|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| agh | **Wizualizacja w systemach biomedycznych** | | | **Wykonał**:  Łukasz Uszko |
| Aplikacja typu demo prezentująca możliwości konturowania | | |
| **Wydział:** EAIiIB | | **Kierunek:** Elektrotechnika PTiB | **Rok akad:**  2013/2014 | **Data wykonania:**  22.06.2014 |

**1.Wstęp.**

Celem projektu było zaprojektować i wykonać aplikację prezentującą możliwości klasy vtkContourFilter. Program został wykonany na systemie operacyjnym Ubuntu 12, w programie QtCreator 3.0 przy wykorzystaniu bibliotek VTK 5.2 oraz QT 4.8.4.

**2. Założenia**

1. Wczytanie danych
2. GUI do sterowania parametrami konturowania
3. Możliwość przypisania koloru do konturów
4. Sterowanie trybem wizualziacji (punkty, wireframe, ciągła powierzchnia...)
5. Zapis wyników do bitmapy, wraz z legendą, opisem.

**3. Opis aplikacji**

##### Napisany program to aplikacja typu demo prezentująca możliwości konturowania klasy [vtkContourFilter](http://www.vtk.org/doc/release/5.6/html/a00379.html) która jest częścią biblioteki vtk. W obszarze, którym jest sześcian, o zadanych w kodzie programu długościach krawędzi (zakładka Dimensions), użytkownik może wygenerować obiekt wybierając go spośród 5 dostępnych w zakładce „Choose the object” a następnie z pomocą wciśnięcia przycisku „Retrieve”.

##### Wybrany przez użytkownika obiekt zostanie wczytany do programu i z wartościami domyślnymi wyświetlony na ekranie. Po wczytaniu obrazu w oknie programu znajduję się menu użytkownika składające z 2 zakładek:

##### - ContourFilter\_Menu,

##### - Shape: [nazwa wybranego kształtu].

##### Zmiana nastaw poszczególnych parametrów dla danego wczytanego obiektu jest intuicyjna i polega na ustawianiu wartości za pomocą suwaków oraz spinboxów.

##### Użytkownik do wyboru ma 5 kształtów:

##### - Cylinder,

##### - Sfera,

##### - Box,

##### - Stożek,

##### - oraz kształt Quadric który reprezentuje graficzny opis funkcji :

##### .

##### Poszczególne współczynniki tej funkcji można zmieniać z poziomu aplikacji.

##### Aplikacja umożliwia modyfikowanie zadanego obiektu poprzez ustawienie jego promienia minimalnego oraz maksymalnego, ograniczających przestrzeń, w której zawierają się kolejne kontury obiektu (Pola „min Value” oraz „max Value” w menu ustawień) W polach min i max możemy przypisać promieniom dane wartości. Liczbę wyświetlanych konturów danego obiektu można ustawić za pomocą suwaka „nr of contours” (wartości od 0 do 10).

##### Po ustawieniu wszystkich pożądanych parametrów możemy zapisać obraz do pliku w postaci .PNG klikając menu „File – Save As…”.

##### Istnieje również możliwość wczytania do programu plików z rozszerzeniami „.vtk , .png, .pdb, .img, .dcm” (menu „File – Open”) - niestety w tej chwili odpowiednie konturowanie każdego z wczytanych obrazów jest niedostępne.

##### 4. Prezentacja aplikacji.

##### Poniżej przedstawiono kilka zrzutów z działania aplikacji:

##### C:\Users\lukasz\Desktop\faza\faza\Screenshot from 2014-06-22 10_23_07.png

##### *Rys. 1. Sfera –liczba konturów: 7*

##### C:\Users\lukasz\Desktop\faza\faza\Screenshot from 2014-06-22 10_24_27.png

##### *Rys. 2. Obiekt wygenerowany za pomocą funkcji Quadric.*

##### C:\Users\lukasz\Desktop\faza\faza\Screenshot from 2014-06-22 10_25_33.png

##### *Rys. 3 Obiekt Box.*

##### C:\Users\lukasz\Desktop\faza\faza\Screenshot from 2014-06-22 10_26_23.png

##### *Rys. 4. Wczytany plik PDB*

##### 5. Omówienie ciekawszych miejsc w kodzie programu

##### - Zapisywanie pliku:

|  |
| --- |
| bool MainWindow::saveFile( ){  vtkSmartPointer<vtkWindowToImageFilter> win2imgFilter= vtkSmartPointer<vtkWindowToImageFilter>::New();  win2imgFilter->SetInput(renWin);  win2imgFilter->SetMagnification(3); // the resolution of the output image 3 times bigger  win2imgFilter->SetInputBufferTypeToRGBA(); //  win2imgFilter->Update();  vtkSmartPointer<vtkPNGWriter> writer= vtkSmartPointer<vtkPNGWriter>::New();  QString imageName = QFileDialog::getSaveFileName(this,tr("Save image"),"picture",tr("vtkImages (\*.png)"));  writer->SetInputConnection(win2imgFilter->GetOutputPort());  writer->SetFileName(imageName.toAscii().data());  writer->Write();  return true;  } |

##### - Wczytywanie pliku:

|  |
| --- |
| void MainWindow::loadImage(const QString& imageName){  QFile file(imageName);  if(!file.open(QFile::ReadOnly)){  QMessageBox::warning(this,tr("App"),tr("Cannot read file %1: \n%2.").arg(imageName)  .arg(file.errorString()));  return ;  }  QApplication::setOverrideCursor(Qt::WaitCursor);  loadProperTypeOfFile(imageName);  QApplication::restoreOverrideCursor(); }//metoda loadProperTypeOfFile( ) -> obsługuje strumienie wejściowe dla danego typu wczytywanego pliku |

##### - Ustawienie menu (File, Help):

|  |
| --- |
| void MainWindow::setMenuOptions(){  QMenu \*fileMenu= new QMenu(tr("&File"),this);  QAction \*openFileAction= fileMenu->addAction(tr("&Open..."));  openFileAction->setShortcuts(QKeySequence::Open);  QAction \*saveFileAction= fileMenu->addAction(tr("&Save As..."));  saveFileAction->setShortcuts(QKeySequence::SaveAs);  fileMenu->addSeparator();  QAction \*quitFileAction= fileMenu->addAction(tr("&Exit"));  quitFileAction->setShortcuts(QKeySequence::Quit);  QMenu \*helpMenu= new QMenu(tr("&Help"),this);  QAction \*aboutAction= helpMenu->addAction(tr("&About"));  connect(openFileAction,SIGNAL(triggered()),this,SLOT(openFile()));  connect(saveFileAction,SIGNAL(triggered()),this,SLOT(saveFile()));  connect(quitFileAction,SIGNAL(triggered()),qApp,SLOT(quit()));  connect(aboutAction,SIGNAL(triggered()),this,SLOT(aboutMessage()));  ui->menuBar->addMenu(fileMenu);  ui->menuBar->addMenu(helpMenu);  QMenuBar\* bar = new QMenuBar();  ui->mainToolBar->addWidget(bar);  bar->addMenu(fileMenu);  bar->addMenu(helpMenu);  }  // funkcje menu wykorzystują obiekt QAction |

##### - Krótki opis szkieletu programu: fabryki „kształtów”

##### Część programu służąca do wyświetlania wygenerowanych obiektów składa się z głównej klasy abstrakcyjnej „Shape” po której dziedziczą klasy Cylinder,Box, Quadric, Cone oraz Sphere, nadpisując metody wirtualne klasy bazowej. Każda klasa posiada własną przeładowaną metodę to pobrania UI każdej z nich: QGroupBox\*getGivenShapeUI(void). Zwraca ona wskaźnik do obiektu QGroupBox przechowującego menu sterowania dla każdego kształtu. Następnie została utworzona klas ShapeFactory która jest odpowiedzialna za utworzenie wybranego obiektu i zwrócenie go w postaci wskaźnika do nowo utworzonego obiektu Shape : Shape\* ShapeFactory::getShape(const QString& shapeType) .

##### 6. Podsumowanie

##### Projekt nie został do końca kompletnie zrealizowany z powodu chronicznego braku czasu wykonującego. Nie zaimplementowano obsługi konturowania na plikach wejściowych. Napisany kod pozwala jednak na łatwą rozbudowę programu ( napisany zorientowanie obiektowo umożliwia na łatwą jego rozbudowę) .

##### 7. Bibliografia

##### www.vtk.org

##### www.qt-project.org