

Warszawa 2019

prof. nzw. dr hab. inż. Włodzimierz Smolik
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



POŁITECHNIKA WARSZAWSKA
WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMATYCZNEJ

Cross-platform DICOM image viewer in C++

Praca przedstawia Praca składa się z sześciu rozdziałów: wstęp, obrazowanie diagnostyczne, biblioteki i narzędzia, implementacja, kompilacja 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 kompilacji kodu źródłowego.

Słowa kluczowe: DICOM; przeglądarka DICOM; obrazy; obrazowanie; C++; Qt; GDCM; programowanie, przeglądarka obrazów medycznych, medyczne diagnostyczne techniki obrazowe

Wielopłatformowa przeglądarka obrazów DICOM w C++

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

Wieloplatformowa przeglądarka obrazów DICOM w C++

The second chapter describes the problem related to medical images. The diagnostic techniques and their basic differences are mentioned. The parameters of digital images and image browsers are explained. In addition, presentations of medical images and image recorders digital images, the libraries and tools used when writing emerging DICOM is described.

The third chapter describes the libraries and tools used when writing and implementing the work. The purpose of using the CMake tool and its advantages are explained. The Qt library, its possibilities, object trees implemented by it and the way of constructing event library as a programming language contained in it have been described. The selection of the GDIM library as a library for handling and loading of DICOM files has been presented and justified. The fourth chapter presents the method of work implemented. The expected range of implemented software functions has been specified. The graphical user interface and its program functions are described. The design of the object structure of the program has been explained. Then, the data structure with the C++ classes is described in detail. Where there was a possibility, a UML diagram is attached. All data processing algorithms are described for better visualization of the image.

The fifth chapter describes the compilation process of the source code. The fifth chapter describes the compilation process of the source code.

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

Widzomy odpowiadaliśmy karcie za skadariusze fakszywy chętnie osiągnąć, że in-
hięzja praca dypłomowa pozostała napisana przez mnie samodzielnie, pod opieką kierującej egzam-

OSWIADCZENIE

POLITECHNIKA WARSZAWSKA

Warszawa, 05 czerwca 2019

- mimesza praca dyplomowa nie narusza praw autorskich w rozumieniu istawy z dnia 4 lutego 1994 roku o prawie autorskim i prawaach pokrewnych (Dz.U. z 2006 r. Nr 90, poz. 44).

poz. 681 z pozna, zm.) oraz dobor osobiścią chlronionych prawem cywilnym, tzw. tarczami prawnymi i prawnymi pozwów (Dz. z. z 2007 r. poz. 100, poz. 681 z pozna, zm.)

- **utrigiszsa** praca dyplomowa nie była wczesnitą podstawą zarządu instytutu zatrudnionego w jednostce organizacyjnej z nadawaniem dyplomów lub tytułu zawodowego;

- Wszystkie informacje umieszczone w halięzszcej pracy, uzyskane ze zródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury opowiadającej o historii kultury polskiej, zgodnie z zasadami etykiety naukowej.

• Znacząca prawa i praktyka międzynarodowa w zakresie zasad公正のための国際法と実践
autorski i prawdziwi pokrewny, prawdziwi własneści przymysłowej oraz zasad公正のための国際法と実践
komercjalizacji.

OSWiadacza, ze ręce picie, dypłomowę w wersji kompaktowej, oraz trzecie picie dypłomowę w zawałowej na noszku elektrycznym (dypłomowę w wersji kompaktowej) oraz trzecie picie dypłomowę w module APD systemu SOS se identyfikuje.

Adam Jędrzejowski

- | | | |
|------|---|----|
| 7.2 | Nazwyzadane Linka w przekształceniach metodycznych DBOOM i DBOOM - nazwyzadane | 6 |
| 2.2 | Wyświetlenie wielu obrazów na raz w jednym oknie w przegladarce Sante DICOM Viewer 3D Pro. Zdjęcie użyte za zgoda Santessoft. | 10 |
| 2.3 | Generowanie obrazów 3D z obrazów tomograficznych w przegladarce Sante DICOM Viewer 3D Pro. Zdjęcie użyte za zgoda Santessoft. | 11 |
| 2.4 | Rekonstrukcji wielopłaszczyznowej w przegladarce Athena DICON Viever. | 11 |
| 2.5 | Zdjęcie użyte za zgoda Medical Harbour. Zdjęcie ze standardu DICON Elementy darych w zbiorze elementów darych. | 14 |
| 3.1 | Przykładowe okienko programu w Qt. Zdjęcie własne. | 23 |
| 4.1 | Okno przegladarki tuz po uruchomieniu. Zdjęcie własne. | 29 |
| 4.2 | Okno przegladarki tuz po uruchomieniu. Zdjęcie własne. | 30 |
| 4.3 | Przykładowa sekcja z obrazem monochromatycznym. Zdjęcie własne. | 31 |
| 4.4 | Przykładowa sekcja z wczesnymi kikkoma obrazami. Zdjęcie własne. | 32 |
| 4.5 | Dyagram klas UML dziedziczenia klas Szkoly:DicomScene. | 34 |
| 4.6 | Dyagram klas UML dziedziczenia klas Szkoly:DicomScene. | 35 |
| 4.7 | Dyagram klas UML dziedziczenia klas Szkoly:DicomScene. | 36 |
| 4.8 | Dyagram klas UML dziedziczenia klas Szkoly:DicomScene. | 40 |
| 4.9 | Wygadanki wraz z umiarem elmentow interfejsu. Zdjęcie własne. | 42 |
| 4.10 | Powtarzanie jednego obrazu w tzw.ach oznaczeniu (po prawej) Zdjęcie własne. | 50 |
| 4.11 | Plata Hot Iron (na środku) i Hot Metal Ble (po prawej) w porównaniu do pełni w skali szarości (po lewej). Zdjęcie własne. | 51 |
| 4.12 | Wizualizacja układu osi współrzędnych kartezjańskich pasekta. Zdjęcie własne. | 59 |
| 4.13 | Podział Phaszczyny secy. Wyświetlono osiem części. Zdjęcie własne. | 62 |
| 4.14 | Przykładowy obraz medyczny (przekroj gęstościowy MR) z oznaczeniem orientacyjnym. Zdjęcie własne. | 63 |

Rozdział 6

Podsumowanie

Celem pracy inżynierskiej było napisanie aplikacji do obsługi obrazów DICOM w C++ z możliwością kompilacji na wiele platform. Cel udało się osiągnąć. Zniesienie ograniczeń wirtualizacji kodu rozwiązaño językiem programowania C++. Zastosowano biblioteki dostępne na różnych platformach: Qt i GDPCM, które również zostały napisane w C++, dzięki czemu uzyskano jednolity program napisany w jednym języku. Zapewniono jednolity sposób kompilacji na platformach przy użyciu narzędzia CMake, dzięki czemu 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 kolorowe 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 GDPCM 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 kompilując bibliotekę GDPCM własnoręcznie.

Spis treści

1	1 Wstęp
3	2 Obrzutowanie diagnostyczne w medycynie
5	2.2 Parametry obrzutow
5	2.2.1 Podstawowe parametry obrzaru cyfrowego
5	2.2.2 Kontраст
7	2.2.3 Rozdzielcość
8	2.2.4 Stosunek sygnału do szumu (SNR)
8	2.2.5 Poziom artefaktów
8	2.2.6 Poziom zniekształceń przestrzennych
8	2.3 Prezentacja obrzazu medycynicz
8	2.3.1 Przegądarki obrzazu
8	2.3.2 Funkcje przegądarki obrzazu
8	2.3.3 Kryteria porównywania przegądarki obrzazu
12	2.4 Format cyfrowych obrzazów medycynicz
13	2.4.1 Standard DICOM V3.0
14	2.4.2 Spodób zapisu danych w pliku DICOM
17	2.4.3 DICOMDIR
17	2.4.4 Imię formaty zapisu
18	3.1 CMak
18	3.2 QT
19	3.2.1 Wyświetlacz
19	3.2.2 Licencja
19	3.2.3 Normy i certyfikaty
19	3.2.4 Globalne typy struktury
20	3.2.5 Klasa QObject
22	3.2.6 Grzedy i interfejsy użytkownika
23	3.3 GDCM
23	3.3.1 Uzasadnienie wyboru
24	3.3.2 Opis
24	3.3.3 Licencja
24	3.3.4 Podstawowe klasy
25	3.3.5 Przykłady użycia
27	3.4 Git

- W przypadku Linuksa i MacOS, plik binarny znajduje się w folderze „/path/sokar-app-bin/Debug”.
- W przypadku Linuksa i MacOS, plik binarny znajduje się w katalogu „/path/sokar-app-bin/”.
- Wszystkie parametry „QtDir” na skrzynce do skompilowania biblioteki Qt.
- Kliknij „Finish”
- W pole „Where is the source code” wpisz „/path/sokar-app/”
- W pole „Where to build the binaries” wpisz „/path/sokar-app-bin/”
- uruchom CMak z menu programu lub za pomocą polecenia „cmake-gui”
- Kompilacja zawiera polecenia, które należą się do naszego projektu i kolejności:

5.5 Kompilacja Sokar

- w polu „Where is the source code” wpisz „/path/gdcm-bin/GDCM”
- w oknie zatytuowanym „CMakSetup” wybierz takie same opcje na w GDCM
- Kliknij „Configure”
- w polu „Where is the source code” wpisz „/path/sokar-app/”
- uruchom CMak z menu programu lub za pomocą polecenia „cmake-gui”
- Kompilacja zawiera polecenia, które należą się do naszego projektu i kolejności:

- w przypadku Linuksa i MacOS, uruchom „make” w folderze „/path/gdcm-bin/”.
- zas w przypadku Windowsa otwórz plik „/path/gdcm-bin/Sokar.sln” i kliknij przycisk „Zas” w przypadku Linuksa i MacOS, uruchom „make” w folderze „/path/sokar-app-bin/”.
- nastepnie kliknij przycisk „Generate”
- ustaw parametry wartości „QtDir” na skrzynce do skompilowania biblioteki Qt.
- Kliknij „Finish”
- w oknie zatytuowanym „CMakSetup” wybierz takie same opcje na w GDCM
- w oknie zatytuowanym „CMakSetup” wybierz takie same opcje na w GDCM
- zas w przypadku Linuksa i MacOS, uruchom „make” w folderze „/path/gdcm-bin/”.
- W przypadku Linuksa i MacOS, plik binarny znajduje się w folderze „/path/sokar-app-bin/Debug”.
- W przypadku Linuksa i MacOS, plik binarny znajduje się w katalogu „/path/sokar-app-bin/”.
- Móżna go uruchomić go klikającą lub z terminala za pomocą komendy „./Sokar”. W przy-
- badku Windowsa plik binarny znajduje się w folderze „/path/sokar-app-bin/Debug”.

5.6 Uruchomienie

4 Implementacja	28
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	56
4.6.4 Ustalanie pozycji pacjenta względem sceny	59
5 Kompilacja	64
5.1 Narzędzia potrzebne do kompilacji	64
5.2 Biblioteki potrzebne do kompilacji	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
6 Podsumowanie	67

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 odhaczyć wszystkie inne opcje, nie są one wymagane do kompilacji 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”

Wstep Rozdzial 1

5.1 Narzézia Potrzébne do kompliacji

- Windows — Visual Studio w wersji 2017 lub nowszej
 - MacOS — Xcode w wersji 10 lub nowszej
 - Linux — pakietu zawierające następujące komendy: make, cmake (w wersji 3.10 lub nowszej), g++ (w wersji 8 lub nowszej)
 - operacyjnego:

5.2 Biblioteki potrzebne do kompilacji

Kod zrozumiejszy rozdział skomplikowany za pomocą powzyszyższych narzędzi. W kodzie programu nie występują żadne elementy odlegające od standardu C++17, więc nie powinno być problemów z użyciem innych wersji biblioteki standard C++17.

5.2.1 Instalacija Qt

Program byt testowany na wersji 3.12, z tego powodu zalecamy ją jesi uzywanie tesi wersji.

Windows i Mac OS

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 spowodowały, 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 zostały 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, kompilowanego 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.

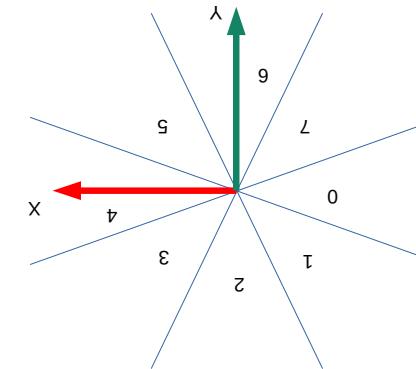
W opracowanym kodzie przeglądarki obrazów do obsługi plików w formacie DICOM wykorzystano bibliotekę Grassroots (Grassroots DICOM library — GDCM).



Rysunek 4.14: Przykładowy obraz medyczny (przekrój głowy MR) z oznaczeniem orientacji obrazu za pomocą liter A, P, R, L, F, H. Zdjęcie własne.

Obrazowanie diagnostyczne w medycynie

Rysunek 4.13: Podział płaszczyzny sceny. Wyroznione osiem części. Zdjēcie własne.



Praktické představování dat je využíváno v různých oborech, například v marketingu, ekonomice, sociologii, psychologii a dalších. Využívá se k efektivnímu prezentování informací, k jejich lepšímu pochopení a snadnějšímu rozumění. Využívá se také v různých profesích, kde je potřeba efektivně prezentovat informace, například v oboru marketingu, kde je potřeba efektivně prezentovat produkty a služby, nebo v oboru ekonomice, kde je potřeba efektivně prezentovat finanční údaje a trendy. Využívá se také v oboru sociologie, kde je potřeba efektivně prezentovat sociální údaje a tendence, nebo v oboru psychologie, kde je potřeba efektivně prezentovat psychické údaje a tendence. Využívá se také v oboru marketingu, kde je potřeba efektivně prezentovat produkty a služby, nebo v oboru ekonomice, kde je potřeba efektivně prezentovat finanční údaje a trendy. Využívá se také v oboru sociologie, kde je potřeba efektivně prezentovat sociální údaje a tendence, nebo v oboru psychologie, kde je potřeba efektivně prezentovat psychické údaje a tendence.

- lewe pole: tytuł częsci 1, tytuł częsci 0 i tytuł częsci 1
 - górnego pole: tytuł częsci 1, tytuł częsci 2 i tytuł częsci 3
 - prawego pole: tytuł częsci 3, tytuł częsci 4 i tytuł częsci 5
 - dolne pole: tytuł częsci 7, tytuł częsci 6 i tytuł częsci 5

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 fali ultradźwiękowej pod różnymi kątami. Z każdego cyklu jest tworzona jedna linia, obraz jest tworzony z wielu linii, które następnie są układane pod różnymi kątami, odpowiadającym ich rzeczywistemu ułożeniu na głowicy. Wielkość obrazu jest zależna od algorytmu rekonstrukcji i jest z góry ustawiona 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 radiofarmaceutyku, czyli związków chemicznego zawierającego izotop promieniotwórczy. Charakteryzuje się on krótkim czasem rozpadu i powinowactwem chemicznym z badanymi organami. Wykrywa się rozpad zachodzący w ciele

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*:

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

Powyzsza macierz różni się od macierzy definiowanej w standardzie. Po pierwsze wartości z _{Dicom} Pixel Spacing (0x0028, 0x0030) zostały pominięte, a nadano im wartość 1. Po drugie - pozycja z _{Tag} _{Dicom} Image Position (0x0020, 0x0032) została zrównana do punktu zerowego, dzięki temu, wynik też będzie względem punktu zero. Wyznaczenie macierzy *imgMatrix* jest jednorazowe.

Po wyznaczeniu sześciu punktów *ScenePosition* dla każdego punktu *PatientPosition*, są one zapisywane. *ScenePosition* odpowiada pozycji punktów na obrazie w pozycji startowej.

Na scenie, na której jest wyświetlany obraz, użytkownik może obracać obraz o dowolny kąt, według własnego uznania. Te przekształcenia są realizowane za pomocą macierzy rotacji, dalej zwana jako *rotateTransform*. Macierz *rotateTransform* jest przesyłana do naszego obiektu *Sokar::ImageOrientationIndicator* za każdym razem, kiedy zostanie zmieniona.

Ostateczne wyznaczenie pozycji punktów pacjenta na obrazie odbywa się przez przemnożenie lewostronne *rotateTransform* i *ScenePosition*.

$$rotateTransform * ScenePosition$$

Wyznaczana jest w ten sposób pozycja sześciu punktów pacjenta na płaszczyźnie sceny wyświetlanej. Następnie określany jest, na której z ośmiu części płaszczyzny jest umieszczony dany punkt. Podział płaszczyzny jest widoczny na rysunku 4.13. Tej płaszczyźnie nadawany jest tytuł w postaci litery, która oznacza stronę pacjenta. Jeżeli punkt znajduje się w centrum, na przecięciu osi, to oznacza, że punkt znajduje się za lub przed ekranem, więc jest pomijany. Następnie do czterech pól wyświetlających zostają wstawione następujące teksty:

- “H” — [0, 0, +1, 1]
 - “F” — [0, 0, -1, 1]
 - “P” — [0, +1, 0, 1]
 - “A” — [0, -1, 0, 1]
 - “T” — [+1, 0, 0, 1]
 - “R” — [-1, 0, 0, 1]

Standard DICOM nazwywane są techniką obrazowania modalności (ang. modality).

- PET-MRI — połączenie PET z rezonansem magnetycznym
 - PET-CT — połączenie PET z wizjorizedowym tomografem kątowym

W standardzie DICOM obraz jest ozaczana jako „P1.1”, a w szczegółach badania fizyczne w sobie rożne techniki, takie jak:

W standardzie DICOM obraz jest oznaczany jako „PT”, detektorów.

- Tomografia SPECI

W standardzie DICOM obraz szkutyczny jest oznaczany jako „NM”.

2.2 Parametry obrázku

2.2.1 Podstawowe parametry obrazu cyfrowego

Wyszczególnione pozytywne cechy osobowe, takie jak empatia, zdolność do komunikacji, zdolność do zarządzania emocjami i zdolność do podejmowania decyzji, są kluczowe dla efektywnego prowadzenia rozmów z pacjentem. Wprowadzenie nowego terminu, takiego jak "pacjent w pozycji pozytywnej", może skutkować zmianą podejścia do rozmów z pacjentem, co może prowadzić do lepszych wyników leczenia.

Wyznačanie pozycji pacjenta

- P_{xyz}^{xyz} — koordynaty wokseli (i, j) we współrzędnych obrazu wyróżone w milimetrach
 - S_{xyz}^{xyz} — trzy wartości z elementu ze znacznika $Image$ $Position$ ($0x0020$, $0x0032$). Oznacza on punkt pozycji pacjenta wyróżony w milimetrach w stosunku do $0x0020$, $0x0032$).
 - X_{xyz}^{xyz} — trzy wartości warstwice znacznika $Image$ $Orientation$ ($0x0020$, $0x0037$)
 - Y_{xyz}^{xyz} — trzy wartości warstwice znacznika $Image$ $Orientation$ ($0x0020$, $0x0037$)
 - i, j — oznaczają współrzędne na macierzy obrazu, odpowiadające kolumnie i wiersz.
 - Δ_i — rzeczywista wielkość pikseli obrazu wyróżona w milimetrach. W algorytmie wyznaczania strojny Pacjenta ta wartość może wynosić 1, ponieważ odpowiada za skalę.
 - Praktyczne rzeczywiste biorąc, pierwsza macierz to wektor reprezentujący pozycję pacjenta, druga jest to przekształcenie macierzy we współrzędnych jednorodnych, trzecia to przekształcenie macierzy we współrzędnych jednorodnych, dziesiąta do przekształcania strojny Pacjenta ta wartość może wynosić 1, ponieważ odpowiada za pozycję na obrzędzie.

Wartości **Tag** Image Orientation (0x0020, 0x0033) skladaja sie z sześciu liczb, odporowiadających dla których wartości X_x, X_y, X_z, Y_x, Y_y, Y_z. Standard DICOM definiuje nastepujacych sposob interpretowania datyrych:

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

^{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 ^{Dicom} Rows (0x0028, 0x0010)) oraz szerokość (zapisaną w ^{Dicom} Columns (0x0028, 0x0011)). Do poprawnej interpretacji znaczenia macierzy służy znacznik ^{Dicom} 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.

Połączenie macierzy

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

```
1 QTransform DicomScene::getPixmapTransformation() {
2     QTransform 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:

*panTransform * rotateTransform * scaleTransform * centerTransform*

4.6.4 Ustalanie pozycji pacjenta względem sceny

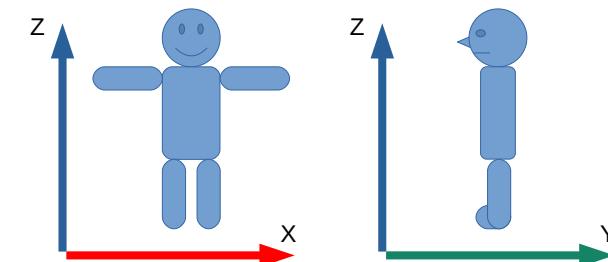
W obrazie DICOM jest pośrednio zapisana informacja o ułożeniu obrazu względem pacjenta. Celem algorytmu jest określenie jaką pozycję przyjmuje pacjent w stosunku do sceny tak, aby można było wyświetlić tą pozycję na scenie.

Format zapisu informacji o orientacji obrazu

Informacje o orientacji oraz pozycji względem pacjenta znajdują się w odpowiednio w tagach ^{Dicom} Tag Image Orientation (0x0020, 0x0037) i ^{Dicom} Tag Image Position (0x0020, 0x0032).

Standard DICOM zdefiniował ułożenie osi we współrzędnych kartezjańskich następująco:

- „x” — osz przechodząca od prawej do lewej strony pacjenta, „L” oznacza zwrot zgodny z osią, a „R” oznacza zwrot przeciwny
- „y” — osz przechodząca od przodu do tyłu pacjenta, „P” oznacza zwrot zgodny z osią, a „A” oznacza zwrot przeciwny
- „z” — osz przechodząca od dołu do góry pacjenta, „H” oznacza zwrot zgodny z osią, a „F” oznacza zwrot przeciwny



Rysunek 4.12: Wizualizacja układu osi współrzędnych kartezjańskich pacjenta. Zdjęcie własne.

szasowa definiowana jako odległość w czasie od dźwihu klatek obrzutowania. Pomiar dokonywany w określonym czasie i ustalone za pomocą MTR, kiedy many rozdrożlowane szasowa jest istotna w obrazach dynamicznych, np. angiogramie, badania kach wrazie że rowne zmiany zachodzące w organizmie w czasie wykonywania kaczej pomiaru wykazują pewnego czasu potoczą danych. W metrycznych przed- czasowa definiowana jako odległość w czasie od dźwihu klatek obrzutowania.

Czasowa

každej techniky.

Przeźwiona

2.2.3 Rozdziały z ośc

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

$$\frac{u_{im}I + x_{im}I}{u_{im}I - x_{im}I}$$

Jechnika z wilejú dechnicji kontrastu jest kontrast Michelsoна wyrażony wzorem:

2.2.2 Kontrast

Orzaz DICOUM i powtórzył zasady, w siedzi biuromiącego o problemie. Z wągą na to, że problematyczne wygryźała mazury w kązdej technice, standard postulada odpowiedni zestaw zaznaczników dla kązdej techniki. Próbkowania nie poszczególnych technik jest opisane w sekci

- Tag Bits Allocated (0x0028, 0x0100) — informuje jak wiele bitów zostało zaallo-
wanych do zapisania jednego piksela
 - Tag Bits Stored (0x0028, 0x0101) — informuje jak wiele bitów zostało zaallo-
wanych do zapisania jednego piksela
 - Tag Bits Written (0x0028, 0x0102) — informuje jak wiele bitów zostało zaallo-
wanych do zapisania jednego piksela
 - Tag High Bit (0x0028, 0x0102) — informuje gdzie znajduje się najwyższy bit
Decom Pixel Representation (0x0028, 0x0103) — informuje czy pozycyjny są ze znakiem
czynnego

Kwahyżaca obrazu, czyli liczba pozycji w obrazu, jest zapisana na czerwach znaczni-

- **Windrowing** — stan okienkowania, odsługiwanego przez Sokoła: *Monochromie: Scene* Do obiektu okienka siedzi wysiąlane zmarły porzeczek Sokoł: *Windrow: Wiadomość: Sokoł* (al) z parametrem Δy i *Sokar*: *Windrow: mAhorzontal* z parametrem Δx . Następnie powróci jesteś generowalny obraz z użyciem zmarły okienka.

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

$$rotate = rotate + \Delta x * 0.1;$$

$$rotate = rotate + \Delta y * 0.5;$$

• *rotate* = 0

rotate — stran rotacji, odbiegający przed soratn: uchomiszeńce
Na makiety rotacji wywoływaną jest funkcja rotacji `Qt::Transform::rotate()` z parametrem rotacie wykazującym podatym wzorem:

- **Rotate** — stan rotacji, obstygivany prez Sokar::DicomScene

Sprawia to, że ruch Piłsudskiego jest bardziej czysty na zasadzie niż ruch pozornym. Tego-tyczne jest mówiącże implementacji określonego skarłowskiego w dwoch osiach, jednakże jest to niewielkie wyzwanie.

scale = *scale* - $\Delta x * 0.001$

scale = scale - $\Delta y * 0.01$

scale = 1

Na matrierczy skaliowaniu wywoływana jest funkcja skalowania $Q_k := Q_k \text{Transform}::scale()$ z parametrem $scale$ wyliczanym podanym wzorem:

- SARAH SKARLOWSKA, DESIGNING WILLY PIZZA SOUP: A DESIGN MASTERS 2009

Na niewiększy przeszumowanie wywoływanie jest unikalna przeszumienie (rys. 11.17a). Transfornator:

- **Pan** — stan przeszuwania, odsztygniwanego przez Sokoła; **DicomSzeen**

Zmiany poprzecz obstruże myzki

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.

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

Powyższe przekształcenie macierze 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 wyśrodkowującej) 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.

Zmiany poprzez oderanie sygnału

Na pasku zadań, nad sceną znajduje się szereg przycisków, które po wciśnięciu wysyłają sygnał do obecnej sceny poprzez obiekt klasy *Sokar::DicomView*. Sposób wysyłania sygnałów jest szerzej opisany w sekcji 4.5.4.

Po otrzymaniu odpowiedniego sygnału jest wykonywana operacja na macierzy. Odbiór wszystkich sygnałów jest implementowany przez wirtualną funkcję *Sokar::DicomScene::toolBarActionSlot()*, która jest slotem.

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

- **ClearPan** — przywraca macierz przesunięcia do stanu zerowego
- **Fit2Screen** — przywraca macierz skali do stanu zerowego, następnie wylicza nową skalę w zależności od wymiarów obrazu i sceny
- **OriginalResolution** — przywraca macierz skali do stanu zerowego
- **RotateRight90** — na macierzy rotacji zostaje użyta funkcja *Qt::QTransform::rotate()* z parametrem 90.
- **RotateLeft90** — na macierzy rotacji zostaje użyta funkcja *Qt::QTransform::rotate()* z parametrem -90.
- **FlipHorizontal** — na macierzy rotacji zostaje użyta funkcja *Qt::QTransform::scale()* z parametrami 1 i -1.
- **FlipVertical** — na macierzy rotacji zostaje użyta funkcja *Qt::QTransform::scale()* z parametrami -1 i 1.
- **ClearRotate** — przywraca macierz rotacji do stanu zerowego

Po jakiekolwiek zmianie macierzy jest wywoływana funkcja *Sokar::DicomScene::updatePixmapTransformation()*, która odświeża macierz przekształcenia na obiekcie *pixmapItem*.

- Odnisząc DŁOŃDZIĘ, zaczęto rozwijać swoje zdolności fizyczne. Wyswierczenie struktury serii badan. Plik DICOMDIR to wilec zindeksowany, który zawiera jacyk zbiór elementów danych, bez obrazów.

Obstuga wielu plików

- Maski (ang. overlay). Jest to mozaikowosc nazozemna maski, elementu, który bedzie pokazywac obiekty, np. tla. Standard DICOM mozaikowa nazozemna wiele mask na jednen obraz.

- Okienkowalne. Termin oznacza się do używania funkcji okna użytkownego w celu zarządzania parametrow w celu lepszej interakcji z użytkownikiem. Wszystkie okienkowalne mają jedno i samo zadanie, a mianowicie to, aby dodać, zmodyfikować, usunąć, czy zaktualizować dane.

Wyśmienite ZLi: Natychmiast Lupa w Przejętadarcie Mediatora DCOM Viewer. Zdjedzie uzytkownika Soltuite UAB.

- Rotacja i odcięta lustrzana, czyli moźliwosc obrócenia obrazu o 90stopni kąt oraz moźliwosc uzytkownia odcięta lustrzana do obrócenia obrazu w dwoch ośach X i Y.



- Przykład użycia takiego nazwiska zaznaczonego powiększenia odrazu. Lupa, składowanie miedziane. Jest to mazwiowe miedziane powiększenie odrazu.
 - Przesuwając (ang. *pan*), zazwyczaj mazwiowe przesuwania odrazu do takiego stopnia, że nie będzie miedziane ekranie lub w okienku programu. Przykład użycia nazwiska zaznaczonego powiększenia odrazu. Lupa, składowanie miedziane. Jest to mazwiowe miedziane powiększenie odrazu do takiego stopnia, że nie będzie miedziane ekranie lub w okienku programu.

Przykład dzia³ania:
::scale().

Współzędne jednorodne dechnujące się jako spłoszki przedstawiały punktowu -wyrażonej przestrzeni rzutowej za pomocą ukladu $n + 1$ wspołrzędnych. W bibliotece Qt jedna z implementacji wspołrzędnych jednorodnych jest klasa `Qt::QTransform`. Implementacja ta pozwala na zastosowanie zachowania maticzny 3 na 3, jak również wykonywanie operacji jak: przesuwanie implementowanej przez `Qt::QTransform::translate()`, obrót implementowanej przez funkcję `Qt::QTransform::rotate()` i skalowanie implementowane przez `Qt::QTransform::scale()`.

4.6.3 Izwarcenie transformatora i ich użycie na obrazie

Współrzędne jednorodne

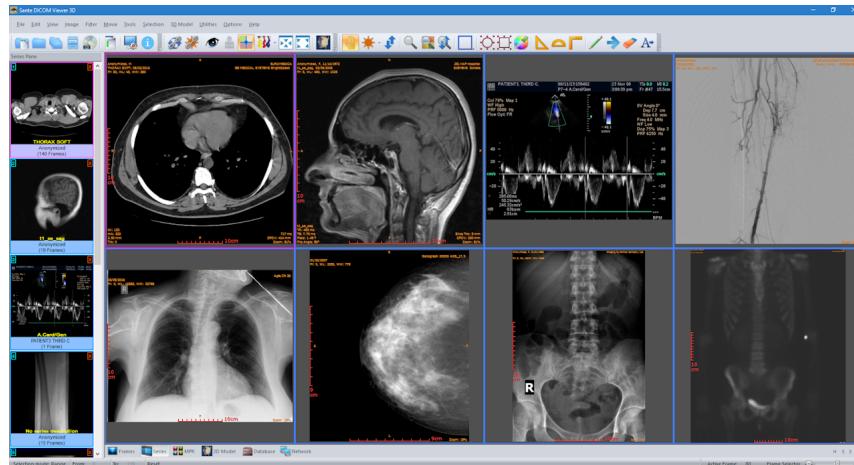
Waves Software: Platforma prezentująca preferencje użytkownika do przesyłania danych do serwera. Platforma pozwala na zarządzanie preferencjami użytkownika, dodawanie nowych preferencji, modyfikowanie istniejących i usuwanie nieaktualnych. Platforma posiada prosty interfejs użytkownika, który pozwala na łatwe zarządzanie danymi.

```
Palette
{
    public void convertFromImage( ImageTypeFormatException e )
    {
        if ( e instanceof ImageFormatNotSupportedException )
        {
            throw new ImageFormatNotSupportedException( "Image format not supported" );
        }
        else if ( e instanceof ImageTypeFormatException )
        {
            throw new ImageTypeFormatException( "Image type format not supported" );
        }
        else
        {
            throw new ImageTypeFormatException( "Image type format not supported" );
        }
    }
}
```

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 wokselu są dostępne to istnieje możliwość wygenerowania nowego obrazu, który byłby przekrojem poprzecznym.

Przykład generowania rekonstrukcji wielopłaszczyznowej jest pokazany na rysunku 2.4

```

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()*

Jest to funkcja pomocnicza ustalająca obecną klasę obecnego „okna”, aby móc wykonać funkcje *Sokar::Monochrome::Scene::genQPixmapOfTypeWidthWindow()*. Kod funkcji:

```

1 template<typename IntType>
2 void Monochrome::Scene::genQPixmapOfType() {
3
4     switch (getCurrentWindow()->type()) {
5         case Window::IntDynamic:
6             genQPixmapOfTypeWidthWindow<IntType, WindowIntDynamic>();
7             break;
8
9         case Window::IntStatic:
10            genQPixmapOfTypeWidthWindow<IntType, WindowIntStatic>();
11            break;
12
13         default:
14             throw WrongScopeException(__FILE__, __LINE__);
15     }
16 }
```

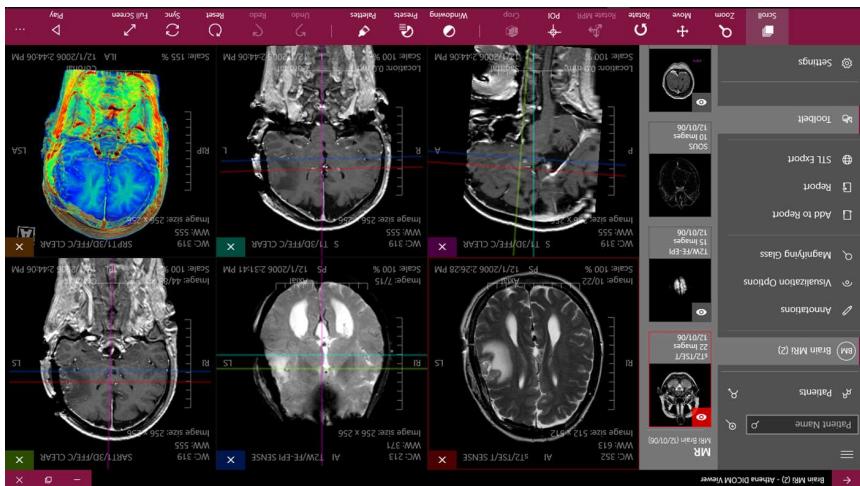
- *Sokar::Monochrome::Scene::generatePixmap()*

Funkcja odświeża okienko i sprawdza, czy odświeżenie obrazu jest konieczne, następnie sprawdza typ liczby wokselu i uruchamia *Sokar::Monochrome::Scene::genQPixmapOfType()*. Kod funkcji:

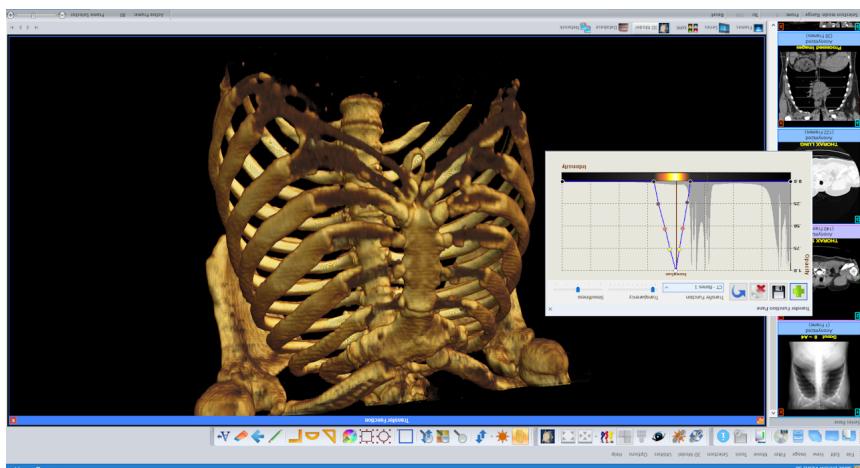
```

1 bool Monochrome::Scene::generatePixmap() {
2
3     /* Odświeżamy okno i sprawdzamy czy odświeżenie obrazu jest konieczne */
4     getCurrentWindow()->genLUT();
5     if (lastPixmapChange >= getCurrentWindow()->getLastChange()) return false;
6
7     /* Sprawdzamy typ liczby wokselu obrazu */
8     switch (gdcmImage.GetPixelFormat()) {
9         case gdcm::PixelFormat::INT8:
10            genQPixmapOfType<qint8>();
11            break;
12        case gdcm::PixelFormat::UINT8:
13            genQPixmapOfType<quint8>();
14            break;
15        case gdcm::PixelFormat::INT16:
16            genQPixmapOfType<qint16>();
17            break;
18        case gdcm::PixelFormat::UINT16:
19            genQPixmapOfType<quint16>();
20            break;
21    }
22 }
```

Rysunek 2.4: Rekonstrukcja wlejopłaszczowni w prześladowce Athena DICOM Viewer. Zdjęcie uzyskane za zgodą Medical Harbor.



Rysunek 2.3: Generowane obrazów 3D z wektora obrazów tomograficznych w przegłębarcie Samte DICOM Viewer 3D Pro



ještě to funkční, když držíte obraz na wptak, tvoří je i mnohem. Lze se vztahovat jen na nezáležitá za pořadí funkci Q (třídu `idealTreeCount()`). Všechny držítky zahrnují odrážecí funkci `isLocalityPrize` ilože wptak. Když funkci:

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

```

12         *bufferr++ = windowptr->getPixel(*origin);
11     /* tylem mlecsou jest dokonywana zaziana liczby na kolor */
10     for (quint64 i = from; i < to; i++) out64[i] = {
9         out64[i] += from;
8     }
7
6     auto windowptr = (Window *) getCurrenWindow();
5     auto out64 = (inttype *) borghbuffer[0];
4     auto buffer = #tagetbuffer[from];
3     template <typename IntType, typename Window>
2 void Monochome::Scene::genPmixmapDyPwidhWindowThread(quint64 from, quint64 to)
1 template <typename IntType, typename Window>
0 hasstypnych punktach. Kod funkcji:

```

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

```
    int line const Pixel &getPixel(qunit64 value) override {
    return *(pixelarray + signedmove + value); }
```

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

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.

Implementacja statyczna z tablicą LUT

W wersji z LUT, podczas tworzenia okienka alokowany jest wektor obiektów *Sokar::Pixel* klasy *std::vector*. Standard DICOM przewiduje, że wokseli mogą mieć wartości ujemne, więc tablica powinna mieć możliwość posiadania takich wartości indeksów, ale C++ nie przewiduje takiej możliwości. Dlatego wprowadzono dwie zmienne pomocnicze *maxValue* i *signedMove*. Zmienna *maxValue* jest to maksymalna wartość jaką dane mogą przyjąć, jest ona równa 2^N , gdzie N to liczba bitów brana z Tag BitsStored (0x0028, 0x0101). A *signedMove* to liczba przesunięcia liczb, przyjmuje wartość zero, gdy dane wokseli są całkowite nieujemne lub wartość przeciwną do *maxValue*, gdy wokseli są ujemne. Długość wektora pikseli jest sumą *maxValue* i *signedMove*. A indeks wokselu w wektorze ma wartość tego woksela zwiększoną o *signedMove*.

Wypełnienie wektora wartościami odbywa się poprzez iteracje po wszystkich możliwych wartościach, przeliczenie ich przez funkcje „okna”, a następnie wstawienie ich do wektora. W celu poprawy szybkości zastosowano sprawdzanie czy wartości są w zakresie „okna”. Poniżej kod funkcji:

```
1 bool genLUT() override {
2     if (WindowInt::genLUT()) {
3
4         /* Przeskalowanie wektora, gdy jest to wymagane */
5         if (arraySize != signedMove + maxValue) {
6             arraySize = signedMove + maxValue;
7             arrayVector.resize(arraySize);
8         }
9
10        /* Wyliczenie najmniejszej wartości */
11        qreal x = qreal(signedMove) * -1;
12
13        auto &background = isInversed() ? palette->getForeground() : palette->
14            getBackground();
15        auto &foreground = isInversed() ? palette->getBackground() : palette->
16            getForeground();
17
17        /* Iteracja */
18        pixelArray = &arrayVector[0];
19        for (int i = 0; i <= arraySize; i++) {
20
21            if (x < x0) {
22                *pixelArray = background;
23            } else if (x > x1) {
24                *pixelArray = foreground;
25            } else {
```

2.4.1 Standard DICOM v3.0

Przewszę komputerowy komputerowy przeszły swoj rok w latach siedemdziesiątych ubiegłego wieku. Oraz myślę że nie byłby bezpośrednim wynikiem badania, a jedynie wynikiem obrotka dalszych pomiarów przesz komputer. Wyczajne pliki gryczne (jak np. jpg, gif), nie nadawały się do zapisu takich obrazów, ponieważ zapisywały obraz w postaci sklejonych RGB. Kazdy producent stosował

2.4 Format cytrowycz obrazów medycznych

Obrzowanie trójwymiarowe

Telekaji Solar: Monochrome Webcam with a 640x480 resolution and a 120° wide-angle lens.

$\mathbf{1}x * v - \mathbf{1}f = 0$

$$({}^0x - {}^1x)/({}^0h - {}^1h) = 2$$

Posiadamy teraz dwa punkty określające oś oszczędzania się do warstw obrazu. Wyznaczono parametry prostej przedziałowej przesz dwa punkty:

$ce1/ = rescaleSlope$

$z_0/ = rescaleSlope$

${}^0 - \equiv \text{rescaleIntercept}$

$$AS = m/(q - s) \# Output Units$$

War toci okienka odnoszą się do warstoci jąz wskładowanej, a ponieważ skałowaniemcalego obrazu jest czasochłonne, przekształwanie okienka da taki sam efekt:

- $OutputUnits = m * SV + b$
 - Standard DICOM prezwiduje, że wszystkie dane powinny być w yskalowane za pomocą zmiennej $gfdzicje$ (0x1053) — wartościę m i b — przekształcać za pomocą funkcji $RescaleLslope$ i $RescaleIntercept$ (0x0028, 0x1052).
 - SV — stored values — wartościowe wskazane za plikami
 - $OutputUnits$ — wartościowe wskazane za plikami.

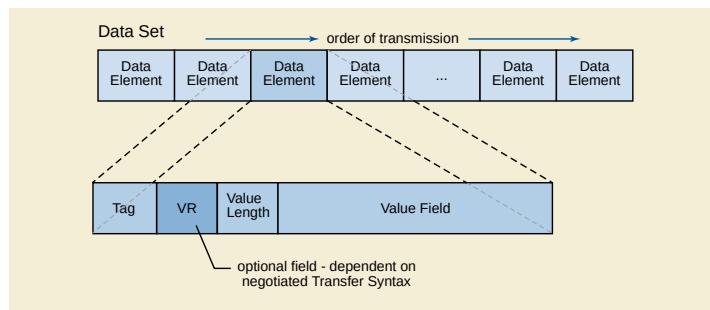
- x_1, y_1 — współzadane drugiego punktu.

- *width* — szerokość okienka
 - *center* — środek okienka
 - *left* — lewy punkt ustawienia
 - *right* — prawy punkt ustawienia

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.

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 ^{Dicom}_{Tag} PatientName (0x0010, 0x0010), czyli zwiera w sobie parametr zawierają nazwę pacjenta.

Dokładne omówienie znaczników znajduje się w sekcji 2.4.2.

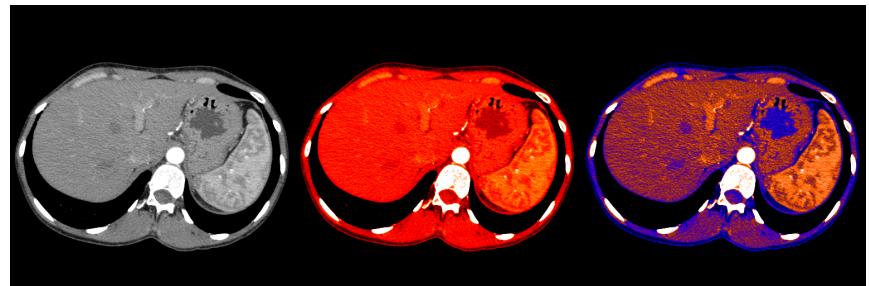
- „Value Representation”, w skrócie „VR” — to dwa bajty w postaci tekstu, informujące o formacie w jakim parametr został zapisany.

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

sposób wielokrotnego wyznaczania wartości funkcji, która wymaga sprawdzenia warunku, czy dana wartość mieści się w wybranym przedziale wartości, w tak zwanym „oknie”, co jest bardzo kosztowne obliczeniowo. Dlatego dobrym pomysłem jest stworzenie mniejszej tablicy typu LookUpTable, wypełnienie jej wszystkimi możliwymi wartościami obrazu, a następnie przerobienie obrazu z tablicą LUT. Ponieważ tablica LUT posiada wszystkie możliwe kombinacje wartości, jej rozmiar można wyznaczyć wzorem: $2^N * 3$, gdzie N to liczba bitów liczby. Standard DICOM definiuje, że liczby mogą mieć 8, 12, 16, 32 i 64 bity, jednakże, 12 bitowe i tak się zapisuje w postaci 16-bitowych w pamięci RAM. Dlatego możliwe wartości wielkości tablicy LUT to w przybliżeniu: 768 bajtów; 196 kilobajtów; 12,5 gigabajtów i 56 eksabajta ($55 * 10^6$ terabajtów). Alokowanie dwóch największych wartości może być niemożliwe, dlatego w pracy wykonano dwie implementacje algorytmu: z tablicą LUT (dla 8- i 16-bitowych obrazów) i bez tablicy LUT (dla 32- i 64-bitowych obrazów). Algorytm składa się z 3 części: wyznaczenie parametrów „okna”, przygotowanie „okna” (tylko gdy jest tablica LUT), wielowątkowa iteracja po obrazie.

Okno z LUT jest implementowane przez `Sokar::Monochrome::WindowIntStatic`. Okno bez LUT jest implementowane przez `Sokar::Monochrome::WindowIntDynamic`. Obie klasy dziedziczą po abstrakcyjnej klasie `Sokar::MonochromeWindow`, która z kolei dziedziczy po `Sokar::SceneIndicator`, dlatego od razu może wyświetlać obecne wartości „okna”. Decyzja o używanym „oknie” jest podejmowana podczas wczytywania obrazu przez klasę `Sokar::Monochrome::Scene`

UWAGA: Standard DICOM zakłada, że danymi mogą być liczby całkowite (`int`) oraz zmiennoprzecinkowe (`float` lub `double`), ale praktycznie, nie ma takich aparatów medycznych, które zapisywałyby takie obrazy, gdzie dane to liczby zmiennoprzecinkowe. Dlatego w pracy założono, że takie obrazy nie będą obsługiwanie.

Wyznaczenie parametrów okna

Najpierw wyznaczane jest okienko, które zmienia wartości obrazu na skalę od zera do jeden:

$$x_0 = \text{center} - \text{width}/2$$

$$x_1 = \text{center} + \text{width}/2$$

$$y_1 = 0.0$$

$$y_0 = 1.0$$

Reprezentacja wartości "VR" to reprezentacja wartości, który informuje w jakim formacie jest zapisany parametr obrazu. Składa się z dwóch bajtów.

Reprezentacja wartości

- Tag Patient ID (0x0010, 0x0020) — id pacjenta, unikalny identyfikator pacjenta, musi istnieć i jest to numer HIS(Hospital Information System)
 - Tag Patient BirthDate (0x0010, 0x0030) — data urodzenia pacjenta
 - Tag Patient Sex (0x0010, 0x0040) — plec pacjenta
 - Tag Patient Age (0x0010, 0x1010) — wiek pacjenta w czasie badania
 - Tag Study Description (0x0008, 0x1030) — opis badania, pole wypełniane przez technika lub lekarza
 - Tag Series Description (0x0008, 0x103E) — opis serii, pole wypełniałe przez technika lub lekarza
 - Tag Series Instance UID (0x0020, 0x000E) — unikalny numer serii, który jest nadawany kazdej nowej badaniu
 - Tag DICOM Series Instance Number (0x0020, 0x0013) — numer instancji ramki, używany w przypadku kiedy z jednego badania zostalo utworzone kilka plików DICOM
 - Tag Modality (0x0008, 0x0060) — modalność określająca rodzaj techniki diagnostycznej
 - Tag Study Date (0x0008, 0x0020) — data wykonyania badania

Laczniak

"Tag", to náhledy značk pozvala jazyček okreslíc čeze go dotyčza dane zápisné wele- ménce danyých. Značkou ještě zložony z dwoch liczb: numeru grupy i numeru elementu.

Instalacja rodażowej z załącznikiem: oznacza zakup: publiczne o przesyłce z numerem, publiczne o przesyłce z numerem grupy i prywatne o przesyłce z numerem. Pierwsza grupa jest demówana przez standard DICOM, zatrudnia-

Obie liczby to 16-bitowe liczby całkowite zapisywane w postaci heksadecymalnej.

Przez aplikacj \acute{e} hipotetyzowane z produktem sprzeda \acute{u} . Obejmuje DCOM definij \acute{e} znaczenie ponad 4000 publikacyj \acute{e} znanosciow oraz kolejna jazka VR powinny miec. Oto kilka przyk \acute{a} dow:

- Tag Dom Patient Name (0x0010, 0x0010) – nazwa pacjenta, czyle nazwisko, krotki zwazek muzi sie pojawic. Moze byc plasty w przypadku kiedy pacjent jest bezimienny

- **Diroom Patient ID (0x0010, 0x0020)** — id Pacjenta, unikalny identyfikator Pacjenta, Tag identifier to numer HIS Hospital Information System

has a society just to number HIS(Hospital Information System)

- **Freedom Patient Age** (0x0010, 0x1010) — wiek pacjenta w czasie badania

- Tag Study Description (0x0008, 0x1030) — opis bądła, pole wypełniane przez

technische Universität Leoben

• [Theo](#) — Serwis Description (0x0008, 0x103E) — opis serii, pole wyprzedaże, prez. technika

- Tag Series Instances UID (0x0020, 0x000E) — unique number series, likely used for tracking individual instances of a series.

- **Dynamic Instance Number (0x0020, 0x0013)** — number instancei ramki, uzywany w

przykładu kiedy z jednego badania zostało utworzoneyc kilka plików DICOM

- **Decom Modality (0x0008, 0x0060)** — modalność określająca rodzaj techniki diagno-

- **Diocom Study Date (0x0008, 0x0020)** — data wykonywania badania

Z uwegu na koniecznoœci ostatecznie ostatecznie zyskaœci zyskowatej oœrau warœtoœci parametry tak zwaneœgi f . Warœtoœci tej funkcji nalezy przeliczyœ, gdy zmielionie zostanie zakoœczone warœtoœci funkcji f . Miedkis kolonu wyznaczaj¹ jest wtedy poprzedz poberane tabell¹ o indeksie rownym warœosci numerycznej w obrazie. Unikamy w ten

Opis

Implementacja algorytmu

Rysunek 4.10: Porównanie jednego obrazu w trzech roznych "oknach". Zdjcie wstawne.



$$\left. \begin{array}{l} 1 \geq a \vee a \geq 1 \\ 1x > a \vee a > 0x \\ 0x \geq a \vee a \geq 0 \end{array} \right\} = (a)f$$

Wyznaczamy parametry a i b , prostymi przekształceniem przekształceniem (x_0, y_0) i (x_1, y_1) . Gdzie y_0 jest rowne 0, a y_1 jest rowne 255. Funkcja „okna” wygeneruje nasze przejęte:

- 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 `Dicom 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):

całości RGB, tak jak RGB nie pokrywa YBR. Posiadają 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 przerobić na trzy liczby, reprezentujące 3 kanały RGB, czerwony, zielony i niebieski. Dlatego do wyświetlania obrazów monochromatycznych o dużym kontraste 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 środkiem okienka *center* i długością *width*.

$$x_0 = \text{center} - \text{width}/2$$

$$x_1 = \text{center} + \text{width}/2$$

W przypadku wierszach istytucji poszawała się problem indeksowania plików i ich przeszukiwania. Wykazane konkretne zadania lub pliku w folderze, w którym znajduje się kilkażeści plików poprzeczą rozwijanie opłytymów. Dla tego standard DICOM definiuje parametry i analizę, jego danych efektywne w celu określania w których plikach znajdują się dane określone w formacie DICOMDIR, który jest plikiem indeksującym pliki DICOM w folderze. Pozwala to na efektywne przechowywanie wyciągniętych danych bez wczytywania plików bieżących.

2.4.4 Inne formaty zapisu

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 maksymalnej 2³² – 2 bajtów.

- middle name — środowwe imię, brak odpowiednika w polskim nazewnictwie
- name prefix — przekształca imię, np: mgr. inż.
- name suffix — suffiks po imieniu, brak odpowiednika

DHinguśe jednego fragmentu powinna maksymalnie wywołać 64 znaki. W przypadku mniemajeszeli jesteś segmentów, many zapłaczyć, ze się pustę.

Przykład: „prof. dr. hab. inż. Jan Nowak pracowiący w dziedzinie masztelię: „Nowak Jan prof. dr. hab. inż. „pracowiący ZEJIM” były zapisany w sposób masztelię: „Nowak Jan prof. dr. hab. inż. „pracowiący ZEJIM”

2.4.4 Linne formaty zapisu

YBR also YCB_C , to model prestressing in kolorow do przełożonych obrazów i wideo. Wykorzystuje do tego trzy typy danyń: Y - skladowa chromatyczna, B - hib CB - skladowa chromatyczna, R - skladowa roznicę między luminancją, oraz R roznicową chromatyczną. Y - B , skladowa roznicę między luminancją, oraz R hib CB - skladowa chromatyczna. Y - R , skladowa roznicę między luminancją a zazwyczajm. Kolor zielony jest uzywany na podstwiać tych trzech wartości. YBR nie pokrywa w

YBR

Wartości obrazu są przesypane do **targetbuffer** dla biblioteki QT.

- B_n — war tosć czterwionego kanału
 - G_n — war tosć zielonego kanału
 - B_n — war tosć niebieskiego kanału

- I — oznacza to, że wartości plików sie niezależne w taki sposób

Obraz monochromatyczny

- **zobrazí mapa** — obiectek obrazu ikonu, klasu `QPixmap`, docelowo powiniene mieć 128 pikseli.

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 rozwiążaniem. 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 organizacje 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++. Kompilacja 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.

- 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 ułożone 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::QImage::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 do wyświetlania ekranie, która może być używana jako urządzenie do malowania w bibliotece Qt.

3.2.1 Wyoming

Wedgejne autorów, Gó powinni się zacytać jak angielskie słowo „cute”, po polsku „kute”. Dedykuję spotężonę programistów nie jest co do tego zgodna. Abyłyby zrozumieć, że dwojaką popularność serwisa oznaczały o tematyce programistycznej, pozażuły, że Odrobięsi do przetoczoną ażet „Q.T.”, po polsku „ku te”.

3.2.2 Licencja

- <https://www.9ctcentre.org/threads/11347-How-do-you-pronounce-Qt>
 - <https://ubuntuforums.org/showthread.php?t=1605716>

Biblioteka Øt jest dystybucowana w drodze wejścia: komercyjne i ofierno. Dysystem biblioteki ma możliwość dostępu do bazy danych, co pozwala na zwiększenie jej możliwości pracy.

3.2.3 Normy i certyfikaty

The **Qt Company** posiada serię certyfikatów od FDA i UE, które ułatwiają wprowadzenie produktów z wykwalifikowaną biblioteką Qt na europejski i amerykański rynek medyczny. Lista posiadańych norm:

- IEC 62304:2015 (2006 + A1)
 - IEC 61508:2010-3 7.A.4 (SIL 3)
 - ISO 9001:2015

Wysyłając informację na temat certyfikatów można przeczytać na oficjalnej stronie Q't pod adresem https://www.qt.io/qt-in-medical/.

3.2.4 Globálne typy struktúr

- `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. Prządko (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.

Fizyka adowa klasa dziedzicza po Objekt

Syntax in sloty

```

int main() {
    // Tworzymy obiekt Przycisku
    auto *guit = new Przycisk("Guit");
    // Tworzymy obiekt Windowa
    auto *window = new Okno("Okno");
    // Tworzymy obiekt przycisku
    auto *guit2 = new Przycisk("Guit2");
    // Przypisujemy dziedziczenia
    // Przypisujemy dziedziczenia
    guit->SetParent(window);
    // W tym momencie przycisk wraz z oknem zostaja usuniete
    delete window;
}

```

Mechanizm ten pozwala nam tworzyć nowe obiekty na stercie i nie marnować sie o ich rozmiarze sprzątając. Jest to o tyle efektywne, że nie trzeba dla kazać go obiektu tworzyć dodatkowe struktury. Inną wersją mechanizmu jest tworzenie obiektów w skrzyniach, a dalej wewnątrz nich.

Na stroku za jadi uje kontrolka klasu *Sokar::DicomGraphics*, dziedziczącej po *Qt::GraphicsView*, która służy do węsiewania sceny.

Kliknięciie tego przycisku wyle Proszę o otworzenie okna ze zbiorem elementów danych pliku obrazu, który jest obecnie wyselektowany na scene.

AKCJA: **modalit  y/udata**.
Po otrzymaniu sygnal obiekty klas y *Sokar::ModalityIndicator* zazdruj  y si  
na scene powinie pokaza  c link uko  y sie w zalezno  ci od stanu pozycji.

Po otrzymaniu sygnału od ekranu klawy *Shift* i *Spacebar* przekonwertowany jest do postaci, w której jest ona zapisana w kodzie ASCII. W tym celu wykorzystywany jest algorytm, który pozwala na odczytanie kodu ASCII z danego znaku. W tym celu wykorzystywany jest algorytm, który pozwala na odczytanie kodu ASCII z danego znaku.

— Hospital Data — Dane szpitala
sie na siebie powinien pokazać lub ukryć się w zaledwieści od stanu pozyję. I o czym mowa? Główne dane, takie jak: imię, nazwisko, adres, telefon, e-mail, itp.

go odrzucza lub zanacza wszyskie pozycje w menu kontekstowym. Wszystkie pozycje sa pozytywne, a mimo to nie oznaczają niczego, co oznaczałoby, że taki element posiadałyby jakieś właściwości, np. w kontekście menu kontekstowym.

Po otrzymaniu sygnału obraz na ekranie powinien wykazywać transformację obrótu.

— Clear Transformation — Wyczęsze przekształcenia obrotu

— Philip Veretrical — Odby Instrukzane pionowow
Alkjaz: **F11P1V3rETRICAL**.

AKcja: **F1iphorizonatal**.
Po odrzuceniu siedni powinien oddać sie ultrazwiezione.

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

```

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
11    int value() const { return m_value; }
12
13    /* Sloty powinny być poprzedzone makrem "slots".
14     * Widoczność slotów można zmieniać. */
15 public slots:
16    void setValue(int value){
17        if (value != m_value) {
18            m_value = value;
19
20            /* Podczas wywoływania sygnału należy
21             * poprzedzić to makrem "emit". */
22            emit valueChanged(value);
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::QPaintDevice*, obiekcie służącym do rysowania. *Qt::QWidget* reprezentuje element graficzny interfejsu użytkownika, ma zaimplementowany mechanizm renderowania, wyświetlanego na ekranie użytkownika, obsługi 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::QAbstractButton::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 zwykłymi obiektami dziedziczącymi po *Qt::QWidget*. Przykład można zobaczyć na rysunku 3.1.

Pasek narzędzi

Pasek narzędzi znajdujący się na górze, implementowany jest przez klasę *Sokar::DicomToolBar*, dziedziczącą po klasie *Qt::QToolBar*. 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*).

Ikony 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.

- bye dosetpna na la Liniuxa, MacOS i Microsoft Windows
Ostatczne podjeto deczyzfe o wyborze biblioteki o nazwie Grassroots DICOM (GDCM),
dosetpna pod adresem <http://gdcm.sourceforge.net/>.

- mieuje licencje powallażakę jesi używawcę w potrzebnyim zakresie
 - hycie darmowa, nazielpiegi otwarto zdrodlowa
 - hycie aktywne rozwijsana — značzna wiekszoſcie bibliotek charakteryzowalna sie typu, zeb byta pozyczona i ostatnia zmaia býta wprawdzońa wiele lat temu, a proces jeli
 - hycie aktywne rozwijsana — značzna wiekszoſcie bibliotek charakteryzowalna sie typu, rozwojuj trwał od 2 do 5 miesięcę

Znalezienie dobrze biblioteki do odszukiwanego jest prosty, ponieważ biblioteka, której poszukiwane jest program, ma adresem https://idolimage.com/prog MAGIN™, do której powstanie portalu internetowego do ich indywidualnych liczb związany jest z inną. Powstała portal internetowy do ich indywidualnych liczb związanych z inną.

3.3.1 Uzasadnienie wyboru

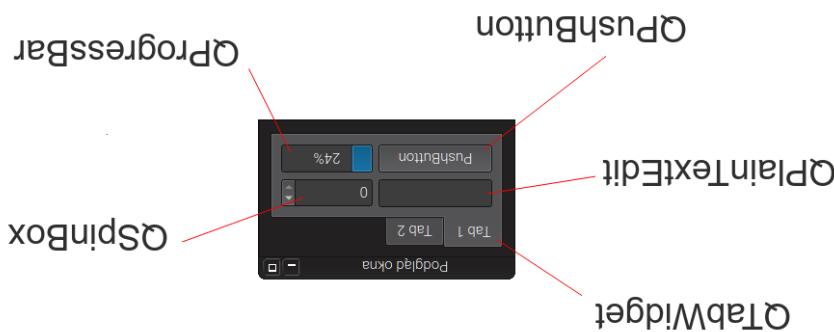
3.3 GDCM

Lysenko 3.1.

- **Użytkownicy — implementacje przekąski:** czyni kontrolę przystosowaną do wpro- wadzania liczb przez użytkownika. Posiada dwa dodatkowe przykłady powalające na łatwy sposób zwilekszyć lub zmniejszyć zasoby. Przykład monza zrobacye na

- *Qt:QProgressBar* — implementuje paszek postępu w dwoch wersjach pozionowanych poziomo: z lewej do prawej i od góry do dołu. Dziedziczy bezpośrednio po klasie *Qt:QWidget*. Przykład pozionów:
 - *Qt:QFrame*, z kolejną po lewej do prawej. Przykład mówiący o pozycji 3,1.
 - *Qt:QAbstractScrollArea*, które dziedziczy po *Qt:QFrame*. Dziedziczy po klasie *Qt:QWidget* po której mówiącą się w poziomie i wysokością.
 - *Qt:QPaintDevice* — implementuje pole mówiące o rozmiarze tekstu i rozmiarze.

U3-Übung 3.1: U3-zyklische orientierte Programme mit Objekten und geschichteten Werten



The image shows a screenshot of a Qt application window titled "Podglid okna". The window contains several UI elements: a "Pushbutton" at the top right, a "PlainTextEdit" area below it, a "TabWidget" with two tabs ("Tab 1" and "Tab 2") at the bottom, a "SpinBox" with the value "0" in the center, and a "Progressbar" with a value of "24%" at the top left. Red lines with labels point to each of these components: "QPushbutton", "QPlainTextEdit", "QTabWidget", "QSpinBox", and "QProgressbar".

Rysunek 4.9: Wyglad zakladki wraz z numeracj \acute{e} elementow interfejsu. Zdjecie w fasonie.



Dodatkowo posiada obiekt `kolekci` scen opisana w sekcji 4.5.3.

- mięsice na scenie z obrazem DICOM na strodku — implementowalny za pomocą klas *Solar::DicomGraphics*, opisany w sekci 4.5.4
 - kolejne na scenie z obrazem DICOM na strodku — implementowalny za pomocą klas *Solar::DicomImage*, opisany w sekci 4.5.4
 - swak filmu w dolięć części — implementowalny za pomocą klas *Solar::MovieBar*, opisany w sekci 4.5.4
 - podgląd miniaturki obrazów w prawej części — implementowalny za pomocą klas *Solar::FrameChooser*, opisany w sekci 4.5.4

- Passk narazdži znađišaši sile na grize — implementovani za pomocu klase *Sofar*;

Segetwarciaja odbywaa sie za pomoca **FileSet**:*create()*. Do funkcji jest przesyany wektor z wezlytami plikami DICOM, nastepnie dzeli ona pliki na zbiory zawiarszyjny wektor z wezlytami plikami DICOM, nastepnie dzeli ona pliki na zbiory zawiarszyjny wektor z samimi serii, tworzy obiekty zbiory DICOM. Osteczenie zwraca jazycie dzelicia telsamej serii, tworzy obiekty zbiory DICOM. Sortowanie plikow DICOM na wektor z glosowymi obiekta zbiory DICOM. Sortowanie plikow DICOM. Konstrukcja klasyczna polegajaca na wybraniu jednego pliku z katalogu, a reszta jest publiczny.

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 protokół komunikacji sieciowej, z których oba powinny zostać rozszerzone dla zapewnienia pełnego zestawu narzędzi badaczy 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 komplikacji 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 za 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>.

- *gdcm::Reader* — klasa służąca do wczytywania pliku DICOM
- *gdcm::ImageReader* — klasa służąca do wczytywania obrazu DICOM, dziedziczy po *gdcm::Reader*, jest wstanie wygenerować obiekt obrazu
- *gdcm::Image* — obiekt obrazu ułatwiający pobieranie informacji
- *gdcm::File* — obiekt 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 wyświetlanych 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 wczytyanych 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.

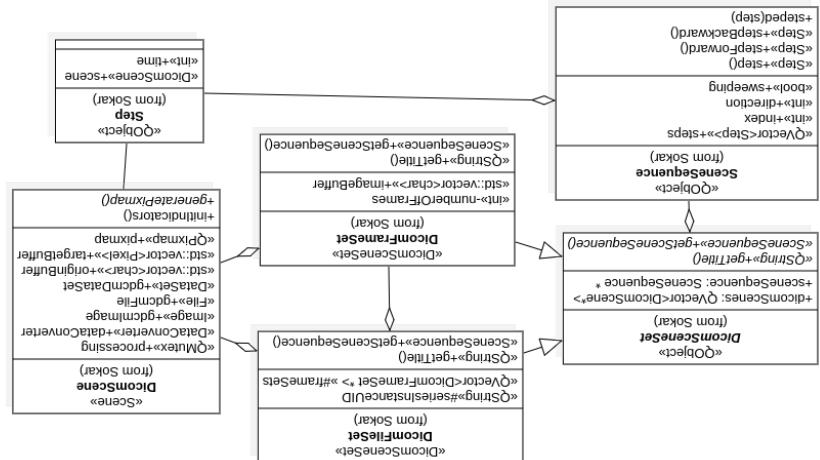
Segregowanie obrazów

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).

- imdeks, w którym obecnie znajdują się sękwencja
 - imdeks, który m. wykorzystać krok w stronę końca sękwencji
 - Sókar::SceneSequence::stepForward() — krok do przodu — zwiększa imdeks tym razem
 - rozdzieli przemiany — watośce logiczna informacja w jazdy sposob ma zachować się, gdy sękwencja dojazde do końca, lub poczatku
 - rozdzieli skwencja — skwencja może być w stronę poczatku lub końca
 - skwencja sękwencji — skwencja może być w stronę poczatku lub końca
 - rozdzieli przemiany — warotśce logiczna informacja w jazdy sposob ma zachować się, jeśli dojazd do końca skwencji skwencja skwencji — skwencja może być w stronę poczatku lub końca
 - imdeks do tyłu.
 - Krok implementowanej przesy klasie Sókar::Step zazwierają nastepującej informacji: wskaznik do sceny oraz czas trwania sceny.
 - Skwencja ma w budownictwie funkcje zapewniające przeszwanie się po imdeksie na kolejne skwencje ma w budownictwie funkcje zapewniające przeszwanie się po imdeksie na kolejne skwencje.
 - Sókar::SceneSequence::stepForward() — krok do przodu — zwiększa imdeks tym razem

Sekwencja scen

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



Bonieje zaprezentowano kilka przykładow uzycia biblioteki GDCM.

- `gdcn::StringFilter` — pomocnicza klasa służąca do konwersji na obiekt tekstu
 - `gdcn::Tag` — obiekt znacznika
 - `gdcn::DataElement` — obiekt elementu danych
 - `gdcn::DataSet` — obiekt zbioru elementów

4.5.3 Kolekcje scen

Abstractkyjia klasa *Sokar*: *DiComScreenSet* implementuje wektor scen za promocu klasycznego. Klasa *Sokar* z jednego pliku. Diagram klas UML przedstawia sie na rysunku 4.8.

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        TagWindowWidth(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(TagWindowWidth)){
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 ^{Dicom} Tag KVP (0x0018, 0x0060)

- MR — rezonans magnetyczny
- „Repetition time” — czas repetycji — pobierany ze znacznika ^{Dicom} Tag Repetition Time (0x0018, 0x0080).
- „Echo time” — czas echa — pobierany ze znacznika ^{Dicom} Tag Echo Time (0x0018, 0x0081).
- „Magnetic field” — pole magnetyczne — nominalna wartość pola magnetycznego wyrażona w teslach pobierana ze znacznika ^{Dicom} Tag Magnetic Field Strength (0x0018, 0x0087).
- „SAR” — swoiste tempo pochłaniania energii — pobierane ze znacznika ^{Dicom} Tag 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.

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

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 niebos, 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 $\text{Dicom}_{\text{Tag}}$ 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 $\text{Dicom}_{\text{Tag}}$ 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 $\text{Dicom}_{\text{Tag}}$ Institutional Department Name (0x0008, 0x1040) i wyświetlany bez ingerencji.
- Producent wyposażenia wraz z modelem urządzenia
Tekst jest obierany z $\text{Dicom}_{\text{Tag}}$ Manufacturer (0x0008, 0x0070) i $\text{Dicom}_{\text{Tag}}$ Model Name (0x0008, 0x1070), oddzielony spacją i wyświetlany bez ingerencji.
- Nazwisko lekarza wykonującego badanie
Tekst jest obierany z $\text{Dicom}_{\text{Tag}}$ Referring Physician Name (0x0008, 0x0090) i wyświetlany bez ingerencji.
- Nazwisko operatora wspierającego badanie
Tekst jest obierany z $\text{Dicom}_{\text{Tag}}$ Operators Name (0x0008, 0x1070) i wyświetlany bez ingerencji.

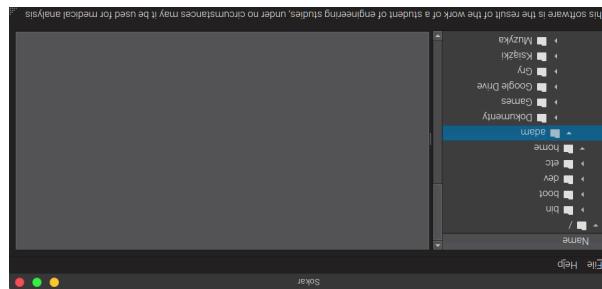
Orientacja obrazu

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

4.1.3. Výsuneček 4.1: Okno přezgádávacího rozpočtu po určitomu čase. Zdělcíe vlastnosti.



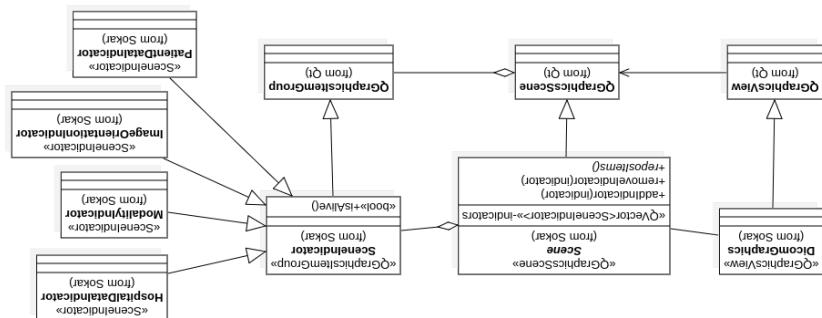
Po uruchomieniu programu uzytkownikowi nalezaze sie glosnie okno, po kazane na menu (obiekt *QMenuBar*), dzwo w plikow (obiekt *QFile*), obiekt *QListWidget* 3 elementy: *Wyjścia* 4.1, implementowane przez klasę *Sofer* (*MainWindow*). Okno zawiera 3 elementy: *zakladka* z obrazami (obiekt *QImage*) i *skazy* (*Sofer::DictomTables*).

Wszystkie funkcje wektorów klas są oznaczone następująco:
Sokar::DataConverter::ToString()
Sokar::DataConverter
Przykład poniżej:
Dla tego, aby zwiększyć czytelność pracy, zostało zastosowana konwencja poprzedzająca klas z aplikacji przedrostkiem *Sokar::*, który za razem jest przestronią nazw programu.

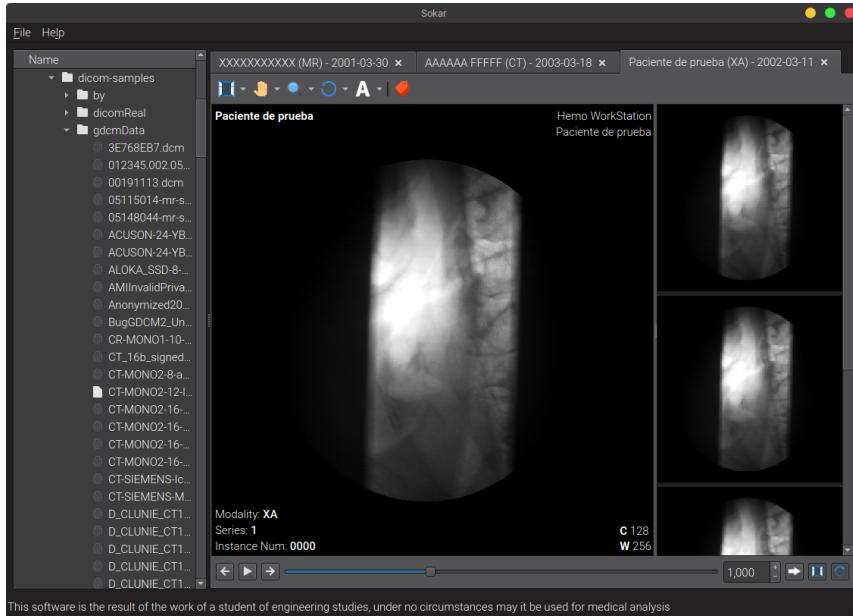
4.3 Graficzny interfejs użytkownika

4.2 Wielopłatformowość

- Nazwa pacjenta oraz plec przekształca się w tagi Patient Name (0x0010, 0x0010) o VR.PN. Nazwa pacjenta zapisana jest w tagu Patient Sex (0x0040, 0x0040) i może mieć następujące wartości:
 - „M” — oznacza męczyzne, wyswietlana jako znak 
 - „F” — oznacza kobietę, wyswietlana jako znak 
 - „U” — oznacza inną płcią i nie jest wyswietlana
- Przykład: „Jan Nowak .
- Identyfikator Pacjenta
 - Unikalny identyfikator pacjenta ze znacznika **Tag** Patient ID (0x0010, 0x0020) wyswietlany jest w takiej formie, w jakiej jest zapisany. W praktyce najczęściej jest to numer z systemu uzyskanego w danym szpitalu, rzadziej numer PESEL.
 - Data urodzenia oraz wiek pacjenta w trakcie badania
 - Data urodzenia zapisana na formacie **YYYY-MM-DD**. Dodać do Patient Birth Date (0x0010, 0x0030) i jest zamieniana na format **YYYY-MM-DD**. Dodać do Patient Date (0x0010, 0x0010), jeśli PacjentAge (0x0010, 0x1010) jest obecny, wyswietlany jest takiż wiek pacjenta w czasie badania.



Wszystkie elementy wyświetlające dane z pliku DICO.M po klasie *Sokar* i SceneLndicator. Diagram klas UML zasadnicze się na rysunku 4.7.



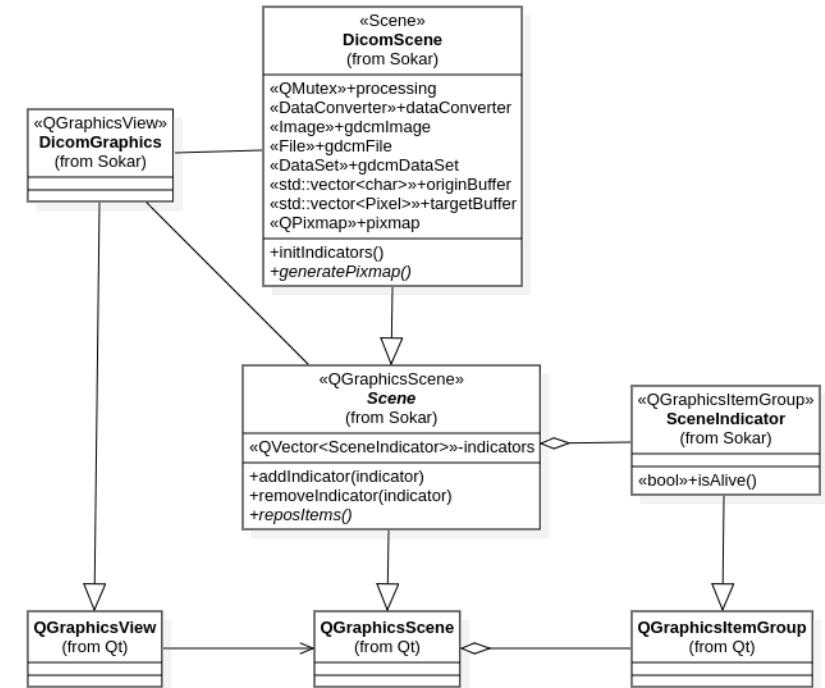
Rysunek 4.2: Okno przeglądarki z wczytanymi kilkoma obrazami. Zdjęcie własne.

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 wczytywany 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*.

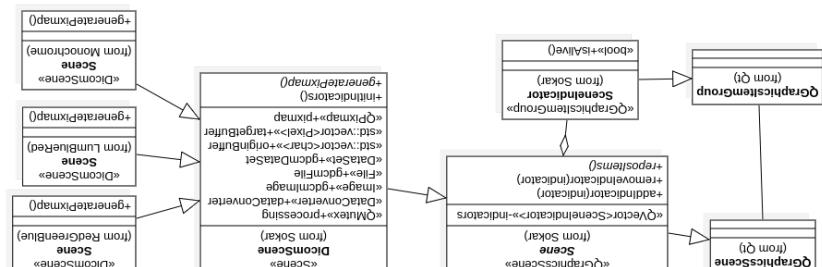
4.4 Projekt struktury obiektowej programu

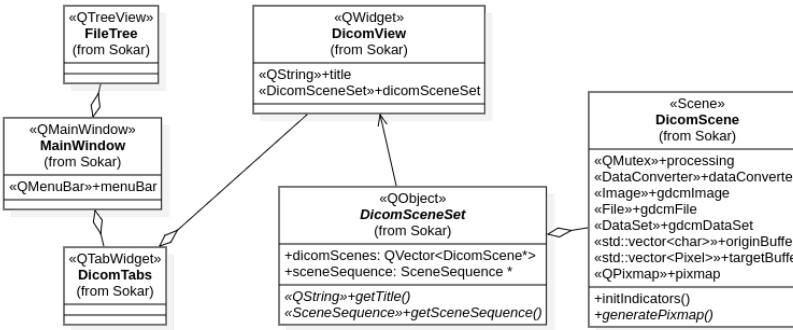
Rysunek 4.3: Przykładowa scena z obrazem monochromatycznym. Zdjęcie własne.



- **Qt::GraphicsScene::group** — element grupujiacy wiele elementow. Pozwala na fatwia implementacyjne bardziej zlozonej struktury
 - **Qt::GraphicsItem::group** — element wyiswietlajacy obraz graficzne, obiekty klasy
 - **Qt::GraphicsItem::group** — element wyiswietlajacy prostą linię z punktami A do B

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-bitowych 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 obracania 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