

Warszawa 2019

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

Adam Jędrzejowski
nr albumu 277417

Wieloplatformowa przeglądarka obrazów DICOM w C++

w specjalności inżynierii oprogramowania
na kierunku Elektronika
Praça dyplomowa
inżynierska

Zakład Elektroniczny [Zakład Medyczny]
Instytut Radiowej Elektroniki i Technik Multimedialnych



POŁITECHNIKA WARSZAWSKA
WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMATYCZNEJ

Wieloplatformowa przeglądarka obrazów DICOM w C++

Praca składa się z sześciu rozdziałów: wstęp; obrazowanie diagnostyczne w medycynie; biblioteki i narzędzia; implementacja; kompilacja; podsumowanie. Wstęp jest powierzchownym wprowadzeniem do tematu i celu pracy.

W drugim rozdziale jest opisane zagadnienie problemowe związane z obrazami w medycynie. Wymieniono techniki diagnostyczne oraz ich podstawowe różnice między sobą. Przedstawiono parametry jakie cyfrowy obraz medycyny posiada. Opisano prezentacje obrazów medycznych. Wyjaśniono czym są przeglądarki obrazów, jakie funkcje mogą posiadać i jakie kryteria wyróżniono do ich porównywania. Opisano format zapisu cyfrowych obrazów medycznych, standard DICOM.

Trzeci rozdział opisuje biblioteki i narzędzie użyte w czasie pisania pracy inżynierskiej. Wyjaśniono 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 załączono diagram UML. Opisano wszystkie algorytmy przetwarzania danych w celu lepszej wizualizacji obrazu.

W piątym rozdziale opisano przebieg kompilacji kodu źródłowego.

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

Wieloplatformowa przegładaarka obrazów DICOM w C++

Na angielski przetłumacze jak będzie po polsku gotowe.

Keywords: DICOM; DICOM viewer; 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: Lupa w przegądarkę MedDream DICOM Viewer. Zdjęcie użyte	9
2.2	Wyswietlalne wilem obrzędów na raz w jednym oknie w przegądarkę Same za szkodą Software ABB. Zdjęcie użyte	10
2.3	DICOM Viewer 3D Pro. Generowana obrzędów 3D z wilem obrzędów tomograficznych w przegądarkę Same DICOM Viewer 3D Pro. Zdjęcie użyte	11
2.4	Rekonstrukcji wilejopaszczyznowej w przegądarkę Athens DICOM Viewer. Zdjęcie użyte za szkodą Medical Harbou.	11
2.5	Elementy danych w żargonie elementów danych. Zdjęcie ze standardu DICOM dostępu pod adresem http://dicom.nema.org/medical/dicom/2019a/ output/chart1/part05/chapter_7.html.	14
3.1	Przekladowe okienko programu w Qt. Zdjęcie własne.	23
4.1	Okno przegądarki z wyciągniętym kilkoma obrzędami. Zdjęcie własne.	29
4.2	Okno przegądarki tuz po uruchomieniu. Zdjęcie własne.	30
4.3	Przekladowa scena z obrzędem monochromatycznym. Zdjęcie własne.	31
4.4	Przekladowa scena z obrzędem monochromatycznym. Zdjęcie własne.	32
4.5	Diagram klas UML dziedziczenia klasą <i>Szakar::DocomScene</i> . Zdjęcie własne.	34
4.6	Diagram klas UML dziedziczenia klasą <i>Szakar::DocomScene</i> . Zdjęcie własne.	35
4.7	Diagram klas UML dziedziczenia klasą <i>Szakar::SceneIndicator</i> . Zdjęcie własne.	36
4.8	Diagram klas UML dziedziczenia klasą <i>Szakar::DocomSceneSet</i> . Zdjęcie własne.	40
4.9	Wykład zakladki wraz z numeracją elementów interfejsu. Zdjęcie własne.	42
4.10	Porównanie żadnego obrzędów z trzech różnych „filmek kina”. Zdjęcie własne.	50
4.11	Palera Hot Iron (na stołku) i Hot Metal Blue (po prawej) w porównaniu do pełnej w skali szarości (po lewej). Zdjęcie własne.	51
4.12	Wizualizacja układu osi współrzędnych kartezjańskich pacjenta. Zdjęcie własne.	59
4.13	Podział płaszczyzny sceny. Wyrożnione osiem części. Zdjęcie własne.	62
4.14	Przekladowy obrzad medyczny (przekrój głowy MRI) z oznaczeniem orientacyjnym.	63
4.15	Obrazu za pomocą liter A, B, C, D, E, H. Zdjęcie własne.	68

Swiadom odpowiedziałoscí prawnie oswiadcza, że niniejsza praca dyplomowa inny-
nierska pt. Wieloplatformowa przegądarka obrzędów DICOM w C++:

OSWIADCZENIE

Wydział Elektroniki i Technik Informacyjnych Politechniki Warszawskiej
POLITECHNIKA WARSZAWSKA

Warszawa, 30 lutego 2019

Rozdział 6

Podsumowanie

Celem pracy inżynierskiej było napisanie aplikacji do obsługi obrazów DICOM w C++ z możliwością kompilacji na wiele platform. Cel udało się osiągnąć. Zniesienie ograniczeń wirtualizacji kodu rozwiązaño językiem programowania C++. Zastosowano biblioteki dostępne na różnych platformach: Qt i GDPCM, które również zostały napisane w C++, dzięki czemu uzyskano jednolity program napisany w jednym języku. Zapewniono jednolity sposób kompilacji na platformach przy użyciu narzędzia CMake. Dzięki czemu aplikacja działa w ten sam sposób na wszystkich testowanych platformach: Linux, MacOs i Windows. Jednolity wygląd aplikacji zapewniła biblioteka Qt, dzięki czemu 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 przekształceniami macierzowymi.

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

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

W przypadku Linuksa i MacOS, plik binary znajduje się „/path/sokar-app-bin/Sokar”. W przypadku Moźna go uruchomić klawiszem „lub z terminala za pomocą komendy „./Sokar”. W przypadku Windowsa plik binary znajduje się w folderze „/path/sokar-app-bin/Debug”.

Spis treści

W przypadku Linuksa i MacOS, uruchom „make” w folderze „/path/gdcm-bin/Sokar_sln” i kliknij przycisk „Generuj”. W przypadku Windowsa otwórz plik „/path/gdcm-bin/Sokar_sln” i kliknij przycisk „Lokalny debugger Windows”.

W przypadku Linuksa i MacOS, uruchom „make” w folderze „/path/sokar-app-bin/”.

W przypadku Linuksa i MacOS, uruchom „make” w folderze „/path/sokar-app-bin/”.

„Generuj”, „otwórz” na sekundę do skompilowanej biblioteki Qt. Następnie kliknij przycisk „Lokalny debugger Windows”.

4 Implementacja	28
4.1 Zakres implementacji	28
4.2 Wieloplatformowość	29
4.3 Graficzny interfejs użytkownika	29
4.4 Projekt struktury obiektowej programu	31
4.5 Struktury danych	32
4.5.1 Konwertowanie danych ze znaczników	32
4.5.2 Scena	33
4.5.3 Kolekcje scen	40
4.5.4 Zakładka	42
4.5.5 Obiekt zakładek	45
4.5.6 Okno główne programu	46
4.6 Algorytmy	47
4.6.1 Cykl generowania obrazów	47
4.6.2 Generowania obrazu monochromatycznego	49
4.6.3 Tworzenie transformat i ich użycie na obrazie	56
4.6.4 Ustalanie pozycji pacjenta względem sceny	59
5 Kompilacja	64
5.1 Narzędzia potrzebne do kompilacji	64
5.2 Biblioteki potrzebne do kompilacji	64
5.2.1 Instalacja Qt	64
5.2.2 Pobranie kodu źródłowego GDCM	65
5.2.3 Pobranie kodu źródłowego Sokar	65
5.3 Przygotowanie katalogów	65
5.4 Kompilacja GDCM	65
5.5 Kompilacja Sokar	65
5.6 Uruchomienie	66
6 Podsumowanie	67

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 odnaleźć wszystkie inne opcje, nie są one wymagane do kompilacji programu. Dalej należy postępować zgodnie z instrukcjami pojawiającymi się na ekranie.

5.2.2 Pobranie kodu źródłowego GDCM

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

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

5.2.3 Pobranie kodu źródłowego Sokar

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

5.3 Przygotowanie katalogów

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

5.4 Kompilacja GDCM

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”.

W przypadku Linuxa i MacOS, uruchom „make” w folderze „/path/gdcm-bin/”. W przypadku Windowsa otwórz plik „/path/gdcm-bin/GDCM.sln” i kliknij przycisk „Lokalny debugger Windows”.

5.5 Kompilacja Sokar

Uruchom CMake z menu programów lub za pomocą polecenia „cmake-gui”. W polu „Where is the source code:” wpisz „/path/sokar-app/”. W polu „Where to build the binaries:” wpisz „/path/sokar-app-bin/”. Kliknij „Configure”. W oknie zatytułowanym „CMakeSetup” wybierz takie same opcje na w GDCM. Kliknij „Finish”. Ustaw parametr

Wstep Rozdzial 1

5.1 Narzézia Potrzébne do kompliacji

- Windows — Visual Studio w wersji 2017 lub nowszej
 - Linux — pakiet zawierający nasze pliki komendy: make; cmake (w wersji 3.10 lub nowszej), g++ (w wersji 8 lub nowszej)
 - MacOS — Xcode w wersji 10 lub nowszej

5.2 Biblioteki potrzebne do komplikacji

Zaręczystowanie obawy może być zapisane w formacie zdefiniowanym przez centra. Najczęściej istnieje możliwość zapisu danych w formacie DICOM (Digital Imaging and Communications in Medicine). Obok obrazów w pliku danych zapisywanie jest w systemie parametry badania takie jak warunek akwizycji, nastawy uzadzienia, pozycja pacjenta wraz z komunikacją w MediCine.

5.2.1 Instalacija Qt

Linux

Windows i MacOS

We will install a package that adds support for the `git` command. This will allow us to use `git` to manage our codebase.

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

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

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

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



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

Obrazowanie diagnostyczne w medycynie

- Instigeje wilej technik образowania wykorzystującą rożne zasady fizyczne zachodzące w materii. Podstawa owe techniki образowania ma zasadniczo dwa zadania: aby zacząć

• Radiografia — RTG

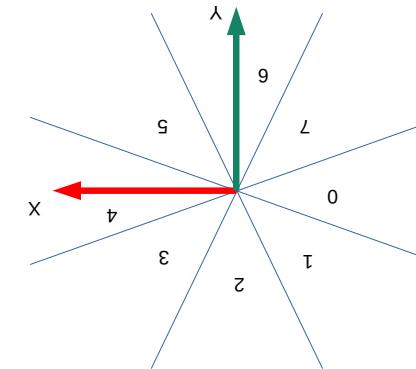
AKLwizyczja w tomografii komputerowej jest podobna do badania RTG, ale w CT wykonywana jest podobnie jak w tomografii pozycjacyjnej, w której pomiarów wzajemnych obiektu badanego i pod-

- Komputerowa tomografia (Computer Tomography — CT)

W standardzie DICOM radiografia cyfrowa jest ozaczana jako "RT".

Pryzylał mowna z obyczajem na rybunku 4.14. Na obrzeże widiżmy, że lewa strona pacjenta zanajduje się po prawej stronie obrzązu, prawą stroną pacjenta po lewej, góra pacjenta na górnym czesci obrzązu. Wyklinka z tegoż, ze obrzą przede wszystkim pacjenta skierowana jest w prawą stronę obrzązu, prawą stroną pacjenta po lewej, góra pacjenta na

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



- lewe pole: tytul czesci 1, tytul czesci 0 i tytul czesci 1
 - gorne pole: tytul czesci 1, tytul czesci 2 i tytul czesci 3
 - prawe pole: tytul czesci 3, tytul czesci 4 i tytul czesci 5
 - dolne pole: tytul czesci 7, tytul czesci 6 i tytul czesci 5

Praktikum prezentacji i kierunki głowy, zasad fizycznych do cześci I i odpowiadających „L”, do cześci II, „R” do cześci III i „P” do cześci V. Punkty „A”, „P”, „L” i „R” oznaczają punkty zanikające w przestrzeni. Wszystkie te punkty znajdują się na skrótku. Do lewego pola wstawiany jest tekst „HL”, do południowego „HR”, do prawego „PF”, i do dolnego „LP”.

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

W standardzie DICOM technika jest oznaczana skrótwcem „CT”.

- Obrazowanie metodą rezonansu magnetycznego — MRI

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

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

- Ultrasonografia

Podczas badania ultrasonograficznego generujemy fale akustyczne o wysokich częstotliwości skierowane w stronę obiektu, następnie rejestrujemy fale odbite. Obrazowana jest różnica gęstości poszczególnych warstw znajdujących się w obiekcie.

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

W standardzie DICOM obraz ultrasonograficzny jest oznaczany 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 związku chemicznego zawierającego izotop. Charakteryzuje się on krótkim czasem rozpadu i powinowactwem chemicznym z badanymi organami. Wykrywa się rozpad zachodzący w ciele poprzez rejestrację

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

UWAGA: Wszystkie obliczenia odbywają się we współrzędnych jednorodnych.

Na równaniu z poprzedniego punktu wykonuje takie przekształcenie:

$$PatientPosition = imgMatrix * ScenePosition$$

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

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

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

gdzie:

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

Wygląd macierzy *imgMatrix*:

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

Powyższa macierz różni się od macierzy definiowanej w standardzie. Po pierwsze wartości z 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, 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]

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

- PET-CT — połączenie PET z wilejorodzonym tomografem kumulacyjnym
 - PET-MRI — połączenie PET z rezonansem magnetycznym

W standardzie DCOM obraz jest ozaczana jako „PT”. W tej wersji badania fizyczne w sobie rożne techniki, takie jak:

- Tomographic PET

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

- ## • Tomografia SPECT

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

Przemieniowała wytworzącągo podczas tego rozpadu, a następnie przedstawiła się go w formie graficznej.

2.2 Parametry obrázku

177

Interesuje nas wyczaczenie pozycji szesćiu (punktów) na placówkach obrazu. Zajmijmy, że pacjent zna już, ile w strodzie ukradli wspinacze, i jest niezadowolony. Mimo my wiec zdefiniowac szescie punktow o nastepujacych wspinaczy, alej uzywamy pod nazwa PatientPosition, ktore beda odpowiadaly stromom pacjenta:

Wyznačanie pozycji pacjenta

- $\Delta_i \in \Delta$ — rzeczywista wielokrotka ilość słów wypisanych w słowniku, po której następuje za-

- $i \in j$ — oznacza ją współrzędne na macierzy obranu, o powiększeniu kolumnie i wiersz.

- X^{xyz} — trzy ośtańne wartości ze znacznika Tag Image Orientation (0x0020, 0x0037)
 - X^{xyz} — trzy pierwotne wartości ze znacznika Tag Image Orientation (0x0020, 0x0037)

- *S_{xyz}* – tuză wateroză și elementă de zincaciu mărge-rostchin (UO₂ZO₂), Ouzacea sau plumb pozicții Peștera waryazony și milmetracă watoskuia do 0x00032].

- D_{xy}^{uz} — koordynaty wokseli (j) we wspaniałej dydycz obrzuciwy rozumie w ilimetrach

$$\begin{bmatrix} 1 \\ 0 \\ f \\ ? \end{bmatrix} IV = \begin{bmatrix} 1 \\ 0 \\ zS \\ f \\ ? \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & f^{-1} \nabla^z X & ? \nabla^z X \\ zS & 0 & f^{-1} \nabla^f X \\ f & 0 & ? \nabla^f X \\ ? & xS & 0 \end{bmatrix} = \begin{bmatrix} 1 \\ zD \\ fD \\ ?D \end{bmatrix}$$

Wartość X w formacie Image Offset (0x0020, 0x0037) składa się z dwóch liczb, odpowiednio określających adresy pionowe i poziome. Standard DICOM definiuje następujące zakresy:

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

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

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

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

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

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

- „MONOCHROME1” i „MONOCHROME2” — ta wartość woksela odwzorowuje skale monochromatyczną, odpowiednio od jasnego do ciemnego i od ciemnego do jasnego.
- „PALETTE COLOR” — ta wartość woksela jest używana jako indeks w każdej z tabel wyszukiwania kolorów palety czerwonej, niebieskiej i zielonej. Palety mają swoje własne tagi. Wartość raczej rzadka i nie spotykana.
- „RGB” — oznacza, że woksel jest trzy-kanałowym pikselem RGB (kanały: czerwony, zielony i niebieski).
- „HSV” (ang. *Hue Saturation Value*) — woksel reprezentuje piksel w modelu przestrzeni barw zaproponowany w 1978 roku przez Alveya Raya Smitha. Model ten nawiązuje do sposobu w jakim widzi oko człowieka. Wartość wycofana.
- „ARGB” — ta wartość woksela to piksel RGB z dodatkowym kanałem przezroczystości. Wartość wycofana.
- „CMYK” — ten woksel to piksel w modelu czterech podstawowych kolorów farb drukarskich stosowanych powszechnie w druku wielobarwnym w poligrafii: cyjan, magenta, żółty, czarny. Wartość wycofana.
- „YBR_FULL” — ten woksel to piksel w modelu przestrzeni barw nazwanej $Y\text{C}_b\text{C}_r$. Dodatkowo standard zdefiniował pochodne tej wartości: „YBR_RCT”, „YBR_FULL_422”, „YBR_PARTIAL_422”, „YBR_PARTIAL_420”, „YBR_ICT”, ale wszystkie są już wycofane.

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

Połączenie macierzy

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

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

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

panTransform * *rotateTransform* * *scaleTransform* * *centerTransform*

4.6.4 Ustalanie pozycji pacjenta względem sceny

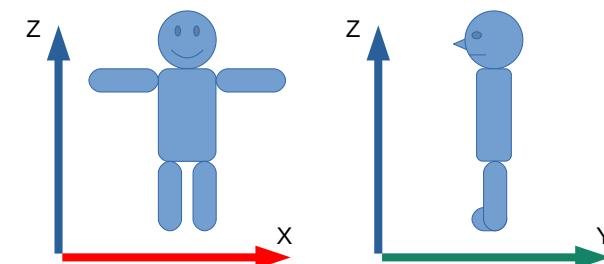
W obrazie DICOM jest pośrednio zapisana informacja o ułożeniu obrazu względem pacjenta. Celem algorytmu jest określenie jaką pozycję przyjmuje pacjent w stosunku do sceny tak, aby można było wyświetlić tą pozycję na scenie.

Format zapisu informacji o orientacji obrazu

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

Standard DICOM zdefiniował ułożenie osi we współrzędnych kartezjańskich następująco:

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



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

Wszystko powinno wyrażać zintegrowaną i zrozumiałą strategię organizacyjną, której zadaniem jest tworzenie wartościowych doświadczeń dla klientów, pracowników i innych interesariuszy. Wszystko powinno być skoncentrowane na realizacji celów i wartości organizacji, a nie na indywidualnych celach jednostek organizacyjnych. Wszystko powinno być skoncentrowane na realizacji celów i wartości organizacji, a nie na indywidualnych celach jednostek organizacyjnych. Wszystko powinno być skoncentrowane na realizacji celów i wartości organizacji, a nie na indywidualnych celach jednostek organizacyjnych. Wszystko powinno być skoncentrowane na realizacji celów i wartości organizacji, a nie na indywidualnych celach jednostek organizacyjnych.

Czasowa

kazdej techniky.

Przeźwiona

2.2.3 Rozdzieleczość

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

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

Jeśli zatem dźwięk kontrastu jest kontrast Michelsona wyraźny wzmocni:

2.2.2 Kontrast

Utrudz DICOM rownter zazwiera w sobie informacje o probkowaniu. Z uwagi na to, że probkowanie wykłada się w kategoriach, standard posiada odpowiedni zestaw znaczników dla każdej techniki. Próbkowanie posiadałyby technik opisane w sekcji 2.1.

- Tag Bits Stored (0x0028, 0x0101) — informuje jak wiele bitów z założkowalnymi posiadają wartości piksela
 - Tag Bits Stored (0x0028, 0x0101) — informuje jak wiele bitów z założkowalnymi posiadają wartości piksela
 - Tag High Bit (0x0028, 0x0102) — informuje gdzie znajdują się najstarszy bit Tag Low Pixel Representation (0x0028, 0x0103) — informuje czym pozycją są te znakiem
 - Tag Low Pixel Representation (0x0028, 0x0103) — informuje czym pozycją są te znakiem

• **Allocated Bits** Allocated (0x0028, 0x0100) — informuje na jak wielu bitów zostało zapisywanych danych, czyli liczba pozycji w obrębie, jesaż zapisana na czterech znacznikach.

- **Windrowing** — stan okienkowatia, odsłaniający przesz Skar: *Monochromie: Scene* Do obiektu okienka siedzi wysiąlane zmielą poprzek fukigie: Skar: *Windows: My Verti- cal(l) z parametrem Δy i Skar: *Windows: myHorizontal(l)* z parametrem Δx . Następ- nie ponownie jest generowany obraz z użyciem zmiennych *zmielą okienka*.*

- Sprowaia to, ze rucz pionowy jest bardziej czysty na zmianie niż ruch pozitomy.

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

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

otate = 0

- **rotate** — stiah rotacií, obnúľa výkaz výdavkov a príjmov zo zadanej forme.
- **rotate(** — stiah rotacií, obnúľa výkaz výdavkov a príjmov zo zadanej forme, ale s výsledkom vložíme do nového podávania.

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

Sprawia to, że ruch pojedynczy jest bardziej efektywny niż ruch pojedynczy. Tero-tyczne jest mówiącze implementacji odrębnego skelowania w dwoch osiach, jednakże jest to nieistotny.

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

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

scale = 1

Na macierzy skalowaniu wywoływaną jest funkcja skalowania $Q_t: Q[Transform]: scale()$ z parametrem $scale$ wykorzystywanym wozrem:

- **Zoom** — stan skalowaniia, obsrugiwanymy przez Sokar: *DiocomScene*

Na maecezty przeszukiwanie wywoływanie jest funkcja `Qt::QTransform::translate()` z parametrami odwołanymi przesunięciu myszki.

- **Pan** — stam Przeszwańska, ośmiodziwarty Przez Sokoła: Dicomscene

Zmiany pojęć obiegowych w myśl

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 to zjawiska fałszujące obraz poprzez tworzenie struktur w obrazie, nie istniejących 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

Standard DICOM przewidział możliwość zapisania wielu typów danych w różnych formatach, nie tylko obrazów, ale też nagrani audio i tekstów. Przeglądarka obrazów DICOM może mieć możliwość od czytania, wyświetlenia lub odsłuchania danych.

Podstawowe operacje na obrazie

- Skalowanie lub powiększenie, czyli możliwość powiększenia lub zmniejszenia wyświetlanego obrazu o pewny 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.

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*.

- Obrusiąga DICOMDIR, jest to rozszerzenie wczesnego pliku DICOMDIR, który umożliwia strukturę serii badan. Plik DICOMDIR to wypełnione zaznaczeniami pliki DICOMDIR, w których zawierająca.

Obstuga wileu plikow

- Maska (ang. *overlays*). Jest to mozaikowe nafaszenie maski, który będzie pokrywał obiekty, np. tła. Standard DICOM mozaikowa nafaszenie wielu mask na jednym obrazie.

- Okienkowalne. Termin oznacza sytuację, w której zmiany obrazu danych nie dotyczą monochromatycznej masywy do wyciągnięcia. Okienkowalne jest szczególowo opisane w sekci 4.6.2 wraz z generowaniem obrazu monochromatycznego.

Hotaję i odlicia listzane, czym mówiącze obroćnia obrząz o zadejny kęt oraz mówiącze użyskana odlicia listzanego obrazu w dwoch osiach X i Y.

Wyswietlacz 2.1: nazwazdzie Lupa w programie MedImage DICOM Viewer. Zdjęcie uzyskane za zgoda Softmeta UAB.



- Przykład użycia takiego narracyjnego zazdruje się na rysunku 2.1.
 - Lupi, skakowanie i śpiewanie. Jest to muzyczne i śpiewane powiekszenie obrazu.
 - Przesuwając (ang. *pan*), zły muzyczny obraz do takiego stopnia, że będące mimeski się na ekranie lub w okienku programu.

Przykład działania:
::scale().

Współzędne jednorodne dekompozycje sie jaka sposob reprezentacji punktów -wymiarowe prezenterem rozwoju za pomocą ukladu $n + 1$ wspolrzednych. W bibliotece Qt jedna z implementacji wspolrzednych jednorodnych jest klasa `Qt::QTransform`. Implementacje obu dostawcowego zachowania mierzy 3 na 3, jak rowniez widowate operacje takie jak przesuwanie implementowanej przez `Qt::QTransform::translate()`, obrot implementowany przez funkcje `Qt::QTransform::rotate()` i skalowanie implementowane przez `Qt::QTransform::scale()`.

4.6.3. Lwórzemie transformat i ich użycie na obrazie

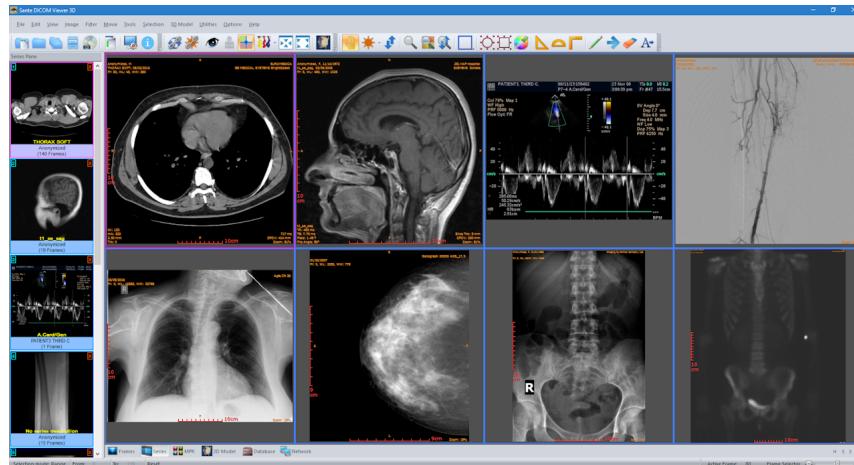
Współrzędne jednorodne

- Lupa, skalowanié mlejšcowe. Jest to moložiwoče mlejšcowego Powielkzeszta obrazu.
- Przykład uzycia takiego narzędzia zasadnicze sige na rysunku 2.1.

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

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

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



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

Generowanie obrazów wolumetycznych

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

Przykład takiego obrazu znajduje się na rysunku 2.3.

Analiza i przetwarzanie danych

- Histogram, czyli możliwość wygenerowania histogramu obrazu.

Histogram to wykres przedstawiający dystrybucję wartości numerycznych obrazu.

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

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

Przykład generowania rekonstrukcji wielopłaszczyznowej jest pokazany na rysunku 2.4

```

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 }
```

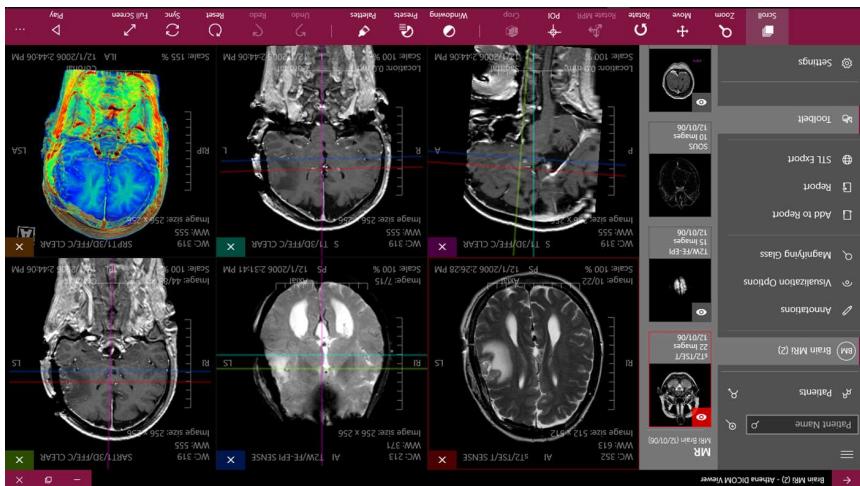
- *Sokar::Monochrome::Scene::generatePixmap()*

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

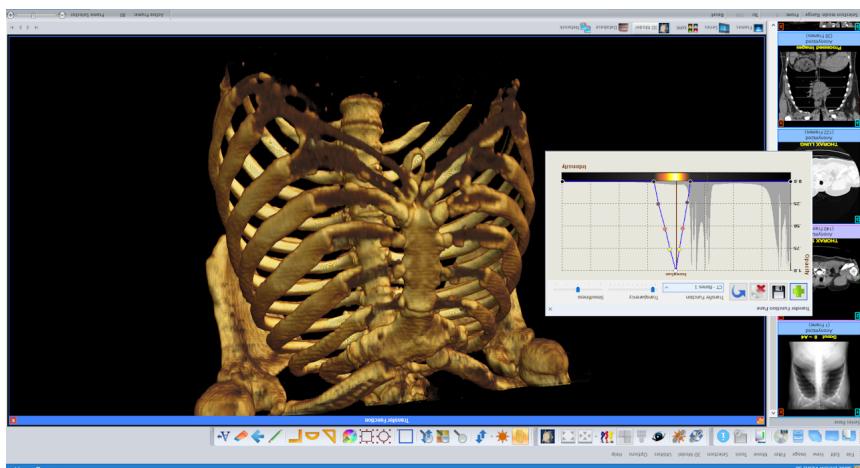
```

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 wlejopłaszczowni w przegłębacie Athena DICOM Viewer. Zdjęcie uzyskane za zgodą Medical Harbor.



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



ještě to funguje, když ráz obrazu na watek, tworzy je i mručení. Losce watekow jest ustalana za pomocí funkci `Q::detail::ThreadCount()`. Wszystki działy na zakresach o długości losci wokół podzialeń prize ilosc watekow. Kod funkcií:

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

```

1 jest funkcia definuje walletu, ktoriy hrejse po obratne. Jegej parameetrami sa zakresly
2 podanee v indeksach wokselej po ktrych ma iterovacie. IntType to jest typ zmiennej
3 woksela obrazu. MinClass klasa okienka. Nazewnictevo budezne kontynuowane w
4 nasledujucich punktach. Kod funkciei:
5 auto buffer = atagetbuffer(from);
6 auto widthptr = (IntType*) byteGetbuffer[0];
7 auto width = (MinClass*) getDifferentWindow();
8 origIn += from;
9 for (quint64 i = from; i < to; i++, origIn++) {
10     /* W tia ma tiec sa jset dokonyvania zaznamu liczby na kolar */
11     *buffer++ = windowptr->getPixel(origIn);
12 }

```

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

Po wygenerowaniu obrazu, należy przetworzyć go przed funkcją „okna”. Do zapisania w pamięci pojęcia okna, należy przetworzyć za pomocą algorytmu, który zapisze w pamięci informacje o obrazie. W C++ typu zmiennej char muszą być zdefiniowane przed komplikacją, co jest bardzo problematyczne. Mażąc dwa typy okienek, kazde odróżniające się typy liczb całkowitych, musieliśmy zainplementować dwa typy okienek, kazde odróżniające się typem zmiennej za pomocą algorytmu, który zapisze informacje o obrazie. Dla tego podzielić ten zestaw zaimplementowany za pomocą funkcji szablonowej:

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

```
*pixelArray = palette->getPixel(a * x + b);
```

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 internetowej), wsparcie przez pocztę internetową, możliwość porozumienia się z twórcą lub opiekunem oprogramowania. Forum, możliwość pytania się społeczności o opinie i ich wymiana. Wiki, strona internetowa w formacie Wikipedii dostępna dla użytkownika.

Obrazowanie dwuwymiarowe

Przewijanie(ang. *scroll*), proces wyświetlania 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. Metadane, przeglądania powinna obejmować analizowanie i wyświetlanie metadanych obiektów DICOM, powinna obejmować wyświetlanie rozdzielczości obrazu, badanie (np. identyfikator podmiotu) oraz znaczniki DICOM specyficzne dla dostawcy (np. specjalne ustawienie urządzenia rejestrującego).

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 przekroczony, 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 jest alokowany 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*. *maxValue* jest to maksymalna wartość jaką dane mogą przyjąć, jest ona równa 2^N , gdzie N to liczba bitów brana z *Dicom 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 być ujemne. Długość wektora pikseli jest sumą *maxValue* i *signedMove*. A indeks wokselu w wektorze ma wartość tego woksela zwiększoną o *signedMove*.

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

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

Standard DICOM jest odpoowiedzią specyfikacji radiologicznej, radiomarcewowej, fizykalno-chemicznej do przewozu obrazów, stacj do przewozu obrazów i modyfikowania obrazów.

2.4.1 Standard DICOM v3.0

2.4 Format cytrowycz obrazów medycznych

Ubrazowane triowy miarowe

Waższa informacja, na jawniejsze informacje powinny być wizualizowane w oknie wyswietlaczka, jako zakładka na obraz. Na przykład aktualna pozycja lub nazwa podmiotu wykonywanej jest opisane w sekcji 4.6.2. Okienkowalne to sposob zamiaty darytach na skale szarości, szare watości obranu a pseudokolor. Histogram wizualizuje wyzakrewnięce warstci kolorów na obrachach, pozwalają opisywać istotne cechy obranu. Wykazowane, możliwosci rysowania bieżą załączaną linią lub innymi kształtow do analizy i wyczacza- DCOM zawierała parametry przetwarzania (np., ilosc pikseli na centymetr). Adho- tacji (opisy), które były wtywzone przez personal medyczny powinny być zapisywane w oknie wyswietlaczka, jako zakładka na obraz. Na przykład aktualna pozycja powinny być zapisywane w oknie wykonywanej jest opisane w sekcji 4.6.2. Okienkowalne to sposob zamiaty darytach na skale szarości, szare watości obranu a pseudokolor. Histogram wizualizuje wyzakrewnięce warstci kolorów na obrachach, pozwalają opisywać istotne cechy obranu. Wykazowane, możliwosci rysowania bieżą załączaną linią lub innymi kształtow do analizy i wyczacza- DCOM zawierała parametry przetwarzania (np., ilosc pikseli na centymetr). Adho- tacji (opisy), które były wtywzone przez personal medyczny powinny być zapisywane w oknie wykonywanej jest opisane w sekcji 4.6.2. Okienkowalne to sposob zamiaty darytach na skale szarości, szare watości obranu a pseudokolor. Histogram wizualizuje wyzakrewnięce warstci kolorów na obrachach, pozwalają opisywać istotne cechy obranu. Wykazowane, możliwosci rysowania bieżą załączaną linią lub innymi kształtow do analizy i wyczacza-

Java MSI of the Sie Tzuangwei, I ordered measures to combat and we'll promote him to Shahr Monochrome Vlmidow; getPixel().

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

$$(0x - 1x) / (0h - 1h) =$$

parametry prostej přezehdzačecího přez dva puntky: r-ostandardy, tříz dva puntky a r-ostandardy dvojsíže sítě do vnitřnosti obrazu; využití

$x_1 = rescaleSlope$

$x^0 = \text{rescaleSlope}$

THEORY OF THE EARTH

$$AS = m/(q - s) \cdot \text{units}$$

Wartosci określają dobrozad się do warstwowych określających, a powietrza sklarowanej całego obszaru jest czasochłonne, przeskalowane określających, a także sam efekt:

- *width* — szerokość okienka
 - *center* — srodek okienka
 - *SV* — stored values — wartości works
 - *OutPutUnits* — wartości wyjściowe
 - *b* — wartości z Tag ReScaleLittercept
 - *m* — wartości z Tag RescaleSlope (0 dzie: glidzic)
 - *OutputUnit*
 - *wzorz.*
 - *Prizegaldaraka pozwalia na inwersję określona. I zmianie ją! zamieniać się wartościom.*
 - *Standard DICOM przewiduje, że wszystkie wartości pozwalają na inwersję określona. I zmianie ją! zmieniać się wartościom.*

gdzie:

1.0

0.0 = 0

$$x_1 = center + width/2$$

$$x_0 = center - width/2$$

Najpierw wyczekamy okienko, które zmienia warstwy obranu na skale od zera, do jedena:

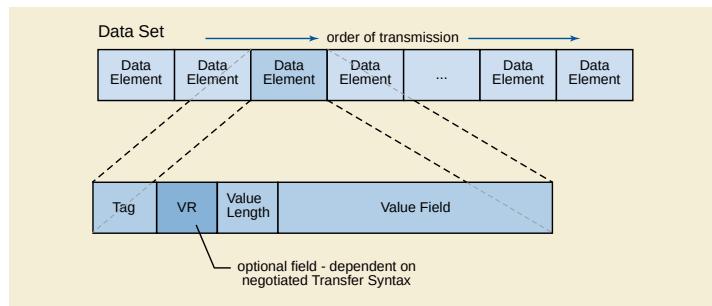
Wyznaczenie parametrow okna

jest wciąż rozwijany i uzupełniany o nowe elementy. W obecnej chwili standard DICOM definiuje 81 różnych typów badań.

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 ulożone 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 przyjają wartości odpowiednio wartość 0010_{16} i 0010_{16} to oznacza, że jest to znacznik ^{Dicom}_{Tag} PatientName (0x0010, 0x0010), czyli zwiera w sobie parametr zawierający nazwę pacjenta.

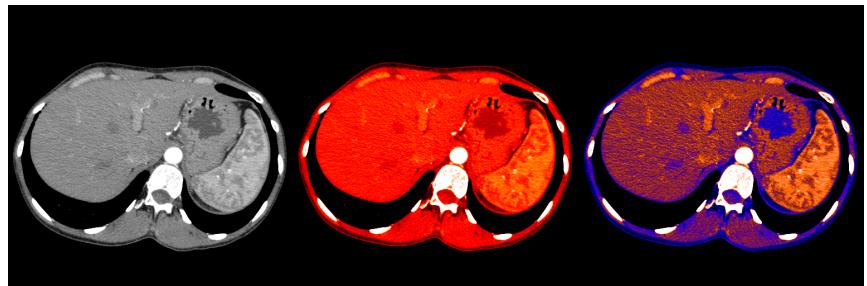
Dokładne omówienie znaczników znajduje się w sekcji 2.4.2.

- **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.



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.

Implementacja algorytmu

Opis

Z uwagi na konieczność osiągnięcia dużej szybkości wyświetlania obrazu, warto jest szacować wartości funkcji f . Wartości tej funkcji należy przeliczyć, gdy zmienione zostaną parametry tak zwanego „okna”. Indeks koloru wyznaczany jest wtedy poprzez pobieranie wartości z tabeli o indeksie równym wartości numerycznej w obrazie. Unikamy w ten 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 zapisywały by takie obrazy, gdzie dane to liczby zmiennoprzecinkowe. Dlatego w pracy założono, że takie obrazy nie będą obsługiwane.

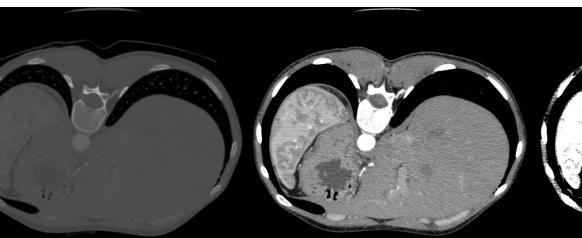
- | | | |
|------------|----------|--|
| Trzydziela | Diagnose | Patient Sex (0x0010, 0x0040) — dana wraz z nazwiskiem pacjenta |
| Trzydziela | Diagnose | Patient BirthDate (0x0010, 0x0030) — dana wraz z nazwiskiem pacjenta |
| Trzydziela | Diagnose | Patient Age (0x0010, 0x1010) — wiek pacjenta wraz z nazwiskiem pacjenta |
| Trzydziela | Diagnose | Series Description (0x0008, 0x1030) — opis serii, pole wypełniane przez technika |
| Trzydziela | Diagnose | Series Instance UID (0x0020, 0x000E) — unikalny numer serii, który jest |
| Trzydziela | Diagnose | Instancie Number (0x0020, 0x0013) — numer instancji ramki, używany w |
| Trzydziela | Diagnose | Modality (0x0008, 0x0060) — modalność określająca rodzaj techniki diagnostycznej |
| Trzydziela | Diagnose | Date (0x0008, 0x0020) — data wykonyania badania |

Laczniak

- Value Field (optionale) — Pole z parametrem o długosci VL.

- **Tag Patient ID (0x0010, 0x0020)** — Id pacjenta, unikalny identyfikator pacjenta, nazywany jest to numer HIS(Hospital Information System)
- **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 technika
- **Tag Series Instance UID (0x0020, 0x000E)** — unikalny numer serii, ktry jest nadawany kazdemu badaniu
- **Tag Series Instance Number (0x0020, 0x0013)** — numer instancji ramek, uzywany w przypadku kiedy z jednego badania zostalo utworzone kilka plikow DICOM
- **Tag Modality (0x0008, 0x0060)** — modalnosc otrzeslajaaca rozdzaj techniki diagnostycznej
- **Tag Study Date (0x0008, 0x0020)** — data wykonywania badania

Rysunek 4.10: Przykładowe obrazy w trzech roznych „funkcji okna”.



Wyzmaczalny Parameter a i b , prostes Przechodzięcej Przeciwnej y_0 jest równie 0, a y_1 jest równie 255. Funkcja „okna” wygłaśnia następująco:

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

Wyzmaczalny Parameter a i b , prostes Przechodzięcej Przeciwnej y_0 jest równie 0 , a y_1 jest równie 255 . Funkcja „okna” wygłaśnia następująco:

$$x_1 = center + width/2$$

$$x_0 = center - width/2$$

okienka center i dñgoscí width.

Reprezentacja wartości

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

Przykładowe VR:

- 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 `FrameIncrementPointer` (0x0028, 0x0009) jest używany kiedy w pliku jest zapisana sekwencja kilku obrazów. Wskazuje on na inny znacznik zawierający informacje, w jaki sposób ta sekwencja ma być wyświetlona.

- DA — Date — data lub dzień

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

Przykład: „19800716” oznacza 16 lipca 1980

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

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

Długość jednej liczby powinna maksymalnie wynosić 16 bajtów. Dostępne znaki to „0”-„9”, „+”, „-”, „E”, „e”, „.”. Biblioteka QT posiada wbudowany konwerter liczb zapisanych w formacie wykładniczym, dlatego mój konwerter dzieli tekst i konwertuje za pomocą QT.

Przykład: „426\468” oznacza dwie liczby 426 i 468. Proszę zwrócić uwagę na spację 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, dlatego mój konwerter używa konwertera z QT.

Przykład: „426” oznacza liczbę 426.

YBR

YBR albo YC_bC_r to model przestrzeni kolorów do przechowywania obrazów i wideo. Wykorzystuje do tego trzy typy danych: Y – składową luminancji, B lub C_b – składową różnicową chrominancji Y-B, stanowiącą różnicę między luminancją a niebieskim, oraz R lub C_r – składową chrominancji Y-R, stanowiącą różnicę między luminancją a czerwonym. Kolor zielony jest uzyskiwany na podstawie tych trzech wartości. YBR nie pokrywa w całości RGB, tak jak RGB nie pokrywa YBR. 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 jeszcze wyświetlić na monitorze RGB. Dlatego należy przekonwertować kolor YBR na kolor RGB, iterując po wszystkich wartościach obrazu.

Poniżej przedstawiono kod źródłowy funkcji zamiany kolorów YBR na RGB.

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

4.6.2 Generowanie obrazu monochromatycznego

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

Pseudokolorowanie obrazu

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

Zwiększenie kontrastu za pomocą „funkcji okna”

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

- W momogeah komputowes wylakiem rekonstrukcji jest maccierz lecz ob oznaczeniach aktywacyjch tip. Wykazanych przestronny wspolczynnika oscylacyjnego Promieniowania. Ze wzgledu na aspekty rozkadu przestronny wspolczynnika oscylacyjnego Promieniowania. Ze wzgledu na aspekty przezwanej medycznej, niezwylknie istotna rzeczyzna jest zapisy ozyginalnych danych numerycznych. Ze tego powodu produceni sprzedzili wizualny formaty plikow cyfrowych. W plikach tych oprocz numerycznych danych obrazowowych zapisane sa parametry warunkow aktywacyjch tip.

2.4.4 Linne formaty zapisu

- W przypadku wikszych mystycznego poszawia sie problem indeksowania plikow i ich przeszukiwania. Wykazanie konkretnego badania lub pliku w folderze, w ktorym znajduje sie kilka plikow poprzec rozwiazaniem lezadego pliku do pameci i analiza jego danych nie jest najslabsze plikow poprzec rozwiazaniem optymalnym. Dla tego standard DICOM definitivie rowniez pliki typu DICOMDIR, ktorzy jest plikiem indeksujacym pliki DICOM w folderze. Pozwala to na efektywne przechodzenie wlosu serii badan bez wczytywania plikow badan.

2.4.3 DICOMDIR

- SS — Signed Short — 16 bitowa liczba całkowita bez znaku
 - US — Unsigned Short — 16 bitowa liczba całkowita bez znaku
 - UT — Unlimited Text — tekst o nieograniczonej długości.
 - Zwykły tekst o długości maksymalnie 2³² – 2 bajtów.

Przykład: „prof. dr. hab. inż. Władek Smolik pracoował w „ZEFIR”” był by zapisany w komunikatze o mianowaniu, miany zatrudnionej, ze względu na:

- żartowy żartowe kompleks — nazwisko, np. Smurk
- givenej żart name kompleks — nazwisko, np. Adam
- middle name — strodkowe imię, brak odpowiednika w polskim nazewnictwie
- name prefix — przedimek przedimiennym, np: mgr. inż.
- name suffix — sufixy po imieniu, brak odpowiednika

„Name“ díleňa poszczególnych skladowych nazwy na ozaczonie fragmenty. „Person“ rozdzielena poszczególnymi skladowymi nazewnicicą, standard DICOM nie przewiduje biegały od poszkeglo standardu nazewnicicą, standard DICOM nie przewiduje podawany w sposob dowolny i od-

- PN — Person Name — nazwa osoby

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

- R_n — war tość czerwonego kanału
 - G_n — war tość żółtego kanału
 - B_n — war tość niebieskiego kanału

gdzie:

- 0 — oznacza to, że wątłosci pikseli są ułożone w taki sposób, aby dla każdego piksela (x, y) wartości $W_{x,y}$ spełniały warunek $W_{x,y} = W_{x,y+1} = \dots = W_{x,y+H-1}$. Wartości pikseli w wierszu y są ułożone w taki sposób, aby dla każdego piksela (x, y) wartości $W_{x,y}$ spełniały warunek $W_{x,y} = W_{x+1,y} = \dots = W_{x+W-1,y}$.

Oríaz monodiktoriatyczny to obraz w dotleniach szarości, o którym do czarnego libo do czarnego do białego. Generowało takiego obrazu abywa sie poprzecz pseudokolorowanię. Czyli proces jest wyjściowy w skali 4,6-2.

- **IconPixmap** obiekt odrzuciu ikonu, klasę *Qt::Pixmap*, 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++. Komplikacja oprogramowania używającego Qt może odbywać się za pomocą dwóch narzędzi: CMake oraz dedykowanego narzędzia „qmake”, zrobionego specjalnie na potrzeby biblioteki Qt. Dzięki czemu cały projekt przeglądarki używa tego samego języka oraz tego samego narzędzia zarządzania komplikacją.

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

4.6 Algorytmy

4.6.1 Cykl generowania obrazów

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

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

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

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

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

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

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

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

3.2.1 Wyoming

Według autorki, Oł powinno się czystać jak antiglęskie słowo „cute”, po polsku „kute”. Dedykująca spotecznośc programistów nie jest co do tego zgodna. Ankiety zrobione na dwóch popularnych serwisach internetowych o tematyce programistycej, pokazują, że ma ją bardziej popularna wśród młodych niż starszych. „Oł”, po polsku „kute”.

3.2.2 Licencja

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

Biblioteka Of jeft dystybuwana w dwoch wersjach: komercyjne i otwarte zrodlowe. Wersja komercyjna jest dostepna w dwoch wersjach: wersja 3, co pozwalala na uzycie biblioteki w celu prywatnego wykorzystania, ale jest dostepna tylko dla licencji GNU General Public License w wersji 3, co pozwala na uzycie biblioteki w celu prywatnego wykorzystania.

3.2.3 Normy i certyfikaty

The Qf Company Posiada szereg certyfikatów od FDA i UE, które ułatwiają wprowadzenie produktów leczniczych bibliotek Qf na europejski i amerykański rynek medyczny. Posiadamy normy: Lista

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

Wypełnij formularz i kliknij przycisk "Zapisz".
Wyszczególnij na temat certyfikatu mówiąc, że jest to dokument, który potwierdza, że podany adres jest właściwym dla Twojej strony internetowej.

3.2.4 Globálne typy struktúr

W rozwiązych systemach operacyjnych seł rozszerza kompatybilność i wszerd też rozszerzało-
ski pozwalała się problem dotyczący zmiany funkcji fundamentalnych. Przykład jest zagnie-
mieńie: ile biorąc ma zmienna [int](#)? Działając na niej do dokumentacji C++, dosięgnie pod-
adresem https://pl.cppreference.com/w/cpp/language/types, możemy dowiedzieć-
się, że [int](#) ma minimum 16 bitów. Natomiast w dokumentacji MSVC, kompilatora firmy
Microsoft, zaznajęcie się pod adresem <https://docs.microsoft.com/pl-pl/cpp/>
mówiące o długości liczb całkowitych uzytej dla danych [int64](#), [int32](#), [int16](#),
i [int8](#)-[int64-int32-int16-int8](#). Widzimy, że aby
mówiąc o długości całkowitych należy uzyć takich typów: [int8](#), [int16](#),
[int32](#), [int64](#). Wszystko to zauważmy, że do systemu komputerowego przekazane przez bibliotekę Qt
jest to problem, który biblioteka Qt rozwiązała wprowadzając dodatkowe typy literałowe,
które dostosowują się do systemu i kompilatora oraz zapewniają pewne podczasy deklaracji,
że dana zmienna będzie zakodowana dla tego systemu. Dodać zauważ, że dla systemów pod-
mieszkowych, takich jak Arduino, nie ma możliwości dostosowania do systemu, zatem
dostępne są typy [qInt8](#), [qInt16](#), [qInt32](#), [qInt64](#), [qUInt8](#), [qUInt16](#), [qUInt32](#), [qUInt64](#), [qLong](#) i [qLongLong](#).
Dla tego, aby móc skorzystać z tych typów, musimy dodać do projektu bibliotekę Qt, a po-
tem dodać ją do naszej aplikacji. W tym celu, po otwarciu projektu w QCreatorze, klikamy
prawym przyciskiem myszy na nazwę projektu, a następnie wybieramy [Add to Project](#) → [Qt Global Headers](#).
Po dodaniu biblioteki, możemy skorzystać z nowych typów, np. [qInt8](#) zamiast [int8](#).
Wszystko to pozwala na skorzystanie z nowych możliwości, np. [qInt8](#) zamiast [int8](#).
Dla tego, aby móc skorzystać z tych typów, musimy dodać do projektu bibliotekę Qt, a po-
tem dodać ją do naszej aplikacji. W tym celu, po otwarciu projektu w QCreatorze, klikamy
prawym przyciskiem myszy na nazwę projektu, a następnie wybieramy [Add to Project](#) → [Qt Global Headers](#).
Po dodaniu biblioteki, możemy skorzystać z nowych typów, np. [qInt8](#) zamiast [int8](#).

Programme okno programu jest implementowaną przez *Sokar: MainWindow*. Jest wywoły-
wane od razu po uruchomieniu programu.

Zawiera w sobie 4 elementy: menu; dżewo ze strukturą plików; okienko z zakladkami
oraz w dolnej części okna sugerującą, aby nie uzywać programu w celach medycznych.

4.5.6 Unknown programme

po moco&funckej Sjofar:::DicomFileSet:::create(), opisanej w sekci& 4.5.3.

Otworzene nowego pliku moze odbyc sie z nastepujacych zasad: obiektu dzisiewa ze struktura plikow w systemie (opisane go w 4.5.4), menu programu (opisane go w 4.5.6), lub poprzednia przeklagnoscie i uproszczenie. Z dwucho pierwszych moze wazycelle tylko po jednym pliku, natomiast trzecim sposobem moze wazycelle zapisywane juzen jak i wiele plikow. Wysylanie prosby o dbywanie sie za pomoca dwucho funkcji: `Sokar::DiskomTabs::addDiskomFile()` i `Sokar::DiskomTabs::addDiskomFiles()`. Kaza z tych funkcji ma dwa parametry: `parametru` sicezki a drugie z wazniejszym plikiem.

- `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 czasu 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 jest zmieniana scena.

4.5.5 Obiekt zakładek

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

Fizyka dowa klasa dziedziczka po Objekt

Sygnalny i sloty

```

    // Tworzymy obiekt przycisku
    auto *pButt = new QPushButton("Quit");
    // Tworzymy obiekt okna
    auto *win = new QWidget();
    // Tworzymy obiekt okna
    auto *win2 = new QWidget();
    // Przypisujemy dodatkowa przyciskowi
    pButt->setParent(win2);
    // W tym momencie przycisk wraz z oknem zostaja usuniête
    delete win2;
}

```

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

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

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

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

Po otrzymaniu sygnału od ekranu klawy *Shift+Space* powiniene pokazać libu ukryć sie w zalezności od stanu pozycji.

— Patient Data — Dame pacienta

AKcja: **Clearrotate**.
Po otrzymaniu sygnału obraz na scenie powinien wykonać transformację obrótu.

Alkja: **EliphorizontaL**.
Po otrzymaniu sygnału obraz na scenie powinien oddać się lustrowane pozycje.

— Rotate Left — Obrot w lewo
Akja: **RotateLeft90**.

```

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

```

3.2.6 Graficzny interfejs użytkownika

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

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

Kilka przykładowych klas obiektów graficznych i ich cechy

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

Pasek narzędzi

Pasek narzędzi znajdujący się na górze, implementowany 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 na 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, góra, dół, kiedy jest wciśnięty klawisz myszy.

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

- Skalowanie (3)

Stan: *Zoom*. Oznacza, że ruch myszki powinien skalować obraz kiedy jest wciśnięty 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:

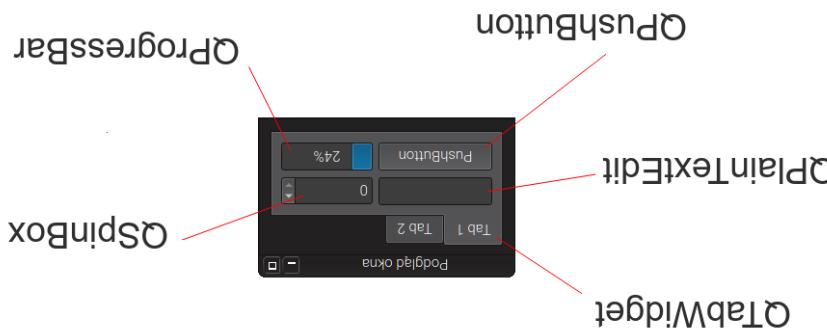
- Rotate Right — Obróć w prawo

Akcja: *RotateRight90*.

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

3.3.1 Uzasadnienie wyboru

8.3 GDCM



WYSZKOWIE ORIGINS PROJECT

- **Q: QSpinBox —** implementuje przeklętkę, cząstki kontrole przystosowaną do wprowadzania liczb przeszłością. Posiada dwa dodatkowe przyciski powiększające i zmniejszające liczbę wprowadzoną.
 - **Q: QProgressBar —** implementuje przeklętkę, wprowadzającą liczbę implementującą po klasie **QWidgetItem**. Przykład pozycji ma postać:

```
QProgressBar *progressBar = new QProgressBar(this);
```

 - **Q: QPlainTextEdit —** implementuje pole mówiące wprowadzanie tekstu zapisanego w plikach. Dziedziczy po klasie **QAbstractScrollArea**, której dziedziczy po **QFrame**, z której po klasie **Q: QWidget**. Przykład można zobaczyć na przykładowym rysunku 3.1.
 - **Q: QTableWidget —** implementuje tabelę, pole mówiące wprowadzanie tekstu zapisanego w tabelach. Dziedziczy po klasie **QAbstractScrollArea**, której dziedziczy po **QFrame**, z której po klasie **Q: QWidget**. Przykład można zobaczyć na przykładowym rysunku 3.1.

Rysunek 4.9: Wykaz zatrudnienia wraz z numeracją elenemów interfejsu. Zajęcie własne.



Dodatkowo posiada obiekt `kolekci` scen opisane w sekci 4.5.3.

- *DiocomToolBar*, opisany w sekcií 4.5.4. poskytuje sié na gorie — implementovany za pomocou triedy *SoftBar*;
 - *DiocomGraphics*, opisany w sekcií 4.5.4. mriežce na scéne z obrazom DICOM na stránku — implementovany za pomocou triedy *SoftBar*;
 - *Swak* filmu w dobej časťi — implementovany za pomocou triedy *SoftBar*;
 - *Sokar* opisany w sekcií 4.5.4. podobným západom, opisany w sekcií 4.5.4. — implementovany za pomocou triedy *SoftBar*;

11.5.4 Zaktadka

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

3.3.2 Opis

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

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

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

3.3.3 Licencja

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

3.3.4 Podstawowe klasy

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

```
gdcm::ImageReader
```

Wszystkie funkcje wewnętrz klas są oznaczone następująco:

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

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

- `gdcm::Reader` — klasa służąca do wczytywania pliku DICOM
- `gdcm::ImageReader` — klasa służąca do wczytywania obrazu DICOM, dziedziczy po `gdcm::Reader`, jest w stanie wygenerować obiekt obrazu

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

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

Kolekcja ramek DICOM

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

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

Została zaimplementowana obsługa ponizszych znaczników:

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

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

Kolekcja plików DICOM

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

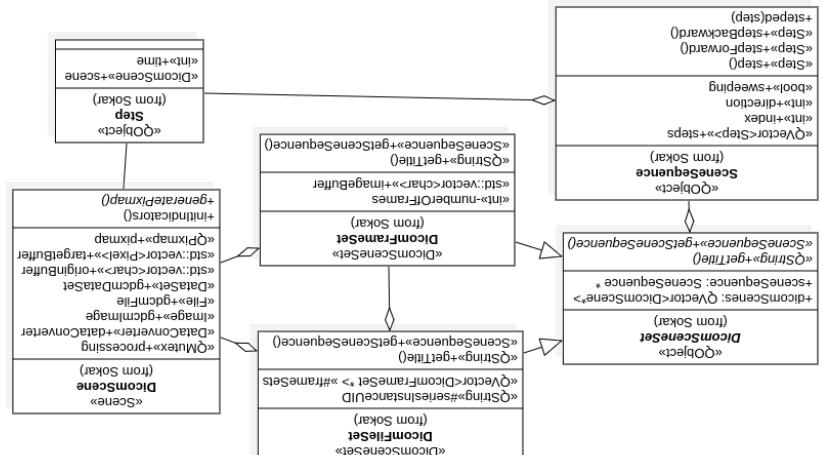
Sekwencja jest tworzona poprzez połączenie sekwencji poszczególnych obrazów.

Segregowanie obrazów

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

Sekwencja scen

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



Abstractyjna klasa **Sokar**: **LichomSceneceset** implements wektor scen za pomoca klasy **Q::Vector**. Jest to obiekt, który przechowuje sceny i tworzy sekwencje scen, które ją stają się możliwe. Dla programu klas **ML** zasadnicze się na ryzyku 4.8. Dlakow i kolęka ją ramka z jednego pliku. Dla programu klas **ML** zasadnicze się na ryzyku 4.8.

4.5.3 Kollekje scen

- *gdcm::Image* — obiekt opisujący struktury obrazu w formacie DICOM
 - *gdcm::File* — obiekt pliku DICOM
 - *gdcm::DataSet* — obiekt zbioru elementów
 - *gdcm::Element* — obiekt elementu danych
 - *gdcm::Tag* — obiekt znacznika
 - *gdcm::StringFilter* — pomocnicza klasa służąca do konwersji na ciągi znaków

3.3.5 Przykład uzywania

- *gdcm::StringFilter* — pomocnicza klasy

- *gdcn::DataElement* — obiekt elementu danych
- *gdcn::DataSet* — obiekt zbioru elementów

- *gdcm::File* — obiekt pliku DICOM

Przykład wczytania pliku

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

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

– „KVP” — Szczytowe napięcie wyjściowe generatora promieniowania rentgenowskiego — wyrażone w kilo voltach, pobierane z $\frac{\text{Dicom}}{\text{Tag}}$ KVP (0x0018, 0x0060)

- MR — rezonans magnetyczny

– „Repetition time” — Czas repetycji — pobierany ze znacznika $\frac{\text{Dicom}}{\text{Tag}}$ Repetition Time (0x0018, 0x0080).

– „Echo time” — Czas echa — pobierany ze znacznika $\frac{\text{Dicom}}{\text{Tag}}$ Echo Time (0x0018, 0x0081).

– „Magnetic field” — Pole magnetyczne — nominalna wartość pola magnetycznego wyrażona w teslach pobierana ze znacznika $\frac{\text{Dicom}}{\text{Tag}}$ Magnetic Field Strength (0x0018, 0x0087).

– „SAR” — Swoiste tempo pochłaniania energii — pobierane ze znacznika $\frac{\text{Dicom}}{\text{Tag}}$ SAR (0x0018, 0x1316).

Generowanie obrazów z danych

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

Przekształcenia macierzowe obrazu

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

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

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

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

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

Prykład wczytania obrazu

W tym pacyfikacyjnym sporwadzie wszyscy wracają do Pionowej Rzeczy. Stosownie do tego.

```

4 /* Tworzymany obiekt obrazu */
5 gdcm:Image::Image gdcm:ImageRead( it );
6 gdcm:Image::Image gdcm:ImageRead( it );
7 it->SetFileName( "path/to/file" );
8 it->Read();
9 /* Przydaje wyspielenia bledu mozemy go obsluzyc. */
10 Klasa gdcm:ImageReader zapisz bledy w pliku nie
11 bledzie obrazu, bedzie bedzie w niesposredniam formacie */
12 return 1;
13 {
14 /* Podstawy obiekt obrazu */
15 const gdcm:Image gdcm:Image( it );
16 const gdcm:Image gdcm:Image( it );
17 /* Tworzymany obiekt obrazu */
18 const gdcm:Image gdcm:Image( it );
19 std::vector<char> zmianaJegoWielkosci;
20 imguiffere.resize( gdcm:Image::GetBuufferLength() );
21 /* Podstawy wymiaru obrazu */
22 const gdcm:Image gdcm:Image( it );
23 const gdcm:Image gdcm:Image( it );
24 const dim3 dim3 = dimension[0];
25 quatin dim3 = dimension[1];
26 std::cout << "wysokosc obrazu " << endl;
27 if( gdcm:Image::GetPhotometricInterpretation() ==
28 gdcm:PhotometricInterpretation::RGB )
29 std::cout << "wysokosc obrazu RGB" << endl;
30 if( gdcm:Image::GetPhotometricInterpretation() ==
31 gdcm:PhotometricInterpretation::MONOCHROME2 )
32 std::cout << "wysokosc obrazu monochromatyczny typu drukiege "
33 if( gdcm:Image::GetPixelFormat() == gdcm:PixelFormat::UINT8 )
34 std::cout << "wysokosc obrazu monochromatyczny typu drukiege "
35 gdcm:PixelFormat::MONOCHROME2 );
36 std::cout << "wysokosc obrazu monochromatyczny typu drukiege "
37 if( gdcm:Image::GetPixelFormat() == gdcm:PixelFormat::UINT16 )
38 std::cout << "wysokosc obrazu monochromatyczny typu drukiege "
39 const gdcm:PixelFormat::UINT16 );
40 const gdcm:PixelFormat::UINT16 );
41 const gdcm:Database &database = it->GetDatabase();
42 const gdcm:Database &database = it->GetDatabase();

```

Dodatakowe informacje o modalności

Petry opis implementacji algorytmu wyznaczania stron zasadu sie w sekcji 4.6.4.

- Get to system kontrolli wersji. Czta praca zostala wykonaana przy asyście tego narzedzidia, a reprezentowana z programem zapisuje sie w pliku `pl/ajedrzejowsk/sokar-writting`. Pod adresem <https://gl.ite.pw.edu.pl/ajedrzejowsk/sokar-app>. Znajduje sie tam plik `ajedrzejowsk/sokar-writting`. Pod adresem <https://gl.ite.pw.edu.pl/ajedrzejowsk/sokar-writting> pod adresem <https://gl.ite.pw.edu.pl/ajedrzejowsk/sokar-writting>.

3.4 Git

- CT — tomografia komputerowa

- Główne cele projektu to stworzenie aplikacji, która pozwala na zarządzanie projektami i zarządzanie zasobami ludzkimi. Aplikacja ma być skonstruowana tak, aby była łatwa w użyciu, a zarządzanie projektami było efektywne i efektywne.

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 władczy 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 przekształceniami macierzowymi

- Opis wykonany przez instytucję lub klasyfikację badania (komponentu)
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 nie przewidywalna.

- 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 nie przewidywalna.

Przykład pełnego tekstu:

Adam Jędrzejowski ♂
HIS/123456
born 1996-07-16, 19 years
Kregosłup ledzwiowy a-p + boczne
AP

Dane jednostki organizacyjnej

Są implementowane przez *Sokar::HospitalDataIndicator*. Pojawia się zawsze na scenie w prawym górnym rogu i zawiera 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

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

4.3 Graficzny interfejs użytkownika

4.2 Wie lohnen Plattformen

W dokumentie są wielokrotnie zauważane do klas z przegładałalci odrzawą. Dla tego, aby zwiększyć cyfertypie pracy, zostala zastosowana konwencja poprzecznia klas z aplikacjí przekrojówkami. Skarabie: ktryy za razem jest przestępem nazw programu. Przykład poniżej:

Skarabie: DataConverteer::DataConverteer()

Wszystkie funkcje wewnatrz klas są ozaczane nastepująco:

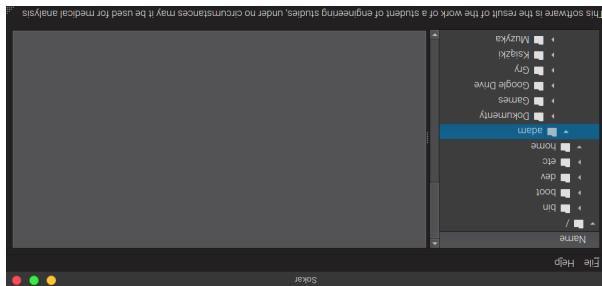
Skarabie: DataConverteer::DataConverteer()

Dodatkowo w dokumentie PDF monia klikać na nazwę klasy i uzyskowalik zostanie przekierowany do TU WYMYŚLIC DO CZECHO

```
Sokar::Data::OneLetter::toString()
```

Użytkownika może otworzyć plik DCOM na tryj sposoby: z menu na górze, z drzewa struktury lub poprzez przekląganie (ang. drag and drop). W dwoch pierwszych przypadkach użytkownik może otworzyć tylko jeden plik, a w trzecim jest możliwość otwarcia wielu plików.

Ryšunek 4.1: Okno přezgládátku tuz po uruchomieniu. Zdjēcie własne.



Po uruchomieniu programu uzyc klawiszu **Wyszukaj** (zakladka **z obrazami** (obiekt **Klasy** (*Object: Main*), menu (**obiekt Klasy** (*Object: MainMenuBar*)), dzwo phikow (obiekt **Sokar** (*Object: Sokar*)), dzwo phikow (obiekt **Skar** (*Object: Skar*)), dzwo phikow (obiekt **DiocomTabs** (*Object: DiocomTabs*)).

Przykaz: „born 1982-08-09, 28 years.”
Data urodzenia zapisana za pomocą formata `YYYY-MM-DD`. Dodałkowo, jeżeli tag `Birth Date` (0x0010, 0x0030) jest zainicjowany na formacie `YYMM`, Patient Birth Date (0x0010, 0x0030) jest interpretowany jako data urodzenia, w której pierwsza i druga cyfra to rok, a trzecia i czwarta to miesiąc, a piąta i szósta to dzień. Jeżeli tag `Birth Date` (0x0010) jest obecny, wyswietlany jest także wiek pacjenta w czasie badania.

- Dla tych, którzy nie mają możliwości korzystania z aplikacji mobilnej, skonstruowano specjalną wersję strony internetowej.

Przykład: „HIS/000000”.

Ulikáhy identifikátor Pacienta ze značkula **Barcode** Patient ID (0x0010, 0x0020) wyswietlalý jest w takiej formie, w jakiej jest zapisany. W przekształcie nazwiski jest to numer z systemu uzywanego w danyim szpitalu, rzadziej numer PESEL.

- Identity faktor Pacjenta

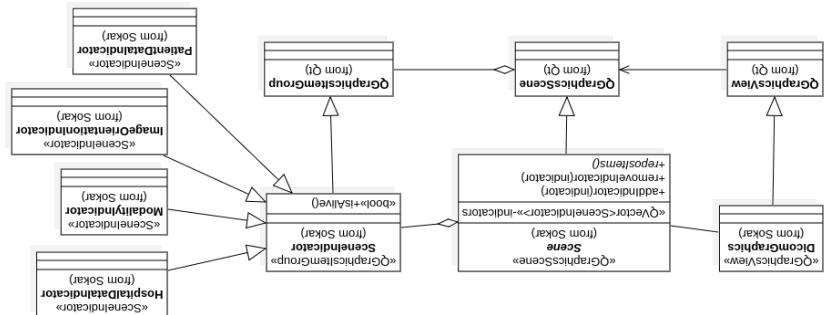
- "M" — oznacza miedzianie, wyświetlaną jako znak 
- "F" — oznacza kobiety, wyświetlaną jako znak 
- "O" — oznacza inne płeć i inne jeść wyświetlaną 
- "Pizzakid: Adam jedzie wiśnią O" — Pizzakid: Adam jedzie wiśnią 

Nazwa pacjenta zapisana jest w **Tag** Patient Name (0x0010, 0x0010) o VR.PN. Płeć, zapisana jest w **Tag** Patient Sex (0x0010, 0x0040) i może mieć nastepujące wartości:

Dane paczkieta jest mączka nowoczesne prezenty. Słuchaj: *Placenta* i *extraordinary* i *posawiają* się zawsze na scenie w lewym górnym rogu. Zawierają natrępujące linie:

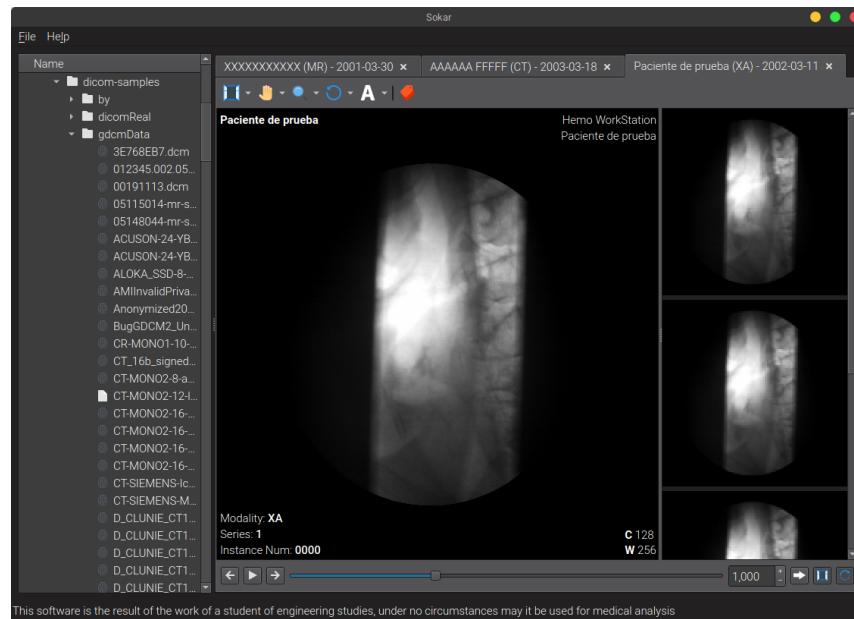
Domysliwe obiekty wyświetlające informacje (tytuły punktów to nazwy klas):

rysunek 4.7: Diagram klas UML dziedziczenia klas *Sokar::SceneIndicator*.



Wyszystkie elementy wyswietlaj±ce dane z pliku DICOM dziedzicza po klasie `Sofer::SceneIndicat or`. Dla g³±ram klas `UML` nazajduje siê na rysunku 4.7.

Po wczytaniu pliki są wyświetlane w zakładkach. Kontener z zakładkami jest implementowany przez klasę *Sokar::DicomTabs*. Przykład programu z wczytanymi kilkoma plikami, w tym jednym z animacją znajduje się na rysunku 4.2

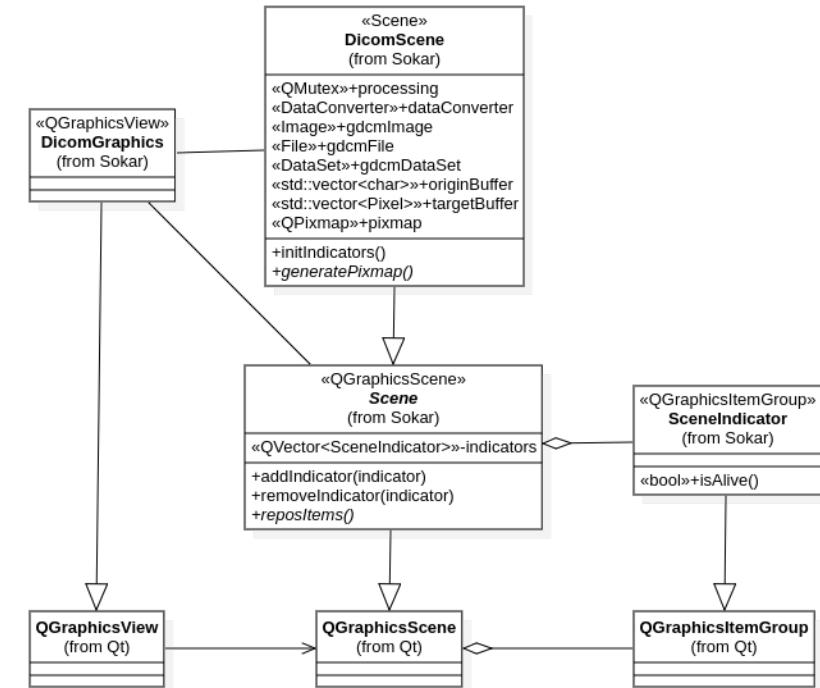


Rysunek 4.2: Okno przeglądarki z wczytanymi kilkoma obrazami. Zdjęcie własne.

Obiekt wewnętrz 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 pomocą, którego 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 wczytany tylko jeden obraz.

Scena to obiekt wyświetlający i generujący obraz na ekranie. Dodatkowo na scenie znajduje 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



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

4.4 Projekt struktury obiektowej programu

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

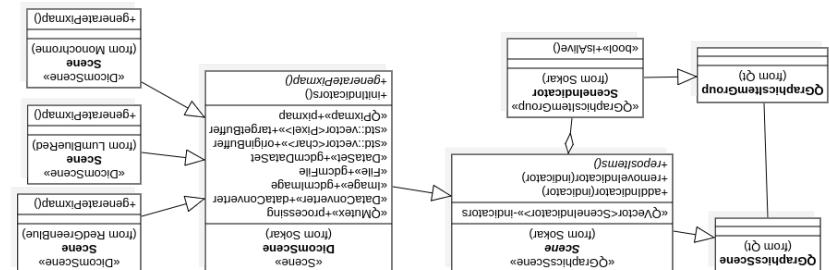


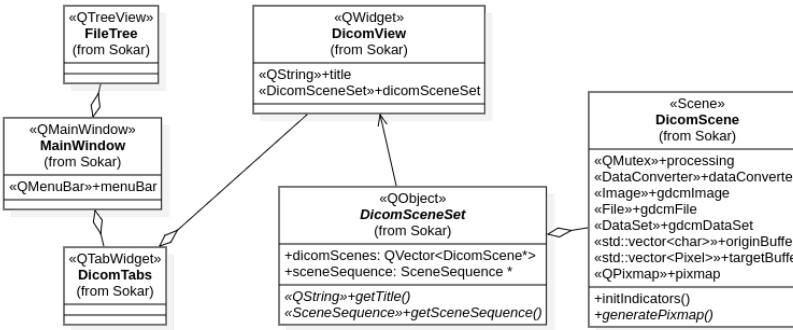
Przykładowa scena z obrazem monocromatycznym z napisem "sie na rysunku 4.3".

W pierwszej warstwie elementy graficzne zasłaniane za pomocą klas **Skar::SceneElement**, dziedziczącą po **Qt::QGraphicsItem**. Skrytka: **Skar::SceneElement** dziedziczy z kolei po **Qt::QGraphicsItemGroup**. Skrytka zasłaniaającą elementy graficzne zasłaniane za pomocą klas **Skar::Scene**, dziedziczącą po **Qt::QGraphicsScene**. Kontrolka graficzna zasłana za pomocą klas **Skar::GraphicControl**, dziedziczącą po **Qt::QGraphicsItem**. Skrytka zasłaniająca elementy graficzne zasłaniane za pomocą klas **Skar::GraphicControl**, dziedziczącą po **Qt::QGraphicsItem**. W drugiej warstwie elementy graficzne zasłaniane za pomocą klas **Skar::Scene**, dziedziczącą po **Qt::QGraphicsScene**. W trzeciej warstwie zasłana klasa **Skar::Scene**, dziedziczącą po **Qt::QGraphicsView**.

- **Qt::QGraphicScene** — element grupyjacy wiele elementow, pozwalajacy na łatwą implementację bardziej złożonych struktur
 - **Qt::QGraphicItem** — element grupyjacy wiele elementow, obiekty klasy
 - **Qt::QGraphicItemGroup** — element wywiązelajacy obiekty graficzne, obiekty klasy
 - **Qt::QGraphicLineItem** — element wywiązelajacy prostą linię z punktu A do B
 - **Qt::QGraphicTextItem** — podstawowy obiekt HTML

Rysunek 4.5: Diagram klas UML dziedziczenia klas *Sokar* i *DiomsScene*.





Rysunek 4.4: Diagram klas UML globalnej struktury programu.

4.5 Struktury danych

4.5.1 Konwertowanie danych ze znaczników

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4.5.2 Scena

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

Wyświetlanie sceny

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

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