Wieloplatformowa przeglądarka obrazów DICOM w C++

Adam Jędrzejowski

Wydział Elektroniki i Technik Informacyjnych Politechniki Warszawskiej
Instytut Radioelektroniki i Technik Multimedialnych
Zakład Elektroniki Jądrowej i Medycznej

27 czerwca 2019

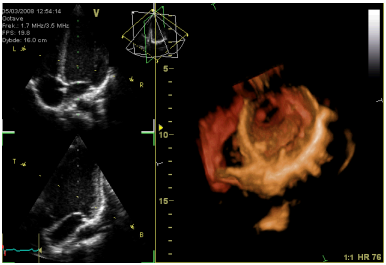
Digital **I**maging and **Co**mmunications in **M**edicine

Standard DICOM jest odpowiedzią społeczności radiologów, radiofarmaceutów, fizyków medycznych na potrzebę wymiany danych pomiędzy różnymi systemami komputerowymi, przeglądarek obrazów, stacji do przetwarzania i analizowania obrazów medycznych.

Standard DICOM v3 definiuje ujednolicony sposób zapisu i przekazywania danych medycznych reprezentujących lub związanych z obrazami diagnostycznymi w medycynie.

Obrazowe techniki medyczne:

- Radiografia, Tomografia komputerowa – współczynnik natężenia promieniowania X przez obiekt
- Obrazowanie metodą rezonansu magnetycznego – sumaryczna gęstość atomów wodoru
- Ultrasonografia – różnica gęstości poszczególnych warstw znajdujących się w obiekcie
- Scyntygrafia, Tomografia SPECT, Tomografia PET – rozkład radiofarmaceutyka w obiekcie



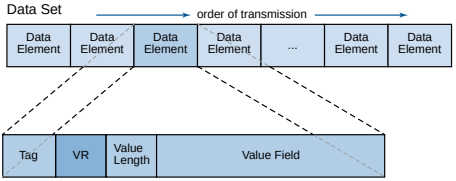
Standard DICOM

Plik w formacie DICOM można traktować jako zbiór elementów danych. Zbiór nazywa się "Data Set" i składa się z rekordów, które nazywają się "Data Element". Elementy danych są ułożone w postaci listy. Element danych może zawierać w sobie listę elementów danych.

Element danych składa się ze znacznika, interpretacji danych, długości danych i danych.

Znacznik składa się z dwóch liczb (grupy i elementu grupy) i informuje co dane zapisują. DICOM definiuje dwa typy znaczników: publiczne i prywatne.

W obecnej chwili standard DICOM definiuje 81 różnych typów badań i ponad 4000 publicznych znaczników.



Nazwa	Identyfikator	Typ danych	Opis
SpecificCharacterSet	(0008,0005)	CS	Używana specyfikacja
InstitutionName	(0008,0080)	LO	Miejsce wykonywania badania
Manufacturer	(0008,0070)	LO	Producent aplikacji
StationName	(0008,1010)	SH	Nazwa urządzenia
PatientID	(0010,0020)	LO	Identyfikator pacjenta
PatientsName	(0010,0010)	PN	Nazwisko pacjenta
PatientsBirthDate	(0010,0030)	DA	Data urodzin pacjenta
PatientsSex	(0010,0040)	CS	Płeć pacjenta
PatientsAge	(0010,1010)	AS	Wiek pacjenta
BodyPartExamined	(0018,0015)	CS	Badana część ciała
StudyDate	(0008,0020)	DA	Data badania
PhotometricInterpretation	(0028,0004)	CS	Format zapisu obrazu
Rows	(0028,0010)	US	Wysokość zdjęcia
Columns	(0028,0011)	US	Szerokość zdjęcia

Cel pracy i założenia

Cel pracy

Opracowanie przeglądarki obrazów DICOM, działającej w wielu systemach operacyjnych

Założenia:

- Wieloplatformowość:
  - eliminacja start wydajności spowodowanych wykonaniem kodu wirtualnego (wirtualizacja kodu binarnego z pomocą maszyny wirtualnej JVM; języki skryptowe, których interpretacja kodu jest równoległa z wykonywaniem)
  - kod źródłowy z możliwością kompilacji na wskazane platformy: Linux, MacOS, Windows.
- Obsługa obrazów statycznych i dynamicznych, serii przekrojów tomograficznych
- Realizacja podstawowych funkcji typowej przeglądarki obrazów medycznych: zmiana kontrastu z pomocą “okienkowanie”, skalowanie, rotacja.

Biblioteki i narzędzia

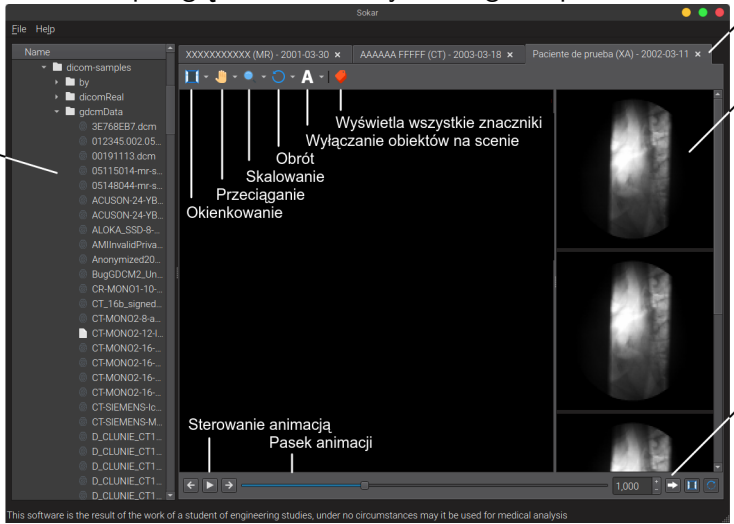
Qt jest zbiorem bibliotek i narzędzi programistycznych dedykowanych dla języków C++, QML i Java. Qt posiada bibliotekę do tworzenia interfejsu graficznego, oraz wiele innych rozwiązań ułatwiających programowanie obiektowe i zdarzeniowe.
Normy: IEC 62304:2015, IEC 61508:2010-3 7.4.4, ISO 9001:2015.
Posiada systemy rodzicielstwa i sygnałów.

GDCM to biblioteka do obsługi standardu DICOM. Posiada możliwość wczytywania plików z dysku jak i z lokalizacji sieciowych oraz wczytywania plików DICOMDIR. Ma wbudowaną dekompresję obrazów i obsługi różnych kodowań tekstu.

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

Projekt interfejsu graficznego

przeglądarka struktury katalogów i plików



Przeglądarka plików w systemie

Zakładki z różnymi plikami / animacjami

Ikony klatek animacji

Opcje animacji:
- prędkość animacji
- przemiatanie
- wspólne "okno"
- wspólne przekształcenie

Projekt interfejsu graficznego – scena

Litera orientacji

Dane pacjenta:
- imię i nazwisko
- identyfikator
- data urodzenia i wiek
- opis badania
- opis serii

Dane szpitala:
- nazwa instytucji
- producent i model urządzenia
- lekarz
- operator

Litera orientacji

Litera orientacji i podziałka

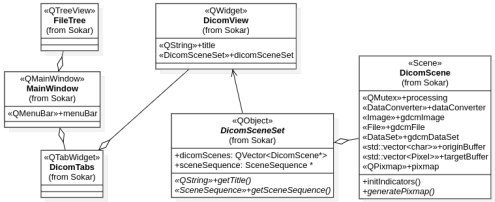
Dane akwizycji badania różnią się w zależności od modalności

Parametry z jakimi jest wyświetlany obraz

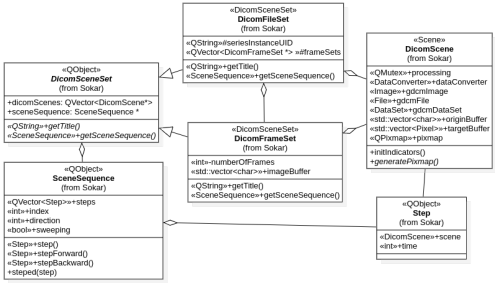
Podziałka i litera orientacji

Budowa obiektowa programu – perspektywy

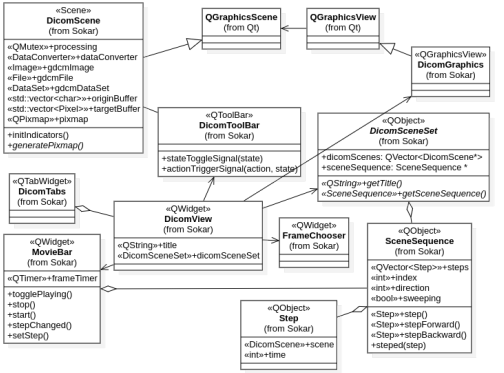
Układ globalnej struktury programu



Struktura przechowywania ramek obrazowych



Obiekt zakładki wyświetlający sekwencje ramek



Dekoder DICOM

Odczyt danych nieobrazowych

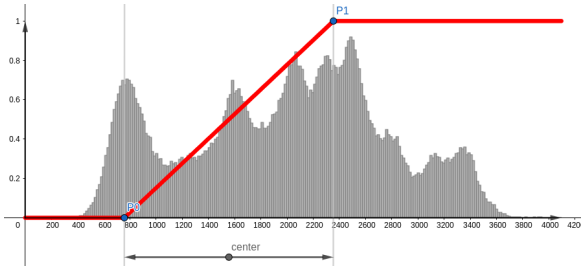
```
gdcm::Reader reader;
reader.SetFileName("/path/to/file");
if (!reader.Read()) return 1;

/* Pobranie obiektu pliku */
const gdcm::File &file = reader.GetFile();

/* Pobranie obiektu zbioru danych */
const gdcm::DataSet &dataset = file.GetDataSet();

/* Inicjalizacja klasy konwersji danych na string */
gdcm::StringFilter stringFilter;
stringFilter.SetFile(file);

/* Definicja znaczników wybranych elementów */
const static gdcm::Tag
    TagPatientName(0x0010, 0x0010),
    TagWindowCenter(0x0028, 0x1050);

/* Odczyt danej tekstowej */
if (dataset->FindDataElement(TagPatientName))
  string name = stringFilter.GetString(TagPatientName);

/* Odczyt danej binarnej */
if (dataset->FindDataElement(TagWindowCenter)) {
  auto& ele = dataset->GetDataElement(TagWindowCenter);
  quint16 center = ele.GetByteValue()->GetPointer(); }
```

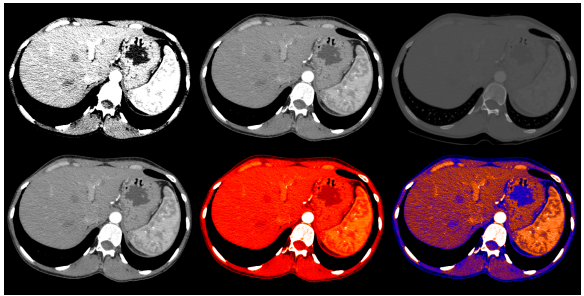
Odczyt obrazów

```
gdcm::ImageReader ir;
ir.SetFileName("/path/to/file");
if (!ir.Read()) return 1;

/* Pobranie obiektu obrazu */
const gdcm::Image &gimage = ir.GetImage();

/* Utworzenie buffora i jego wypełnienie */
std::vector<char> imgbuffer;
imgbuffer.resize(gimage.GetBufferLength());
gimage.GetBuffer(&imgbuffer[0]);

/* Odczyt parametrów obrazu */
const quint* dimension = gimage.GetDimensions();
quint dimX = dimension[0];
quint dimY = dimension[1];

if (gimage.GetPhotometricInterpretation() ==
    gdcm::PhotometricInterpretation::RGB)
    cout << "Jest to obraz RGB";

if (gimage.GetPhotometricInterpretation() ==
    gdcm::PhotometricInterpretation::MONOCHROME2)
    cout << "Jest to obraz monochromatyczny 2";

if (gimage.GetPixelFormat() == gdcm::PixelFormat::UINT8)
    cout << "Pixel jest 8-bitową liczbą całkowitą";
```

Poprawa kontrastu za pomocą “okienkowania”

Wydział Elektroniki i Technik Informacyjnych
POLITECHNIKA WARSZAWSKA





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

$$OutputUnits = m * SV + b$$

- m — wartość z RescaleSlope (0x0028, 0x1053),
- b — wartość z RescaleIntercept (0x0028, 0x1052),
- SV — stored values — wartość woksela z pliku,
- $OutputUnits$ — wartość wynikowa.

Implementacja

$$\begin{array}{lll} P_0: & x_0 = center - width/2 & y_0 = 1.0 \\ P_1: & x_1 = center + width/2 & y_1 = 0.0 \end{array}$$

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

$$\begin{array}{ll} x_0 -= rescaleIntercept & x_0 /= rescaleSlope \\ x_1 -= rescaleIntercept & x_1 /= rescaleSlope \end{array}$$

$$a = (y_1 - y_0)/(x_1 - x_0) \qquad b = y_1 - a * x_1$$

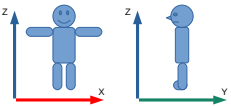
Tablica LUT

$8b$	$12b$	$16b$	$32b$	$64b$
$768B$	$196kB$	$196kB$	$12,5GB$	$55 * 10^6\,TB$

Orientacja pacjenta

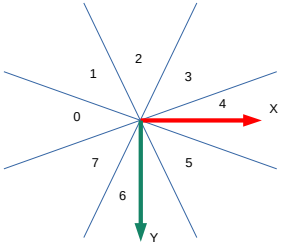
Zapis informacji o orientacji w DICOM

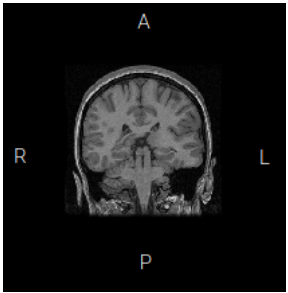
$$\begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} X_x\Delta_i & Y_x\Delta_j & 0 & S_x \\ X_y\Delta_i & Y_y\Delta_j & 0 & S_y \\ X_z\Delta_i & Y_z\Delta_j & 0 & S_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} i \\ j \\ 0 \\ 1 \end{bmatrix}$$



$$rotateTransform * \left(\begin{bmatrix} X_x & Y_x & 0 & 0 \\ X_y & Y_y & 0 & 0 \\ X_z & Y_z & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} * PatientPosition \right)$$

- P – koordynaty woksela we współrzędnych obrazu,
- S – trzy wartości z elementu ze znacznikiem Image Position,
- X, Y – położenie pierwszego wiersza i kolumny względem układu współrzędnych związanych z pacjentem
- i i j — oznaczają współrzędne na macierzy obrazu,
- Δ_i i Δ_j — rzeczywista wielkość piksela obrazu w mm.





Implementacja

$$PatientPosition = imgMatrix * ScenePosition$$

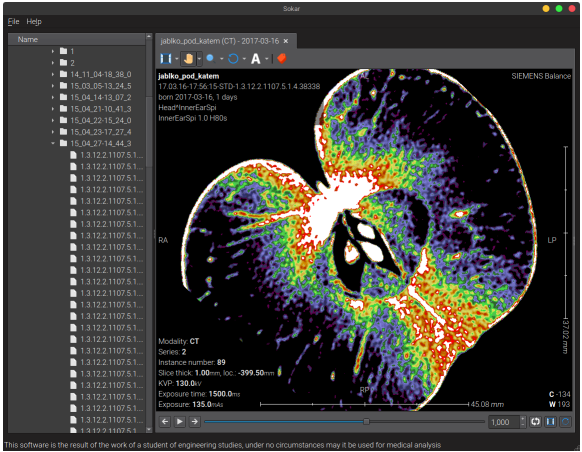
$$imgMatrix^{-1} * PatientPosition = imgMatrix^{-1} * imgMatrix * ScenePosition$$

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

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

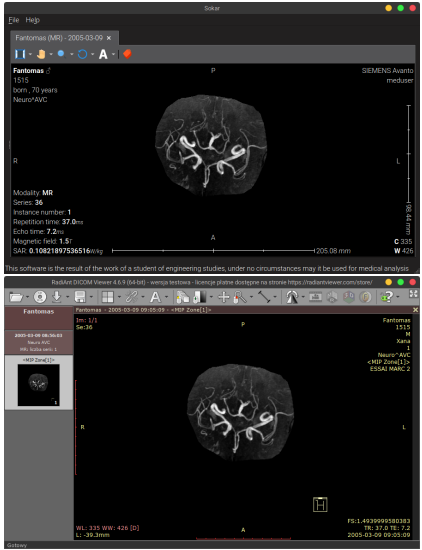
Funkcje przeglądarki

- Wprowadzono obsługę serii obrazów jako całości, włączając w to przegląd obrazów w serii, animacje, wspólne okna w skali barwnej oraz wspólne przekształcenia macierzowe.
- Obsługa podstawowych operacji na obrazie ułatwiających jego oglądanie i ocenienie, takich jak: przenoszenie, skalowanie, obrót.
- Dynamiczna zmiana parametrów wyświetlania obrazu, polepszanie kontrstu za pomocą "okienkowania" z możliwością zastosowania wielokolorowych palet.
- Wyliczanie orientacji pacjenta w stosunku do obrazu.
- Przeglądanie obrazów w systemie.



Testy

- Zapewniono jednolity sposób kompilacji na platformach przy użyciu narzędzia CMake.
- Napotkano problem z biblioteką GDCM – biblioteka wymaga kompilacji przez użytkownika.
- Skompilowano i przetestowano na platformach: Linux, MacOS i Windows.
- Wykonano testy na obrazach różnych modalności:
  - CT z zakładowego tomografu
  - MR z zakładowego tomografu
  - USG zapisanego w formacie RGB
  - Medycyna nuklearna – obsługa obrazów multiframe
- Wykonano testy algorytmu wyznaczania orientacji pacjenta na obrazie – porównanie z innymi przeglądarkami



Wnioski

- Opracowano aplikację do obsługi obrazów DICOM w C++ z możliwością kompilacji na wiele platform.
- Zastosowano biblioteki dostępne na różnych platformach: Qt i GDCM, które również zostały napisane w C++, dzięki czemu uzyskano jednolity program napisany w jednym języku.
- Jednolity wygląd aplikacji zapewniła biblioteka Qt, co sprawia, że interfejs aplikacji jest prawie taki sam na każdym systemie.
- Przeprowadzone testy pokazały poprawność implementacji algorytmów.