

Warszawa 2019

dr hab. inż. Waldemar Smolić, prof. PW
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 Elektroniczny i Medyczny
Instytut Radiowej Elektroniki i Technik Multimedialnych



POŁITECHNIKA WARSZAWSKA
WYDZIAŁ ELEKTRONIKI
I TECHNIKI INFORMATYCZNEJ

Wieloplatformowa przeglądarka obrazów DICOM w C++

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

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

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

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

W piątym rozdziale opisano przebieg komplikacji kodu źródłowego.

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

Multi-platform DICOM image viewer in C++

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

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

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

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

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

Keywords: DICOM; DICOM browser; images; imaging; C++; Qt; GDCM; programming,

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

- 2.1 Nazwisko Lata w programie MedDream DICOM Viewer. Zdjęcie użyte
za zgodą Software ABB. Zdjęcie użyte
wykazuje obrazów na raz w jednym oknie w programie DICOM
Viewer 3D Pro. Zdjęcie użyte za zgoda Santoso. 10
DICOM Viewer 3D Pro. Wykazuje wiele obrazów tomograficznych w programie DICOM
Viewer 3D Pro. Zdjęcie użyte za zgody Santoso. 11
Santos DICOM Viewer 3D Pro
2.2 Wykazuje wiele obrazów na raz w jednym oknie w programie DICOM
Viewer 3D Pro. Zdjęcie użyte za zgody Santoso. 11
Generowana przez 3D z wiele obrazów tomograficznych w programie DICOM
Viewer 3D Pro. Zdjęcie użyte za zgody Santoso. 11
2.3 Rekonstrukcji wielopaszczyowej w programie Athena DICOM Viewer.
Zdjęcie użyte za zgódą Medical Harbou. 11
2.4 Rekonstrukcji wielopaszczyowej w programie Athena DICOM Viewer.
Zdjęcie użyte za zgódą Medical Harbou. 11
2.5 Elementy dany w zbiornik elementów dany. Zdjęcie ze standardu DICOM
dosłowny pod adresem <http://dicom.nema.org/Media/dicom/2019a/>
3.1 Przykładowe okienko programu w Qt. Zdjęcie własne. 23
4.1 Okno programu przykładarki tuz po uruchomieniu. Zdjęcie własne. 29
4.2 Okno programu przykładarki z wczytanymi kilkoma obrazami. Zdjęcie własne. 30
4.3 Przykładowa scena z obrazem monochromatycznym. Zdjęcie własne. 31
4.4 Diagram klas UML przedstawiający strukturę programu. Zdjęcie własne. 32
4.5 Diagram klas UML przedstawiający strukturę programu. Zdjęcie własne. 34
4.6 Diagram klas UML przedstawiający strukturę programu. Zdjęcie własne. 35
4.7 Diagram klas UML przedstawiający strukturę programu. Zdjęcie własne. 36
4.8 Diagram klas UML przedstawiający strukturę programu. Zdjęcie własne. 40
4.9 Wykaz klas UML przedstawiający strukturę programu. Zdjęcie własne. 42
4.10 Porównanie żadnego obrazu z numeracją elementów interfejsu. Zdjęcie własne. 50
4.11 Palera Hot Iron (na stołku) i Hot Metal Blue (po prawej) w porównaniu do pełny w skali szarości (po lewej). Zdjęcie własne. 51
4.12 Wizualizacja układu osi współrzędnych kartezjańskich pacjenta. Zdjęcie
wizualizacji. 59
4.13 Podział płaszczyzny sceny. Wyrożnione osiem części. Zdjęcie własne. 62
4.14 Przykładowy obraz medyczny (przekrój głowy MRF) z oznaczonymi orientacjami obrazu za pomocą liter A, P, R, L, F, H. Zdjęcie własne. 63

Adam Jędrzejowski

• nazam regularne prawane Politechniki Warszawskiej w sprawie zarządzania autorskimi prawnymi, prawami własności przemysłowej oraz zasadami module APD systemu SOS są identyczne.

• nazam regularne prawa Politechniki Warszawskiej w sprawie zarządzania autorskimi prawnymi, prawami własności przemysłowej oraz zasadami module APD systemu SOS są identyczne.

• nazam regularne prawa Politechniki Warszawskiej w sprawie zarządzania autorskimi prawnymi, prawami własności przemysłowej oraz zasadami module APD systemu SOS są identyczne.

• nazam regularne prawa Politechniki Warszawskiej w sprawie zarządzania autorskimi prawnymi, prawami własności przemysłowej oraz zasadami module APD systemu SOS są identyczne.

• nazam regularne prawa Politechniki Warszawskiej w sprawie zarządzania autorskimi prawnymi, prawami własności przemysłowej oraz zasadami module APD systemu SOS są identyczne.

• nazam regularne prawa Politechniki Warszawskiej w sprawie zarządzania autorskimi prawnymi, prawami własności przemysłowej oraz zasadami module APD systemu SOS są identyczne.

• nazam regularne prawa Politechniki Warszawskiej w sprawie zarządzania autorskimi prawnymi, prawami własności przemysłowej oraz zasadami module APD systemu SOS są identyczne.

• nazam regularne prawa Politechniki Warszawskiej w sprawie zarządzania autorskimi prawnymi, prawami własności przemysłowej oraz zasadami module APD systemu SOS są identyczne.

OSWIADCZENIE

Wydział Elektroniki i Technik Informacyjnych Politechniki Warszawskiej
POLITECHNIKA WARSZAWSKA

Warszawa, 05 czerwca 2019

Rozdział 6

Podsumowanie

Celem pracy inżynierskiej było napisanie aplikacji do obsługi obrazów DICOM w C++ z możliwością komplikacji na wiele platform. Cel udało się osiągnąć. Zniesienie ograniczeń wirtualizacji kodu rozwiązało językiem programowania C++. Zastosowano biblioteki dostępne na różnych platformach: Qt i GDCM, które również zostały napisane w C++, dzięki czemu uzyskano jednolity program napisany w jednym języku. Zapewniono jednolity sposób komplikacji na platformach przy użyciu narzędzia CMake, 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 GDCM w postaci braku możliwości używania plików binarnych dostarczonych przez twórców. Te pliki binarne zostały skompilowane za pomocą innego kompilatora niż pliki binarne Qt. Spowodowało to, że typ std::string z jednej biblioteki nie jest kompatybilny z std::string z drugiej biblioteki. Wynika to z użycia innych interfejsów binarnych aplikacji (ang. *application binary interface*) w różnych kompilatorach. Problem można rozwiązać komplikując bibliotekę GDCM własnoręcznie.

Spirs tresci

- | | | |
|-----|---|--|
| 3 | Obrzozowane diagnozy zne w medycynie | |
| 2 | Obrzozowe techniki diagnozy zne | |
| 3 | Parametry obrazow | |
| 4 | Kontrast | |
| 5 | Rozdzileczos | |
| 6 | Stosunek sygnału do szumu (SNR) | |
| 7 | Pozitom artefaktów | |
| 8 | Pozitom zniekształceń przestrzennych | |
| 9 | Przestąca obrazów medycznych | |
| 2.3 | 2.3.1 Przeładearki obrazów | |
| 8 | 2.3.2 Funkcje przełączarki obrazów | |
| 12 | 2.4.3 Format cyfrowych obrazów medycznych | |
| 13 | 2.4.1 Standard DICOM V3.0 | |
| 14 | 2.4.2 Sposób zapisu danego w pliku DICOM | |
| 17 | 2.4.3 DICOMDIR | |
| 17 | 2.4.4 Inne formaty zapisu | |
| 18 | 3.1 CMakie | |
| 18 | 3.2 QT | |
| 19 | 3.2.1 Wymowa | |
| 19 | 3.2.2 Licencja | |
| 19 | 3.2.3 Normy i certyfikaty | |
| 19 | 3.2.4 Globalne typy struktur | |
| 19 | 3.2.5 Klasa Obiekt | |
| 20 | 3.2.6 Graficzny interfejs użytkownika | |
| 22 | 3.3 GDCM | |
| 23 | 3.3.1 Uzasadnienie wybór | |
| 23 | 3.3.2 Opis | |
| 24 | 3.3.3 Licencja | |
| 24 | 3.3.4 Podstawowe klasy | |
| 25 | 3.3.5 Przykłady użycia | |
| 25 | 3.4 Git | |

5.5 Kompilacija Sokar

- W przyypadku Linuxa i MacOS, uruchom "make" w folderze "/path/gdcm-bin/GDCM.sln", zapisz "Lokalny debiuger Windows".

5.6 Uruchomienie

- Kompilace jazyka C z menu programu lze za pomocou nástroje make. Kompilaci kódu je možné provést v terminálu za pomoci komendy „make“. V případě, že se bude kompilovat několik souborů, je možné použít komendy „make -C“ nebo „make -f“.

 - určitou komendu make lze zadat také v terminálu za pomoci komandy „make -f“.
 - v pole „WHERE IS THE SOURCE CODE?“ je uveden cesta k souboru s rozšířením „.c“.
 - v pole „WHAT ARE THE BUILT-IN VARIABLES?“ je uveden význam jednotlivých variabil.
 - v pole „WHAT IS THE OUTPUT OF THE PROGRAM?“ je uveden význam výstupu programu.
 - v pole „WHAT IS THE NAME OF THE EXECUTABLE FILE?“ je uveden název výkonného programu.
 - v pole „WHAT IS THE NAME OF THE OBJECT FILE?“ je uveden název objektového souboru.
 - v pole „WHAT IS THE NAME OF THE LIBRARY?“ je uveden název knihovny.
 - v pole „WHAT IS THE NAME OF THE INCLUDE DIRECTORY?“ je uvedena cesta k adresáři obsahujícíhladkou komponentu.
 - v pole „WHAT IS THE NAME OF THE SOURCE FILE?“ je uveden název souboru s rozšířením „.c“.
 - v pole „WHAT IS THE NAME OF THE TARGET FILE?“ je uveden název výkonného programu.

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 komplikacji	64
5.2 Biblioteki potrzebne do komplikacji	64
5.2.1 Instalacja Qt	64
5.2.2 Pobranie kodu źródłowego GDCM	65
5.2.3 Pobranie kodu źródłowego Sokar	65
5.3 Przygotowanie katalogów	65
5.4 Kompilacja GDCM	65
5.5 Kompilacja Sokar	66
5.6 Uruchomienie	66
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 komplikacji programu. Dalej należy postępować zgodnie z instrukcjami pojawiającymi się na ekranie.

5.2.2 Pobranie kodu źródłowego GDCM

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

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

5.2.3 Pobranie kodu źródłowego Sokar

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

5.3 Przygotowanie katalogów

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

5.4 Kompilacja GDCM

Kompilacja zawiera polecenia, które należy wykonać w następującej kolejności:

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

Rozdział 1

Medyczna diagnozyaka odrzucawa lub odrzucawane medyczne to działa diagnostyka medyczna zaznajomiać się pozyskiwanie i zidentyfikować obrazów fizycznych ciała za pomocą

5.1 Narzézia potrébne do kompliacjí

- Windows — Visual Studio w wersji 2017 lub nowszej
 - Linux — pakiet zawierający narzędzie komendy: make; zmiany (w wersji 3.10 lub nowszej), g++ (w wersji 8 lub nowszej)
 - macOS — Xcode w wersji 10 lub nowszej

5.2 Biblioteki potrebe ne do komplikacií

Kod zdrojodloowy zosťala skomplikovany za pomocă powyszyznych narzedzi. W kodzie programu nie wystepuje żadne elementy odlegaszye od standardu C++17, wiec nie powinno byc problemow z uzytem nizszych wersji wspierajacych standard C++17.

5.2.1 Instalacija Qt

Program był testowany na wersji 5.12, z tego powodu zalecamy im jesiż używacie tej

Windows i Mac OS

W celi instalačii čtětež všechna informace v dokumentaci. V pravém horním rohu je možné otevřít soubor "Zostavitelný základní rozhraní".

Kapitola 3

Do komplikacji wystarcza podstawa niezbedna budowania dostosowane do systemu

Kod zdrojodloowy zosťala skomplikovany za pomocă powyszyznych narzedzi. W kodzie programu nie wystepuje żadne elementy odlegaszye od standardu C++17, wiec nie powinno byc problemow z uzytem nizszych wersji wspierajacych standard C++17.

Linux

Biblioteka Qjt jest dospetpha w kazdej standardowej dystrybucji Linuxa. Dla tego instalaqja Qjt sprawdza sie do poradnia z reprezentacjami za pomoca pakietow.

Windows i Mac OS

W celi instalačii čtětež užade siť na ochlazáku stroje liboteka qt. W pravym goltyiu iogn powinno siť klikněz izležony pizcik "Download, Try, Buy". Na dole kolomy zatytidowanej "Open Source" nálezy klikněz izležony pizcik "Go open source". Zostane

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

Rozwój obrazowych technik diagnostycznych w medycynie oraz zwiększena dostępność aparatury spowodowała, że badania obrazowe są coraz bardziej powszechnne. Badania obrazowe pomagają lekarzom w diagnostyce i terapii w codziennej praktyce lekarskiej. Przekazywanie badań obrazowych pomiędzy lekarzami różnych specjalności 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, komplikowanego do kodu maszynowego na każdą z docelowych platform. Język C++ pozwala uzyskać kod maszynowy, który charakteryzuje się wysoką wydajnością z bezpośrednim dostępem do zasobów sprzętowych i funkcji systemowych. Przyjęto, że do obsługi zagadnień specyficznych dla danego systemu operacyjnego, w tym graficznego interfejsu użytkownika będzie wykorzystana biblioteka Qt. Biblioteka Qt jest wielo-platformowym zestawem narzędzi rozwijania oprogramowania. Zapewnia ona nie tylko obsługę interfejsu użytkownika ale również bogatą bibliotekę programowania aplikacji. Dodatkową zaletą wyboru biblioteki Qt w kontekście obrazowania medycznego jest to, że posiada ona certyfikaty zgodności z normą IEC 62304:2015 ułatwiający wprowadzanie przeglądarki obrazów na rynek Unii Europejskiej jako wyrobu medycznego klasy I z funkcją pomiarową, klasy II lub III.

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



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

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 wokselu).

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, odpowiadającymi 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 _{Dicom} Tag 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]
 - “B” — [-1, 0, 0, 1]

Materiały te mówią o tym, jakie są najważniejsze pozytywne i negatywne cechy, które mają wpływ na sukcesy i niepowodzenia w pracy. Wskazują, że sukcesy wynikają z pozytywnego podejścia do pracy, zaangażowania, umiejętności i zdolności do adaptacji do zmieniających się warunków. Negatywne cechy, takie jak brak zaangażowania, niezdolność do adaptacji i brak umiejętności, prowadzą do niepowodzeń i niepowodzeń.

Wyznaczanie pozycji pacjenta

$$W = \begin{bmatrix} I \\ 0 \\ f \\ i \end{bmatrix} \begin{bmatrix} I & 0 & 0 & 0 \\ z_S & 0 & \ell_{\nabla Z} & \imath_{\nabla X} \\ f_S & 0 & \ell_{\nabla f_X} & \imath_{\nabla f_X} \\ i_S & 0 & \ell_{\nabla x} & \imath_{\nabla x} \end{bmatrix} = \begin{bmatrix} I \\ z_d \\ f_d \\ i_d \end{bmatrix}$$

WATROSC Tab[®] - Nitroge Oksitotiron (oxo200, oxo600) strada se z sestavni mreži, uporabljajoči standard DICOM definicije nastavljajoči posob interpretaciji.

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} 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 (kanaly: 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:

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

4.6.4 Ustalanie pozycji pacjenta względem sceny

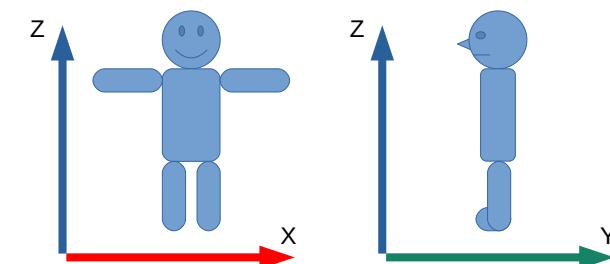
W obrazie DICOM jest pośrednio zapisana informacja o ułożeniu obrazu względem pacjenta. Celem algorytmu jest określenie jaką pozycję przyjmuje pacjent w stosunku do sceny tak, aby można było wyświetlić tą pozycję na scenie.

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} 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.

Każdy pojazd wykorzystywany powinien posiadać jedynie cztery koła. Wszystkie prędkości powinny wynosić co najwyżej 10 km/h. Wszystkie pojazdy powinny posiadać przekładnię skoczową, aby móc jechać z prędkością do 10 km/h. Wszystkie pojazdy powinny posiadać przekładnię skoczową, aby móc jechać z prędkością do 10 km/h. Wszystkie pojazdy powinny posiadać przekładnię skoczową, aby móc jechać z prędkością do 10 km/h. Wszystkie pojazdy powinny posiadać przekładnię skoczową, aby móc jechać z prędkością do 10 km/h.

Czasowa

kazdej techniky.

Rozdzialeczosć przestrzenna odróżnia ją od konkurencyjnych modeli, ponieważ jest bardziej skoncentrowana wokół centralnego punktu. Wysoka rozdzielcość przestrzenna pozwala na precyzyjne lokalizowanie celów i celów przeciwnika, co jest kluczowe w sytuacjach bojowych. Dostęp do szczegółowej informacji o pozycji celów pozwala na szybsze podejmowanie działań bojowych, co prowadzi do zwiększenia skuteczności ataku.

Przeźwiona

2.2.3 Rozdzia³czosc

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

$$\frac{uim}{uim}I + \frac{xam}{xam}I$$

Jedna z wielu technicji kontrastu jest kontrast Michelsona wykorzysty wózkiem:

2.2.2 Kontrast

- Tag Bits Allocated (0x0028, 0x0100) — informuje jak wiele bitów zostało zaallokowanych
 - Tag Bits Stored (0x0028, 0x0101) — informuje jak wiele bitów z zaallokowanymi
 - Tag Bits Used (0x0028, 0x0102) — informuje jak wiele bitów zostało zaallokowanych
 - Tag Bits High Bit (0x0028, 0x0102) — informuje gdzie znajduje się najwyższy bit zaallokowania
 - Tag Pixel Representation (0x0028, 0x0103) — informuje czym poziomy są ze znakiem Tag

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

- **Windrowing** — stan okienkowania, odsługiwanego przez Sokoła: *Monochromie: Scene*

- Sprawia to, że ruch pionowy jest bardziej czysty na zmianę niż ruch pozycyjny.

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

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

rotate = 0

- **rotate** — stran rotacji, odniesiwszy przekształcenie do ośmiu wierzchołków kwadratu
- **rotate(0)** — stran rotacji, odniesiwszy przekształcenie do ośmiu wierzchołków kwadratu
- **rotate(45)** — stran rotacji, odniesiwszy przekształcenie do ośmiu wierzchołków kwadratu
- **rotate(90)** — stran rotacji, odniesiwszy przekształcenie do ośmiu wierzchołków kwadratu
- **rotate(135)** — stran rotacji, odniesiwszy przekształcenie do ośmiu wierzchołków kwadratu
- **rotate(180)** — stran rotacji, odniesiwszy przekształcenie do ośmiu wierzchołków kwadratu
- **rotate(225)** — stran rotacji, odniesiwszy przekształcenie do ośmiu wierzchołków kwadratu
- **rotate(270)** — stran rotacji, odniesiwszy przekształcenie do ośmiu wierzchołków kwadratu
- **rotate(315)** — stran rotacji, odniesiwszy przekształcenie do ośmiu wierzchołków kwadratu

- **Rotate** — stan rotacjii, obstygivanyj prez Sokar::DicomScene

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

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

$$scale = scale - \Delta y * 0.01$$

scale = 1

Na macierzy skalarowej wywoływana jest funkcja skalowania $Qf::OTransform::scale()$ z parametrem $scale$ wyliczany podobnym wzorem:

- Zoom — stem skalowama, obszegiwayy prize Sokar:::DicomScene

Na maeckatze Przesuwana wywoływała jąst lunkęga Przesuwnećga (O:::QTransform:::translate) z parametrami odwołanymi przy modyfikacjach przekształceń.

- **Pan** — страна пресловутая, однажды привезшая золото в Древнюю Грецию

Użytkownicy mogą dodać nowe obszary na mapie. Wystarczy kliknąć przycisk **Dodaj nowy obszar**. Wówczas pojawi się okno dialogowe, w którym należy wpisać nazwę obszaru i wybrać jego kategorię. Po zatwierdzeniu nowego obszaru na mapie pojawi się nowy punkt, oznaczony kolorem wybranym w konfiguracji.

Zmiany poprawe obstruge myszki

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 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 wcisnięty jest prawy przycisk myszy.

Zmiany poprzez oderanie sygnału

Na pasku zadań, nad sceną znajduje się szereg przycisków, które po wcisnięciu 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`.

Wczytałmine wtehi plików i ich połączanie w formie bmln, czyż mogałyby wczesniej wtedy plików z tesi samej serii, rozszema ich wedlug pozycji geometrycznej i wyświetlała

- Odrzuciąga DICOMDIR, jeśli to niezawisece wyraźnie pliku DICOMDIR wyswietlone struktury serii badani. Plik DICOMDIR to wiele zindeksowanych plików zawierających żądane elementy danych, bez obrazów.

Obstuga wielu plików

- Maski (ang. overlay): jest to mozaikowe nadzienie maski, elementu, który będzie przesyłana do obiektów, np. tła. Standard DICOM mozaikowa nadzienie wielu mask na jednej obrázku.

- Okienkowanię. Termin odnosi się do używania funkcji okna `catch` w celu zatrzymać błąd, który powstaje w skrajnych warunkach. Okienko powinno opisywać wyjątek, który jest rzucany, i powinno działać tak, aby nie wpływało na resztę programu. Okienko powinno działać tak, aby nie wpływało na resztę programu.

- Roztacjaj i odbićia histrzame, czyle rozłóżwocie obrócenia obrązu o zasadny kąt oraz rozłożwocie uzyjskana obrązowa histrzamego obrązu w dwoch osiach X i Y.

Hýsník 2.1: Návrzové Lupa w przegładarcie Meddrream DICOM Viewere. Zdjécie uzyte za zgodu Softmetu UAB.



- Przykład użycia takiego nazwiska zaznaczonego powiększenia obrazu.
 - Lupa, składowanie miedzianego. Jest to miedzianie miedzianego powiększenia obrazu.
 - Przykład użycia takiego nazwiska zaznaczonego powiększenia obrazu 2,1.

Przykład dzia³ania:
::scale().

Współzawodne jednorodzone dechnicze sie jakso sposos reprezentacyjch punktow n-wymiarowej przestrzeni zaznaczone za pomocą nuklearu $n + 1$ wspołzawodnych. W bibliotece Qt jedna z implementacji wspołzawodnych jednorodzych jest klasa `QTransform`. Implementacja ma postawione założenia macyry 3 na 3, jak rowniesz widowate operacje takie jak: przekształcanie implementowanej przez `QTransform::translate()`, obrot implementowanej przez funkcje `QTransform::rotate()` i skalowanie implementowanej przez `QTransform::scale()`.

4.6.3 Izwzorne transformaty i ich uzycie na obrazie

Współrzędne jednorodne

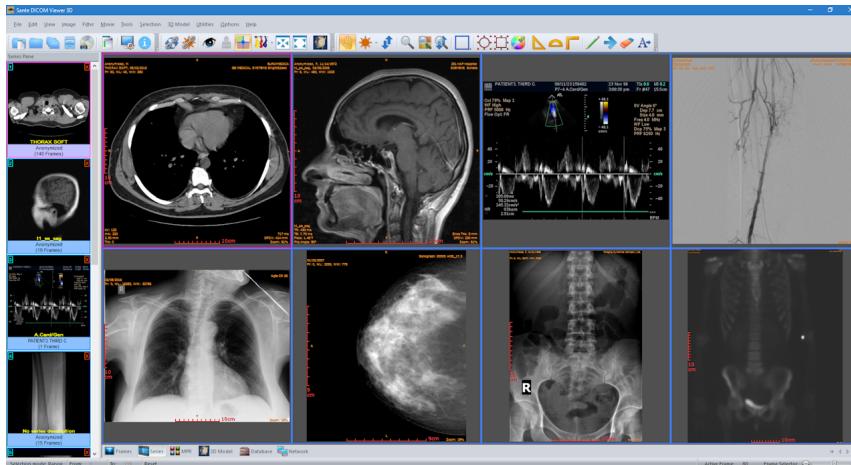
Nrisa Sofar: „Wszystkie jednostki palarnie zlokalizowane u nas w wariancie obrzani. Palarnia przygotowana do spalania biomasy i odpadów komunalnych. Wszystkie jednostki palarnie zlokalizowane u nas w wariancie zwarciajskiej Sokołowej. Palarnia nie ma zdecentralizowanej jednostki palarniowej do zera od jedynie dla kolor RGD, Holiček. Biogazownie w lasachne palarny nie będące częścią standardu. Przykładowy wykazek palki paletej:

```
    }  
    return true;  
}  
public Image convertFromImage(Image image) {  
    return Image.createImage(Image.convertFromImage(image));  
}
```

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 wyświetlanym jest ekranie komputera za pomocą trójwymiarowej grafiki komputerowej.

Przykład takiego obrazu znajduje się na rysunku 2.3.

Analiza i przetwarzanie danych

- Histogram, czyli możliwość wygenerowania histogramu obrazu.

Histogram to wykres przedstawiający dystrybucję wartości numerycznych obrazu.

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

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

Przykład generowania rekonstrukcji wielopłaszczyznowej jest pokazany na rysunku 2.4

```

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 }
```

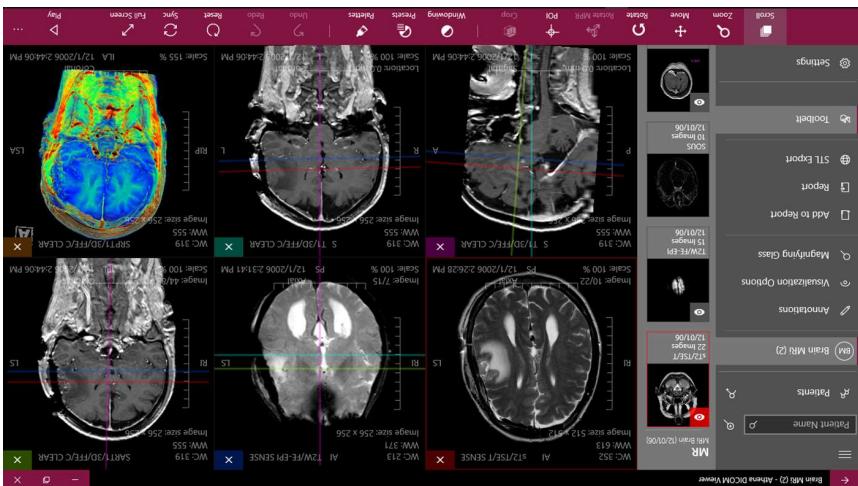
- *Sokar::Monochrome::Scene::generatePixmap()*

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

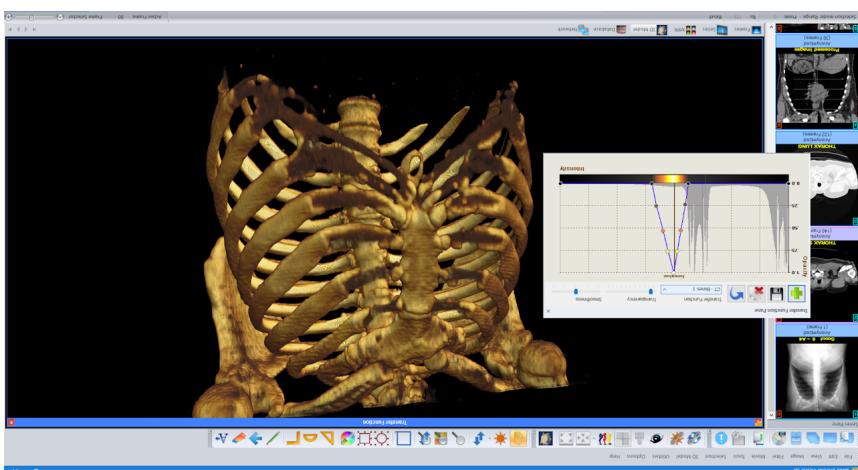
```

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 woksela 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 wielopłaszczyznowej w przegłębacie Atena DCOM Viewer. Zdjęcie uzyskane za zgodą Medical Harbour.



Hysunek 2.3: Główne obrazów 3D z wektora tomograficznego przedstawione na rysunku 2.3.



Funkcja zmiany wartości obrzaru na kolor przedstawia się następująco:

```
intline const Pixel_getPixel(quint64 value) override {  
    return *pixelArray + (pixelMove + value);  
}
```

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

ještě to hinkají, ktorá džez obraz na vatek, tvarozky je i můcham. Lásce vatekovo ještě zazápasila za posměch, ktorá džez obozají Žitá: *QThreadd:idealtthreeadCount()*. Väčšia džezifikácia na zakresťach o džezovci iloſci woksehl podzileonej prize iloſe wétkow. Krod funkcií:

```
/* Tworzenie wektora wtkow */
```

void Monochrome::Scene::genHypertextWindow()

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 woksele 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 woksele 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 funkcję „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 {
```

Studia DCOM jest oficjalnym pośrednictwem dyplomowym, tradycyjnie kontynuującym tradycję dyplomów zatrudnienia i studiów podyplomowych, stacjji do przewarzania i aktualizowania profesjonalnych umiejętności.

2.4.1 Standard DICOM v3.0

Pierwsze żonglowanie komputerowe przezyły swojego rozwijawącego się latem zdecydowanie w zakresie możliwości i możliwości użytkowania. W tym samym czasie pojawiły się nowe technologie, takie jak komputery osobiste, a także nowe aplikacje i programy, co umożliwiło rozwój komputerów domowych.

2.4 Format cytrowycz obrazów medycznych

Firmejutdárás zámvára opére rekonstrukcióját, telepítését követően azonban a működési időszakban a zámvári szolgáltatókhoz közelítve a hosszú várakozások miatt a felhasználók elutasították a szolgáltatást. A vállalatnak ezért kötelező volt megoldani a problémát, hogy megelőzze a környezetben kialakuló pozitív hatásokat.

Obrázovanie trojwymiarowe

Tatwamasiyah na die toekomsel geskeide word oos vir omkomsel deur die komende fynkji Soekar::MonochromeweWimadow::gePixel() .

$\mathbf{t}x * v - \mathbf{t}y =$

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

Poříada my těraž dva plnkytý okneka odnošazče se do wartsoci odrazu. Wyznaczone parametry prosté přezchodzacej prize dva punkty:

$ce_1/ = rescaleSlope$

ϵ_0 = rescaleSlope

$1 - \text{rescaleIntercept}$

$\sigma = \text{rescaleIntercept}$

$$AS = m/(q - snts - OutputUnits)$$

WATROSĘ określającą odnoszącą się do wartości ją wykorzystywanej, a ponieważ serią powyższe

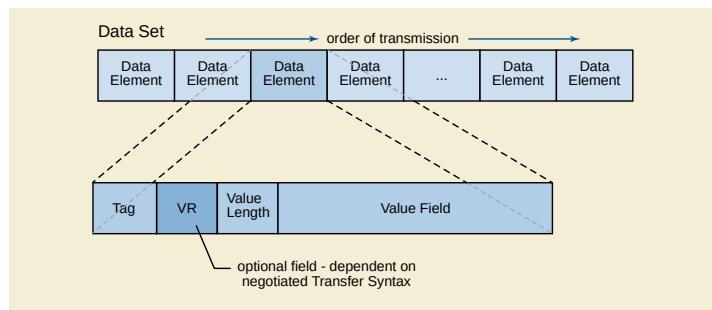
- $OutputUnits = m * SV + b$
 - m — warotśe z $\text{Diagm RescaleSlope}$ (0x0028, 0x1053)
 - b — warotśe z $\text{Diagm RescaleIntercept}$ (0x0028, 0x1052)
 - SV — stored values — wartości woksela z pliku
 - $OutputUnits$ — wartości wykładowane za pomocą gildzie:

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

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 zawiera w sobie parametr zawierający nazwę pacjenta.

Dokładne omówienie znaczników znajduje się w sekcji 2.4.2.

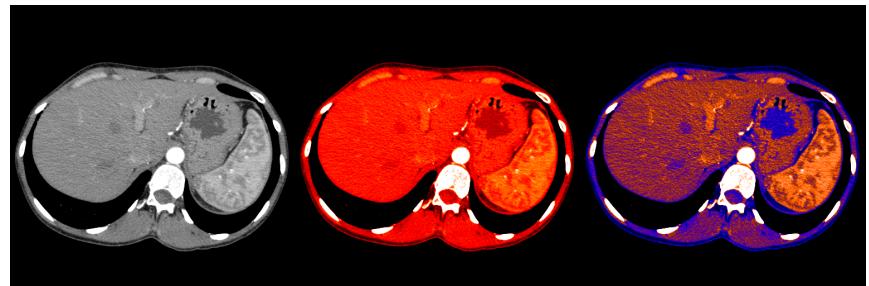
- „Value Representation”, w skrócie „VR” – to dwa bajty w postaci tekstu, informujące o formacie w jakim parametr został zapisany.

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

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

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

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



Rysunek 4.11: 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, której informacje w jakim formacie jest zapisany parametr obrazu. Składa się z dwóch bajtów.

- Tag Patient Name (0x0010, 0x0010) — nazwa Pacjenta, czysty znacznik, kiedy pacjent jest bezimienny
 - Tag Patient ID (0x0010, 0x0020) — id Pacjenta, unikalny identyfikator Pacjenta, musi istnieć pojawici. Mózgaby pusty w przypadku kiedy pacjent jest bezimienny
 - Tag Patient BirthDate (0x0010, 0x0030) — data urodzenia Pacjenta
 - Tag Patient Sex (0x0010, 0x0040) — Płeć Pacjenta
 - Tag Patient Age (0x0010, 0x1010) — wiek Pacjenta w czasie badania
 - Tag Study Description (0x0008, 0x1030) — opis badania, pole wypełniane przez technika
 - Tag Series Description (0x0008, 0x103E) — opis serii, pole wypełniane przez lekarza
 - Tag Series Instance UID (0x0020, 0x000E) — unikalny numer serii, kiedy jest nadawany kazdemu badaniu
 - Tag Instance Number (0x0020, 0x0013) — numer instancji ramki, używany w przypadku kiedy z jednego badania zostalo utworzone kilka plikow DICOM
 - Tag Modality (0x0008, 0x0060) — modalność określająca rodzaj techniki diagnostycznej
 - Tag Study Date (0x0008, 0x0020) — data wykonyania badania

Znacznik

"Lag" to umożliwić uzaciszku pozwalającą odkrycie czegoś dotyczące zapisane w elemencie danym. Znaczniik jest złożony z dwóch liczb: numeru grypy i numeru elementu.

Tag 1: Jeden zaznacza (xoo10; xoo10) — nazwa pączekini, czyż zazwyczaj musi się poszukać. Może być pusty w przypadku kiedy paczett jest bezimienny

- **Tag** Patient ID (0x0010, 0x0020) — Id Pacjenta, unikalny identyfikator Pacjenta, mającyścieli jest to numer HIS(Hospital Information System)

- Tag: I agree that the above (oxo10, oxo20) — data in other places
- Dicom DataSet Sx (0x0010, 0x0040) the paciente

- Dicrom Patient Age (0x0010, 0x1010) — wiek Pacjenta w czasie badania

- **Decom Study Description (0x0008, 0x1030)** — opis badania, pole wyprzedażne prezesa technika lub lekarza

• The **lekarz** — opis serii, pole wypromowane przez telemira

- **Series** — **uid** (0x0020, 0x000E) — unikally number series, ktoriy jest nazaway katalogu badani

- Dicom Instance Number (0x0020, 0x0013) — number instancejaci ramlki, uzywany w przekazaniu kilej z jednego badania do kolejnego itwozonych kilek Dicom

- Tag Modality (0x0008, 0x0060) — modalnoće okrešla je ka različitim tehnikama.

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

Pryzyladowe VR: Działalność struktury z dziedziny dyżutowej.

Z uwegu na koniecznoœci ostatecznie dodaœi sztybkoœci wyswietlania obranu warœto jœst szacowœ warœoci finkci f . Warœoci tej finkcij nalezy przeliczyæ, gdy zmiennœe zostanœ parametry tak zwaneego „okna“. Miedzy kolonami wyznaczany jest wtedy poprzedni warœoci z tabeli o indeksie równym warœoci numerycznej w obrazie. Umieamy w ten

Opis

Implementacja algorytmu

Rysunek 4.10: Podównanie jednego obrazu w trzech różnych „oknach”. Zdjēcie własne.



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

Wyznaczamy parametry a i b , prostymi przekształceniemi przekształcając równanie (x_0, y_0) w (x_1, y_1) . Gdzie y_0 jest rowne 0, a y_1 jest rowne 255. Funkcja „okna” wygeneruje następujące:

- 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 `Tag Dicom 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 zmienoprzecinkowa lub ciąg kilku liczb zmienoprzecinkowych 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, rozzielony 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 kontraście stosuje się twór zwany okienkiem. Jest to funkcja, która mapuje n-bitowy obraz na 8-bitowy obraz w skali szarości. Wykorzystuje się 8 bitów ponieważ monitor RGB jest w stanie wyświetlić 256 odcieni szarości.

Zwiększenie kontrastu za pomocą „funkcji okna”

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

$$x_0 = center - width/2$$

$$x_1 = center + width/2$$

W tomosgrali komputerowej wyklikiem rekonstrukcji jest macierz liczb opisujących prawne i medyczne, niezwykle istotna w dziedzinie przetwarzania promieniowania. Ze tego powodu producent sprzecząc się zapisy optymalizacyjne dla DICOM w formacie plików cyfrowych. Wykorzystując dane o przebiegu promieniowania, można określić parametry warunków rozkładu przestrzennego współczynników oślabiania promieniowania. Ze względu na aspekty prawne i medyczne, niezwykle istotna jest zapisy optymalizacyjne dla DICOM w formacie plików cyfrowych. Wykorzystując dane o przebiegu promieniowania, można określić parametry warunków akwizycji itp.

2.4.4 Inne formaty zapisu

W przypadku węzłów zapisanych w formacie pozycji są problemy zidentyfikowane w przeszłości. Wyszukanie konkretnego badania lub pliku w folderze, w którym znajdują się pliki zapisywane, jest rzadkością. Dlatego standard DICOM definiuje nowe pliki typu DICOMDIR, który jest plikiem indeksacyjnym. Dla każdego pliku DICOM w folderze, Powszechnie przechowywanie plików pozycji jest problemem. Dla każdego plikuDICOMDIR, który jest plikiem indeksacyjnym, jest klaster plików poprzecznych, w których znajdują się pliki DICOM, które mają identyczne nazwy. Dla każdego plikuDICOMDIR, który jest plikiem indeksacyjnym, jest klaster plików, w których znajdują się pliki DICOM, które mają identyczne nazwy. Oznacza to, że wartości plików są ulokowane w taki sposób, że każdy plik posiada identyczny wzorzec nazwy.

2.4.3 DICOMDIR

Zwykły tekst o długosci maksymalnej 2³² – 2 bajtów.

- UT — Unlimited Text — tekst o nieograniczony długosci.
- US — Unsigned Short — 16-bitowa liczba całkowita ze znakiem następującym: „Nowak_Jan_Prof. dr. hab. inż. Jan Nowak pracownik ZEJIM”.
- SS — Signed Short — 16-bitowa liczba całkowita bez znaku następującym: „Nowak_Jan_Prof. dr. hab. inż. Jan Nowak pracownik ZEJIM”.

Dlugość jednego fragmentu powinna maksymalnie wynosić 64 znaki. W przypadku mniejszego ilości segmentów, many zakończe, że są puste.

- name suffix — suffiks po imieniu, brak odpowiadanka
- name prefix — prefiks przed imieniem, np: mgr. inż.
- middle name — środkowe imię, brak odpowiadanka w polskim nazewnictwie
- given name complex — imię, np: Adam
- family name complex — nazwisko, np: Smolik

YBR także YCC, to model przestroni kolorów do przedstawiania obrazów wideo. Wykorzystywany jest uzyjowany przez wiele technik warstwowych. YBR nie pokrywa całego zakresu chrominacji Y-B, stanowiącą różnicę między luminancją a czerwoną. Y-B — skadowa chromatyczna Y-B, stanowiąca różnicę pomiędzy luminancją, oraz R — żółciową chromatyczną Y-B, stanowiącą różnicę pomiędzy luminancją a niebieskim, oraz B — niebieską chromatyczną Y-B, stanowiącą różnicę pomiędzy luminancją a czerwoną.

YBR

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

- B_n — wartości indeksujące kanały
- G_n — wartości indeksujące kanały
- R_n — wartości czerwonego kanału

gdzie:

$H_1, H_2, H_3, H_4, \dots, G_1, G_2, G_3, G_4, \dots, B_1, B_2, B_3, B_4, \dots$

- I — oznacza to, że wartości plików są ulokowane w taki sposób

$H_1, G_1, R_1, G_2, R_2, H_3, G_3, B_3, G_4, B_4, \dots$

- 0 — oznacza to, że wartości plików są ulokowane w taki sposób

Obrázek zapisany w RGB nie trzeba wcale sposobu obrabiać, dane już są prawidłowe do wyświetlenia. Należy je odpowiednio posortować, jeżeli zaczodzi taka postać. Sposób posortowania wartości plików określają konfigurację trzeciego argumentu sortowania. Jeżeli jest to ostatni argument, to wartości pozycyjne pozostaną bez zmiany. Mając na przykład dwie wartości pozycyjne wartości (0x00028, 0x0006). Oznacza to, że wartości plików zaczynają się od końca pliku określonego przez ostatni argument. Jeżeli ostatni argumentDICOM Configuration Tag jest określony, to wartości pozycyjne pozostaną bez zmiany.

RGB

Obrázek monochromatyczny to obraz w określonej szarości, od której do czarnej lub od białej. Generowanie takiego obrazu odbywa się poprzez pseudokolorowanie. Czyli proces jest wykonywany w sekcji 4.6.2.

Generowanie pozycyjnych typów obrazów jest wykonywane poiniogej.

Generowanie pozycyjnych typów obrazów jest wykonywane w funkcji **QImage::operator=(QImage)**. Funkcja wywoływana jest implementowana w funkcji **QImage::operator=(QImage)** i obsługuje **pixmaps**.

Cała obszarówka obrazu jest implementationami funkcji **operator=(QImage)**. Po wywołaniu funkcji obiekt **targetBuffer** powinien zawierać obraz wygenerowany z określonym parametrami. Funkcja zwieracą powinna wartości logiczne, które informują nas czy **targetBuffer** rzeczywiście zostało zmieniony. Następnie obiekt **Pixmap** jest na nowo generowany na bazie **Image**.

- **iconPixmap** — obiekt obrazu ikony, klasa **Qt::QPixmap**, docelowo powiniene mieć 128 pikseli na 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 organizację Qt Project, jest zbiorem bibliotek i narzędzi programistycznych dedykowanych dla języków C++, QML i Java.

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

W tej pracy wybrano biblioteki Qt z uwagi na to, że posiada interfejs w C++. 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 funkcję `Sokar::About::GDCM()`
- About Sokar — otwiera okno z informacjami o aplikacji, implementowane przez funkcję `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.

Ważne zasady bezpieczeństwa:

- Właściwość `getLastError()` — pozwala na uzyskanie informacji o błędach.
- Właściwość `getLastErrorString()` — pozwala na uzyskanie szczegółowej informacji o błędach.
- Właściwość `getLastErrorCode()` — pozwala na uzyskanie kodu błędu.
- Właściwość `getLastErrorType()` — pozwala na uzyskanie typu błędu.
- Właściwość `getLastErrorFile()` — pozwala na uzyskanie nazwy pliku, w którym wystąpił błąd.
- Właściwość `getLastErrorLine()` — pozwala na uzyskanie numeru linii, w której wystąpił błąd.
- Właściwość `getLastErrorColumn()` — pozwala na uzyskanie numeru kolumny, w której wystąpił błąd.
- Właściwość `getLastErrorFunction()` — pozwala na uzyskanie nazwy funkcji, w której wystąpił błąd.
- Właściwość `getLastErrorThread()` — pozwala na uzyskanie nazwy wątku, w którym wystąpił błąd.
- Właściwość `getLastErrorProcess()` — pozwala na uzyskanie nazwy procesu, w którym wystąpił błąd.
- Właściwość `getLastErrorModule()` — pozwala na uzyskanie nazwy modułu, w którym wystąpił błąd.
- Właściwość `getLastErrorCategory()` — pozwala na uzyskanie kategorii błędu.
- Właściwość `getLastErrorDescription()` — pozwala na uzyskanie opisania błędu.
- Właściwość `getLastErrorSource()` — pozwala na uzyskanie źródła błędu.
- Właściwość `getLastErrorData()` — pozwala na uzyskanie dodatkowych danych.

3.2.4 Globalne typy struktury

Widoczne kategorie struktur:

- ISO 9001:2015
- IEC 61508:2010-3 7.4.4 (SIL 3)
- IEC 62304:2015 (2006 + A1)

The Q^t Company posiada szereg certyfikatów od FDA i UE, które ułatwiają prowadzenie produkcji bezpiecznej dla rynku medycznego.

3.2.3 Normy i certyfikaty

GNU General Public License wersji 3, co pozwala na uzycie tej biblioteki mojej pracy. Wersja o której mowa nie posiada wielu możliwości, ale jest dostępną dla licencji.

Biblioteka Qt jest dostępujona w dwóch wersjach: komercyjnej i otwartej zgodnie z licencją.

3.2.2 Licencja

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

Wiedząc autorów, Qt powinno się czyste jak angielskie słowo „cute”, po polsku „kutu”.

Jednakże społeczeństwo programistów nie jest co do tego zgodna. Ankiety zrobione na strukturę pliku w systemie (opisane w 4.5.4), menu programu (opisane w 4.5.6), lub poprzecznym pliku, natomiast trzecim sposobem można wyciągać ją po pojęciu przekątnej i upięzczenie pliku. Z wolą przewszystkich moźna wyciągać ją i wleć plik.

Odpomnikli do przyczynyankiet:

3.2.1 Wyjoma

Kilka przykładow:

Dla tegoż programu można uzyskać typy fundamentalne dostarczane przez bibliotekę Qt.

Na przykład, aby uzyskać informacje o pliku, należy skorzystać z funkcji `QFile::open()`.

Warto zwrócić uwagę, że dla plików zapisujących dane, funkcja `QFile::open()` powinna być wywoływana po utworzeniu pliku.

• File Recent — program zapamiętuje ostatnio wczytane pliki i pozwalają na ich ponowne wczytanie z tego samego menu.

• Open — otwiera okienko wyboru pliku, implementowane przez `QFileDialog::getOpenFileName()`, nastałe w formacie JPEG, BMP, GIF lub PNG. Zapisywane jest nazwisko obrazu do pliku.

• Export as — zapisać obrazu w formacie JPEG, BMP, GIF lub PNG. Zapisywane jest nazwisko obrazu do pliku.

• Help — wyjaśnia aplikacji z aplikacji.

W głównej części okna programu znajdują się menu, obiekt klasy `QMenuBar`. Struktura menu programu:

- File — menu programu, który implementuje przesunięcie pliku.
- Open — otwiera okienko wyboru pliku, implementowane przez `QFileDialog::getOpenFileName()`.
- Save — zapisuje ostatnio wczytane pliki i pozwalają na ich ponowne wczytanie z tego samego menu.
- Exit — wyjście z aplikacji.

W lewej części okna znajdują się elementy listy implementowane przez `QListWidget`.

Zawiera ona siedem modeli danych plików systemu, który z kolei jest implementowany przez klasy `QFileSystemModel`. Po wybór pliku sciezka jest przesyłana do obiektu zakladki.

W środkowej części programu znajdują się obiekty zakladki opisane w zakladce.

Przez zakladkę można dodać nowy element listy implementowanej przez `QListWidgetItem`.

Zauważ, że nowy element ma strukturę pliku, obiekt z zakladki.

W prawej części okna znajdują się elementy listy implementowanej przez `QListWidget`.

Główne okno programu jest implementowane przez `QMainWindow`. Jest wywoływanie przesunięcia pliku.

Wczytywanie pliku:

Otworzanie pliku może odbyć się za pomocą parametru `i Sokar::DicomTables::addDicomFiles()`. Każda z tych funkcji ma dwa parametry: nazwę pliku i nazwę tabeli.

Wyjątkiem jest tabela `i Sokar::DicomTables::addDicomFile()`, która posiada dwoje argumentów: nazwę tabeli i nazwę pliku.

Po dostarczeniu sciezki do obiektu, pliku zostaje wczytana za pomocą funkcji `qdcm::DicomTables::DicomTables::create()`, opisanej w sekcji 4.5.3.

Spis treści nowych plików:

- `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.

Syntax is sloty

```

int main() {
    // Tworzymy obiekt Przycisku
    auto *guit = new Przycisk("Guit");
    // Tworzymy obiekt okna
    auto *window = new Okno(guit);
    // Przypisujemy dziedziczenia przyciskowi
    window->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 się o ich dodatkowe wskazówka lub wektora wskaźnikow w deklaracji klasy, a dzięki temu można poznajesz spójność. Jest to o typie efektywnie, że nie trzeba dla klaszego obiektu tworzyć dodatkowej wskazówki.

- Hospital Data — Dame szpitala
- Akcja: **HospitalData**.
Po otzymaniu sygnalnego obiektu klasы *Sokar::HospitalData* zaznaczamy ją i klikamy w zakładce *Properties* w sekcji *Modality Indicator* zaznaczamy pole *Acquisition* i klikamy przycisk *Apply*.
- Image Acquisition — Dame akwizycji
Po otzymaniu sygnalnego obiektu klasы *Sokar::Image* zaznaczamy ją i klikamy w zakładce *Properties* w sekcji *Modality Indicator* zaznaczamy pole *Acquisition* i klikamy przycisk *Apply*.
- Tagi (5)
 - Akcja: **Modalities**.
Po otzymaniu sygnalnego obiektu klasы *Sokar::ModalityIndicator* zaznaczamy ją i klikamy w zakładce *Properties* w sekcji *Modality Indicator* zaznaczamy pole *Acquisition* i klikamy przycisk *Apply*.
 - Akcja: **OpenDataset**.
Klikając teżgo przycisku wyśle proszę o otworzenie okna ze wszystkimi elementami danego pliku obrazu, który jest obecnie wyświetlany na scenie.

Ten element portra wylaczyc wszystlane niektorych elementow na scene. Kilkumiecie go odznacza lub zaznaca wszyskie pozycje w menu kontekstowej. Wszystkie pozycje sa pojedynczo dzuznaczany.

Menu rozwijalne:

- Patient Data — Dame pacjenta
- Akcja: PatientData.

Po otwym menu sygnal obiekt klasy *Sokar::PatientDataIndicator* zazdrujacy sie na scene powinien pokazac lub ukryc sie w zaleznosci od stanu pozycji.

- Rotate Left — Obrot w lewo
- Rotate Left90 — Obrot w lewo o 90 stopni
- Flip Horizontal — Odwróć lustrzanie poziomo
- Akcja: **FlipHorizontal**.
- Po otrzymaniu sygnału obraz na scenie powiniene obrócić się o 90 stopni w lewo.
- Po otrzymaniu sygnału obraz na scenie powiniene obrócić się o 180 stopni w lewo.
- Akcja: **RotateLeft90**.
- Flip Vertical — Odwróć lustrzanie pionowo
- Akcja: **FlipVertical**.
- Po otrzymaniu sygnału obraz na scenie powiniene odwrócić się w lustrzanie poziomo.
- Akcja: **ClearTransformation**.
- Wyczęście przekształcenia obrotu

```

1 #include <QObject>
2
3 class Counter : public QObject {
4     /* Każda klasa dziedzicząca po QObject musi na samym
5      * poczatku swojej definicji miec 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 byc poprzedzone makrem "slots".
14     * Widoczosc slotow moza zmieniac. */
15 public slots:
16    void setValue(int value){
17        if (value != m_value) {
18            m_value = value;
19
20            /* Podczas wywozywania sygnału nalezy
21             * poprzedzic to makrem "emit". */
22            emit valueChanged(value);
23        }
24    }
25
26    /* Sygnały powinny byc poprzedzone makrem "signals".
27     * Wszystkie sygnały sa 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, wyświetlania 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 wcisnięcia przycisku jest przez obsługę sygnału *Qt::AbstractButton::clicked()*. Przykład można zobaczyć na rysunku 3.1.
- *Qt::QTabWidget* — implementuje zakładki, takie jak w przeglądarce internetowej. Dziedziczy bezpośrednio po klasie *Qt::QWidget*. Zawartości zakładek mogą być 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::ToolBar*. Posiada on zespół ikonek z rozwijalnymi menu kontekstowymi.

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

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 wcisnięty 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 wcisnięty 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 Luminx, MacOS i Microsoft Windows
Ostatczne podjeto deczyze o wyborze biblioteki o nazwie Grassroots DICOM (GDCM),
dosetpna pod adresem <http://gdcm.sourceforge.net/>.

- mieuje licencje powallażakę jesi używawc w potrzebnyim zakresie
 - hyc darmowa, nadjepieli otwarto zdrojowa
 - hyc aktynne rozwijsana — značzna wiekszoſe bibliotek charakteryzowalna sie tyim, ze bŷta porzuciona i ostatečna zmaia bŷta wprawdzońa wiele lat temu, a proces jeli rozwój trwał od 2 do 5 miesięcy

Znalezienie dobréj biblioteki do odsyłania nie jest prostie, ponieważ jest ich bardzo dużo, a iżch liczbą wieleż rośnie. Powstał portal internetowy do tego celu nazywany "I Do MAGINC", dostępny pod adresem <https://idoimageing.com/programs>. Biblioteka, której poszukiwanie w tej pracy powinna:

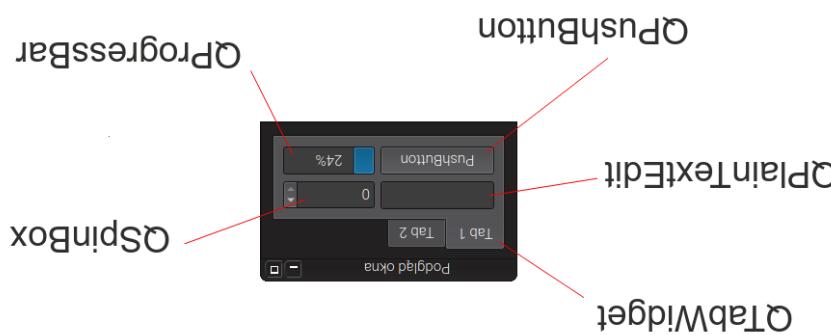
3.3.1. Uzasadnienie wyboru

3.3 GDCM

W latawy sposob zwiększyć lub zmniejszyć zawartość. Przykład mówiący na przykładzie użycia rozwojnika. Rosada dwa dodatkowe przykłady poważające yskimka 3.1.

- QProgressBar** — implementuje przeklikę, czyli kontrolkę przytroszoną do wpro-
wadzania wartości. Działać będzie bezpośrednio po klasie **QWidget**. Przykładowa pozycja paska postępu w widoku wersja z pozycją 31.

cytus 3:1, 17; *zygadole* or *zygadole* program w/ *cyt.* zjadćce wiśnie;



The screenshot shows a window titled "Zakladka". Inside, there are three main components: a **QTabWidget** with two tabs labeled "Tab 1" and "Tab 2"; a **QSpinBox** with the value set to 0; and a **QLineEdit** containing the text "Podglad okna". Red arrows point from the labels to their respective components.

Rysunek 4.9: Wykiad zakladki wraz z numeracją elementow interfejsu. Zdjcie wlasne.



Dodatakowo posiada obiekt katalogi scen opisana w sekcji 4.5.3.

Dicom Vitez,
Kazachstan' z odrzazm jese' imprezowana prez' rias'e sorkat'.
Dicom Vitez
Mterfejsy graqczny Scholar@ Dicomo Vitez wyswieta nastpujace elementy:

Изучение языка включает в себя изучение грамматики, лексики, фонетики и т.д.

www.suzanndean.com

Sergięgaśca abywua się za pomocą aplikacji DicomFileSet: create(). Do funkcji jest przesyłany wektor z wczytanymi plikami DICOM, nastąpiła dezeria ana pliki na zbiory zwarcia wektora te samej serii, tworzy obiektu zbioryw DICOM. Ostatecznie zwarcia zapisze jazdzieca takiż samym serii, nastąpiła dezeria ana pliki na zbiory zwarcia wektora z kolejnością obiekta zbioryw DICOM. Sortowanie plików DICOM wedle nazwy ich kolejności obokwya się za pomocą aplikacji std::sort we właściwa klasę Skała... DocomFileSet: sortByFileIndex();

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 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 wczytanych plików DICOM. Na początku pliki są sortowane na podstawie liczby zawartej w elemencie o znaczniku *Dicom Tag Instance Number* (0x0020, 0x0013). Dla każdego pliku jest tworzony obiekt *Sokar::DicomFrameSet*.

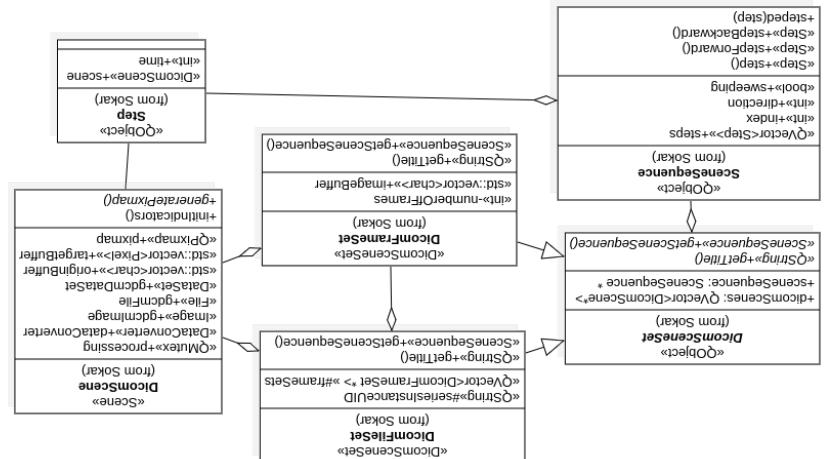
Sekwencja jest tworzona poprzez połączenie sekwencji poszczególnych obrazów.

Segregowanie obrazów

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

Sekwencja scen

Rysunek 4.8: Diagram klas UML dziedziczenia klasy `Sokar::DicomSceneSet`.



Ponizej zaprezentowane kilka przykładow użycia biblioteki GDCM.

3.3.5 Przykład użycia

- `gdcn::StringFilter` — поинтническая линия
 - `gdcn::Tag` — объект знания
 - `gdcn::DataElement` — объект элемента данных
 - `gdcn::Dataset` — объект збора элементов

Abstractyjna klasa **Sokar**: **Lichomscenecset** implementuje wektor scen za pomocé klasí **Q::Vector**. Ješt to obiect, který przechováve scény i tworzy skenovací scén. Ktora ješt **seznam** ramek otrázav. Sú dve implementacie kolokáži sestava: kolokážia plikov ! kolokážia ramka z jednego pliku. Diagram klas **UML** nazadujúc sié na **rysunku 4.8**.

4.5.3 Rolekje scen

Przykład wczytania pliku

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

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

– „KVP” — szczytowe napięcie wyjściowe generatora promieniowania rentgenowskiego wyrażone w kilovoltach, pobierane z ^{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} SAR (0x0018, 0x1316).

Generowanie obrazów z danych

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

Przekształcenia macierzowe obrazu

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

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

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

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

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

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 ^{Dicom} 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 ^{Dicom} 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:

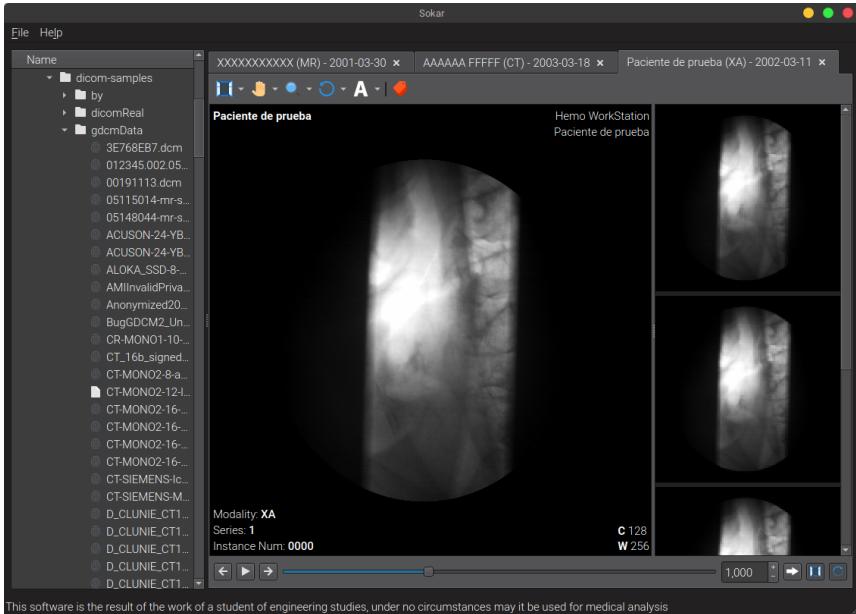
- Nazwa instytucji
Tekst jest obierany z ^{Dicom} Tag Institutional Department Name (0x0008, 0x1040) i wyświetlany bez ingerencji.
- Producent wyposażenia wraz z modelem urządzenia
Tekst jest obierany z ^{Dicom} Tag Manufacturer (0x0008, 0x0070) i ^{Dicom} Tag Model Name (0x0008, 0x1070), oddzielony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

Orientacja obrazu

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

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

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



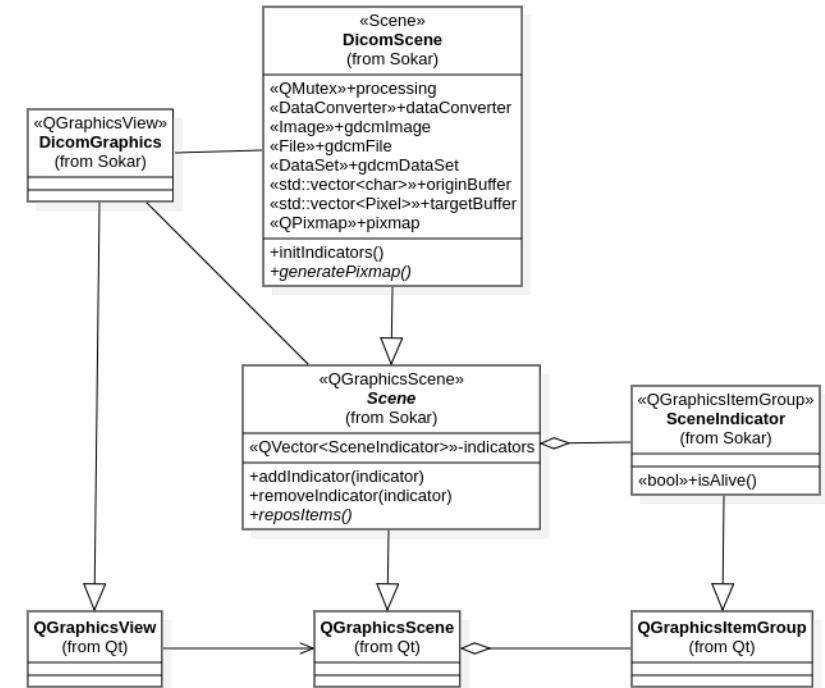
Rysunek 4.2: Okno przeglądarki z wczytanymi kilkoma obrazami. Zdjęcie własne.

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

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

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

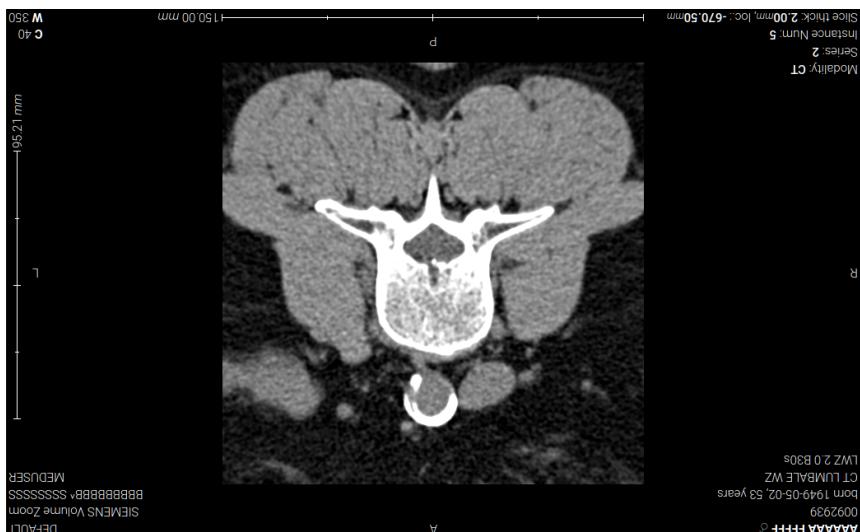
Przykładowa scena z obrazem monochromatycznym znajduje się na rysunku 4.3.



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

4.4 Projekt struktury obiektowej programu

Moziwosć wyzwolona ammacyjna poszukiwała się wtedy, gdy wędle zakłócały bieżącą zanadpływą się rytmie nizi jedna ramka obratu. Młodza to osiągnięcie wyzujących while obratu z te samej serii lub wczystą obratą pozostały wele ramki. Wówczas pod seńną powiatu się pasieki, umieliśmy ją sterować niezmienniczą, a po prawie strome obrót z konarami poszczęgólnych ramek obratu. Dokładny opis fizyczny i ich funkcji zanadpływą się w sekci 4.5.A.

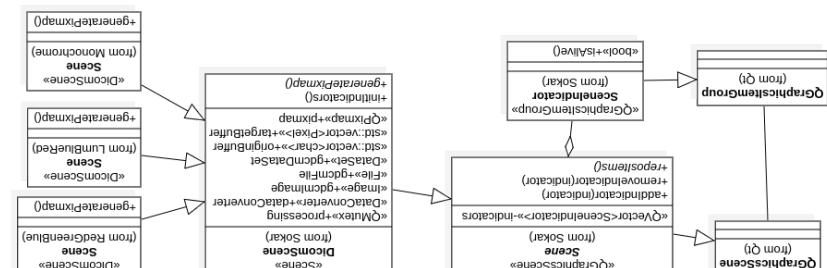


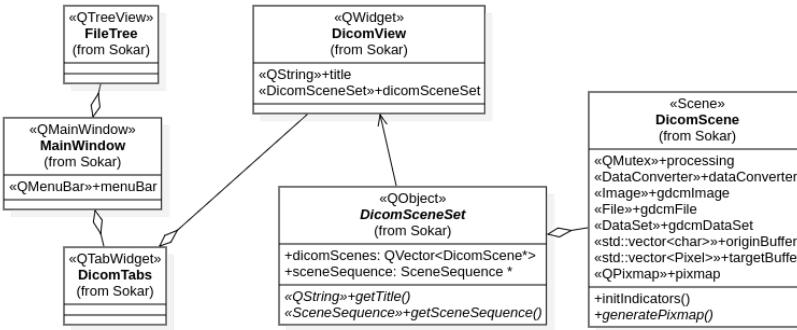
Rysunek 4.3: Przykładowa scena z obrazem monochromatycznym. Zdjęcie własne.

- **Qt::GraphicsScene** — element grupy **QGraphicsGroup** który wiele elementów. Pozwala na łatwe implementację bardziej złożonych struktur
 - **Qt::GraphicsPixmapItem** — element wyswietlający obrázky graficzne, obiekty klasy
 - **Qt::GraphicsClassItem** — element wyswietlający prostą linię z punktu A do B

- *Qt::GraphicsEffect* — element wyświetlający tekst. Obsługuje on m.in. wiele efektów, takich jak przeskalowanie, przekształcanie macierzy, zmiany koloru itp.

Rysunek 4.5: Diagram klas UML przedstawiający klasę `Sofar`: `DecomScene`.





Rysunek 4.4: Diagram klas UML globalnej struktury programu.

4.5 Struktury danych

4.5.1 Konwertowanie danych ze znaczników

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Oprócz powyższych funkcji jest jeszcze kilka innych funkcji pomocniczych oraz kilka aliasów.

Ogólne zasady konwersji, które dotyczą wszystkich danych:

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

4.5.2 Scena

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

Wyświetlanie sceny

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

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