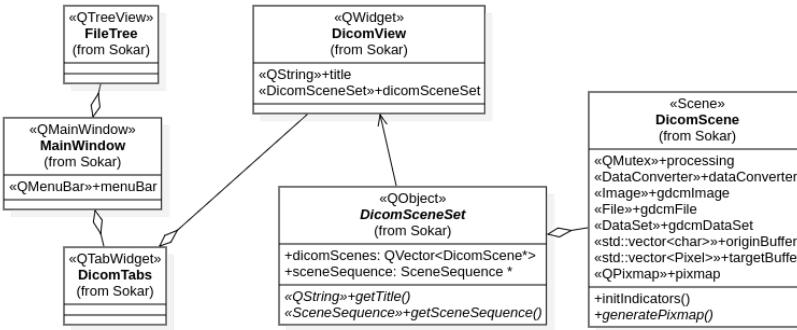
**Łyśnuk 4.3:** Przykładowa scena z obrazem monochromatycznym. Zdjęcie własne.



- *Otw: GraphicsImageItem* — element wyświetlający prostokąt z punktem A do B,
  - *Otw: GraphicsImageItem* — element wyświetlający prostokąt z punktem A do B,
  - *Otw: GraphicsImageGroup* — grupa obrazów graficznych, obiekt klasy
  - *Otw: GraphicsImageGroup* — grupa obrazów graficznych, obiekt klasy
  - *Otw: GraphicsImageGroup* — grupa obrazów graficznych, obiekt klasy

Rysunek 4.5: Diagram klas UML dziedziczenia klas *Sokar::DicomScene*.





Rysunek 4.4: Diagram klas UML globalnej struktury programu.

## 4.5 Struktury danych

### 4.5.1 Konwertowanie danych ze znaczników

Każdy plik DICOM posiada zbiór elementów danych. Zapisane elementy danych należy przekonwertować na obiekty danych odpowiadające potrzebom programu. Dlatego został zaimplementowany obiekt klasy *Sokar::DataConverter* zajmujący się konwersją danych z pliku DICOM na dane w formacie odpowiadającym programowi.

Obiekt konwertera jest tworzony na podstawie pliku DICOM i przy wywoływaniu konwersji należy podać tylko znacznik, który nas interesuje. Takie rozwiążanie pozwala na przesyłanie do wszystkich obiektów jednego względnie małego obiektu konwertera, co ułatwia zarządzanie dostępem do pliku DICOM.

Klasa *Sokar::DataConverter* posiada następujące funkcje, pozwalające na konwertowanie danych:

- *Sokar::DataConverter::toString()*

Funkcja konwertuje element na obiekt tekstu *Qt::QString*.

- *Sokar::DataConverter::toAttributeTag()*

Funkcja konwertuje element o znaczniku typu VR:AT na obiekt znacznika *gdcm::Tag*.

- *Sokar::DataConverter::toAgeString()*

Funkcja konwertuje element o znaczniku typu VR:AS na tekst w postaci czytelnej, np: „18 weeks” lub „3 years”.

- *Sokar::DataConverter::toDate()*

Funkcja konwertuje element o znacznik typu VR:DA na obiekt klasy *Qt::QDate*, który ma w sobie wbudowaną konwersję na tekst zależny od ustawień językowych aplikacji.

- *Sokar::DataConverter::toDecimalString()*

Funkcja konwertuje element o znacznik typu VR:DS na obiekt wektora posiadającego liczby rzeczywiste. *qreal* jest aliasem do typu zmiennoprzecinkowego, na systemach 64-bitowy jest to *double*.

- *Sokar::DataConverter::toIntegerString()*

Funkcja konwertuje element o znacznik typu VR:IS na 32-bitową liczbę całkowitą ( *qint32*).

- *Sokar::DataConverter::toPersonName()*

Funkcja konwertuje element o znacznik typu VR:PN na obiekt tekst zawierający imię w formie pisanej.

- *Sokar::DataConverter::toShort()*

Funkcja konwertuje element o znacznik typu VR:SS na 16-bitową liczbę całkowitą ze znakiem ( *qint16*).

- *Sokar::DataConverter::toUShort()*

Funkcja konwertuje element o znacznik typu VR:US na 16-bitową liczbę całkowitą bez znaku ( *quint16*).

Oprócz powyższych funkcji jest jeszcze kilka innych funkcji pomocniczych oraz kilka aliasów. Ogólne zasady konwersji, które dotyczą wszystkich danych:

- Większość VR jest zapisanych jako tekst, kodowanie i dekodowanie tekstu jest zapewniane przez bibliotekę.
- Większość danych może mieć kilka wartości oddzielonych backslashem „\”, dlatego konwerter dla VR, w których standard przewiduje wiele wartości, zawsze zwraca wektor z tymi wartościami.
- Wszystkie dane są zapisane parzystą ilością bajtów. W przypadku tekstu dodaje się znak spacji na końcu danych. Taka spacja jest pomijana w analizie danych.

### 4.5.2 Scena

Scena jest obiektem jednej ramki obrazu i jest odpowiedzialna za pośrednie wygenerowanie obrazu oraz jego wyświetlenie na ekranie. Implementowana jest ona przez klasę *Sokar::DicomScene*, dziedziczącą po *Sokar::Scene*, natomiast *Sokar::Scene* dziedziczy po *Qt::QGraphicsScene*. Diagram klas UML znajduje się na rysunku 4.5

#### Wyświetlanie sceny

Qt zapewnia własny silnik graficzny, który pozwala na łatwą wizualizację przedmiotów, z obsługą obrótu i powiększania. Silnik ten jest implementowany w postaci scen za pomocą *Qt::QGraphicsScene*. Natomiast klasa *Qt::QGraphicsView* dostarcza element interfejsu graficznego, który jest miejscem do wyświetlania scen.

Na scenie mogą być wyświetlane obiekty dziedziczące po *Qt::QGraphicsItem*. Obiekty te mogą być dodawane, usuwane i przesuwane ze sceny w czasie rzeczywistym. Dodatkowo