

# Wprowadzenie do VTK

Materiały pomocnicze do ćwiczeń z przedmiotu

*Wizualizacja danych biomedycznych*

Mirosław Socha

Katedra Metrologii i Elektroniki AGH  
Kraków 2014

# Instalacja VTK w Ubuntu

## Menadżer pakietów Synaptic:

- główne pakiety **vtk5**:

```
libvtk5.8 libvtk5-dev
```

- rozszerzenie do Qt **vtk5-qt4**:

```
libvtk5.8-qt4 libvtk5-qt4-dev
```

- wrapery:

```
python-vtk python-vtkgdcml vtk-tcl
```

- dodatki:

```
vtk-example vtkdata vtk-doc libvtkgdcml-tools
```

S	Pakiety
<input checked="" type="checkbox"/>	libvtk5-dev
<input checked="" type="checkbox"/>	libvtk5-qt4-dev
<input checked="" type="checkbox"/>	libvtk5.8
<input checked="" type="checkbox"/>	libvtk5.8-qt4
<input checked="" type="checkbox"/>	python-vtk
<input checked="" type="checkbox"/>	tcl-vtk
<input checked="" type="checkbox"/>	vtk-examples
<input checked="" type="checkbox"/>	vtkdata

## Instalacja z konsoli:

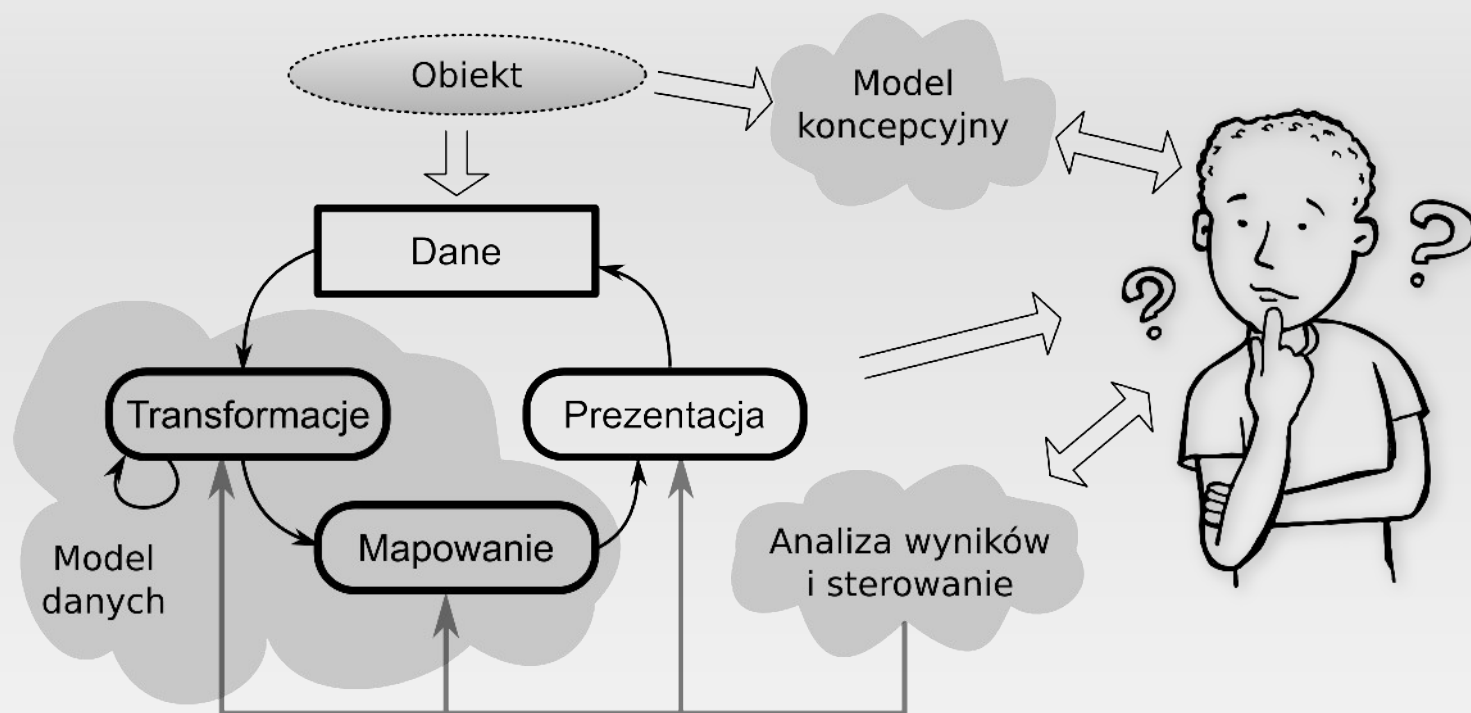
- `sudo apt-get install nazwa_pakietu`

# Visualization ToolKit (VTK)



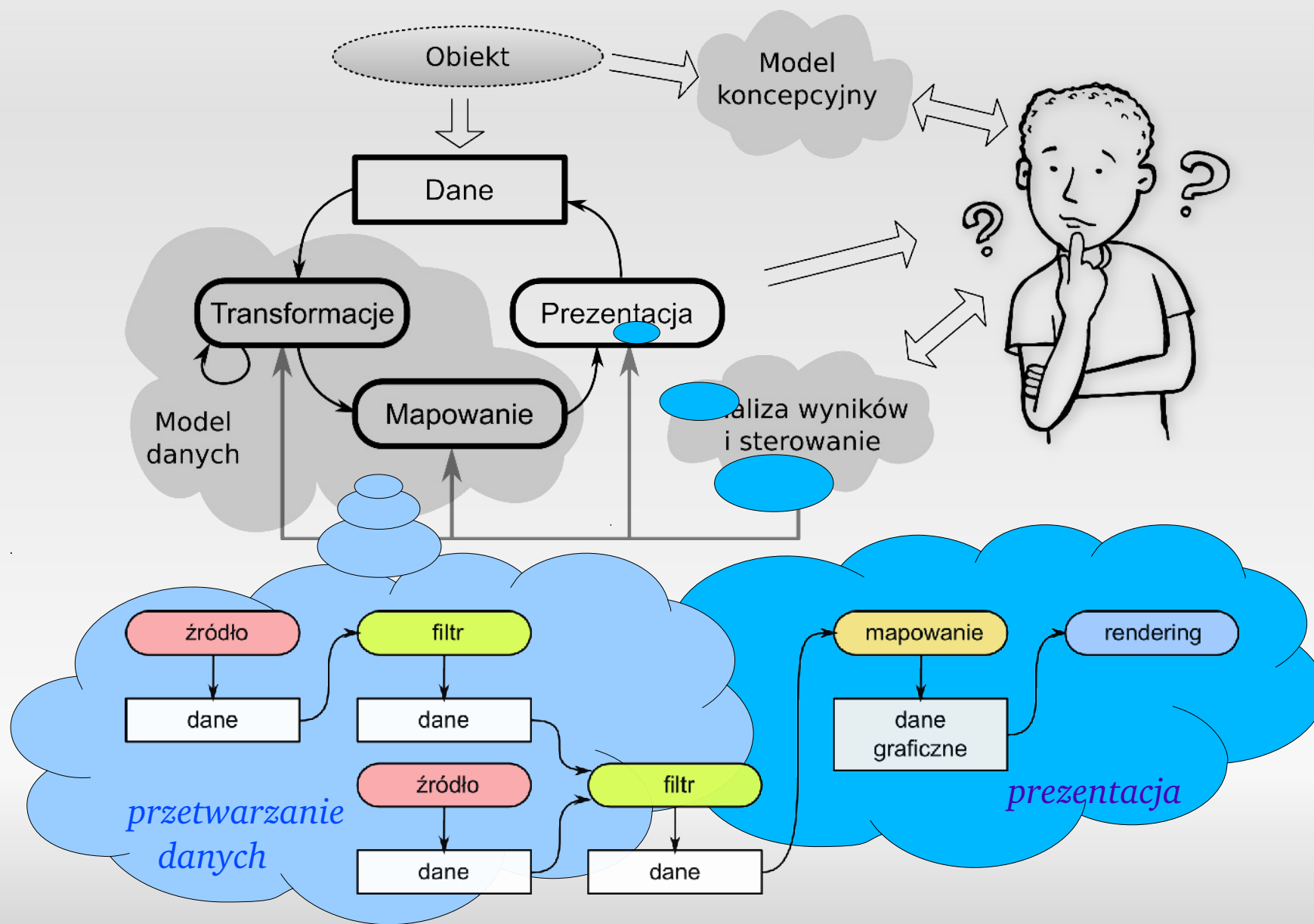
- otwarta biblioteka do wizualizacji i przetwarzania obrazów
- napisana w C++ (kilka tysięcy klas)
- dostępna w postaci kodu źródłowego
- konfiguracja kodu przez *cmake*
- wykorzystuje **OpenGL** i **MPI**
- interfejs do języków Tcl/Tk, Python, Java
- zarządzanie Kitware: [www.kitware.com](http://www.kitware.com)
- bardzo popularna i intensywnie rozwijana!

# Wizualizacja – co to jest?



*Iteracyjne przetwarzanie i prezentacja danych,  
których celem jest poznanie i zrozumienie  
badanego Obiektu*

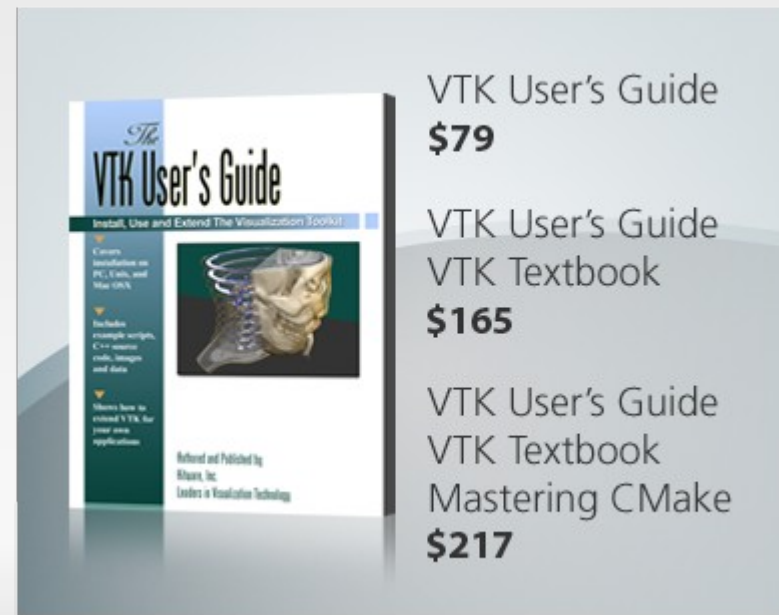
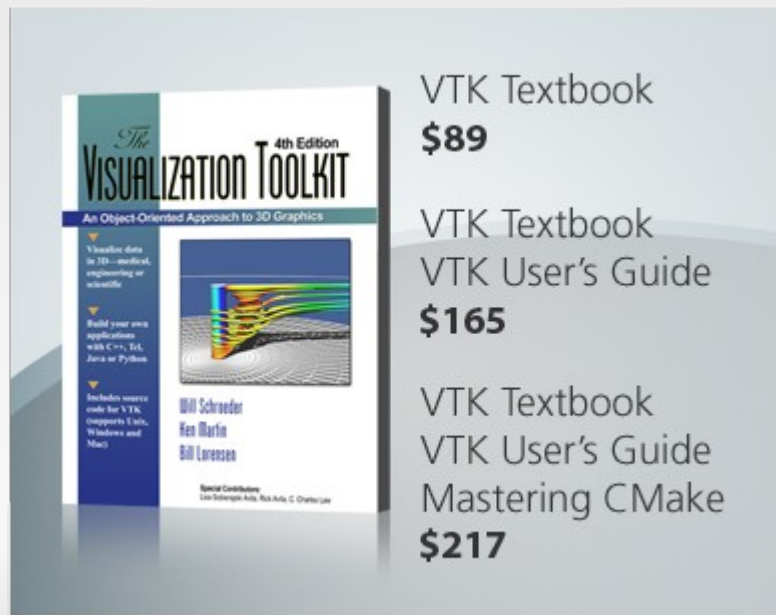
# Wizualizacja – realizacja w VTK



# Dokumentacja

**Książki Kitware** – bardzo dobra baza wiedzy!

- Visualization Toolkit – teoria
- VTK User's Guide – praktyka

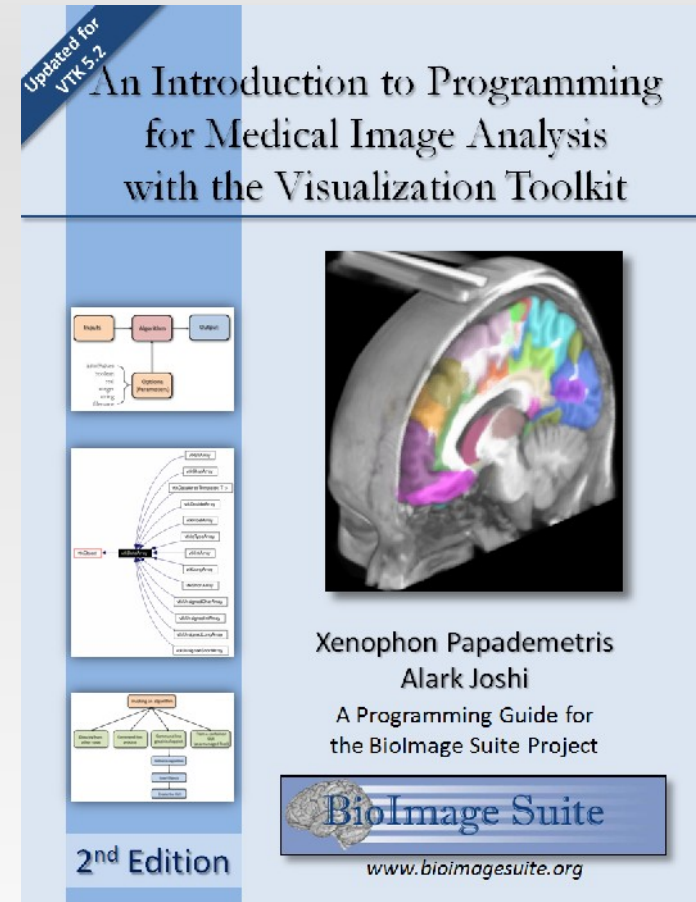


# Dokumentacja

Xenophon Papademetris and Alark Joshi:

## "Introduction to Programming for Image Analysis with VTK - 2<sup>nd</sup> Edition"

- darmowa, do pobrania **pod tym adresem**
- "...for the graduate seminar "*Programming for Medical Image Analysis*" (ENAS 920a) that was taught at Yale University, Department of Biomedical Engineering, in the Spring of 2009"



Tcl/Tk, VTK, ITK

# Dokumentacja on-line

[www.vtk.org](http://www.vtk.org)

Podstawowe  
źródło wiedzy!

- .../VTK/help/documentation.html
  - Doxygen: <http://www.vtk.org/doc/release/5.8/html/>
- .../Wiki/VTK
  - <http://www.vtk.org/Wiki/VTK/Examples>
  - <http://www.vtk.org/Wiki/VTK/Examples/Cxx>
  - <http://www.vtk.org/Wiki/VTK/Tutorials>
- Mailing-list: Users, Developer
- <http://www.bu.edu/tech/research/training/tutorials/vtk/#INTRO>
- ...



# Pierwszy program!

- Kopiujemy zawartość:

```
/usr/share/vtk/Tutorial/
```

do własnego katalogu, np. komendą w konsoli:

```
cp --recursive /usr/share/vtk/Tutorial ~/work/vtk/
```

- Przechodzimy do katalogu:

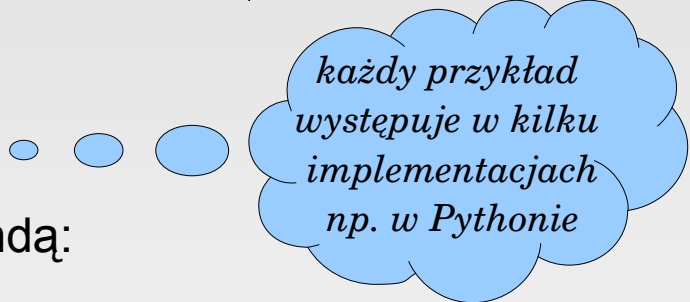
```
cd ~/work/vtk/Tutorial/Step1/Python/
```

- Uruchamiany program (skrypt w języku Python) komendą:

```
./Cone.py      lub      python Cone.py
```

- Podglądamy kod, zwracając uwagę na połączenie obiektów w strumień wizualizacji:

```
gedit ./Cone.py
```



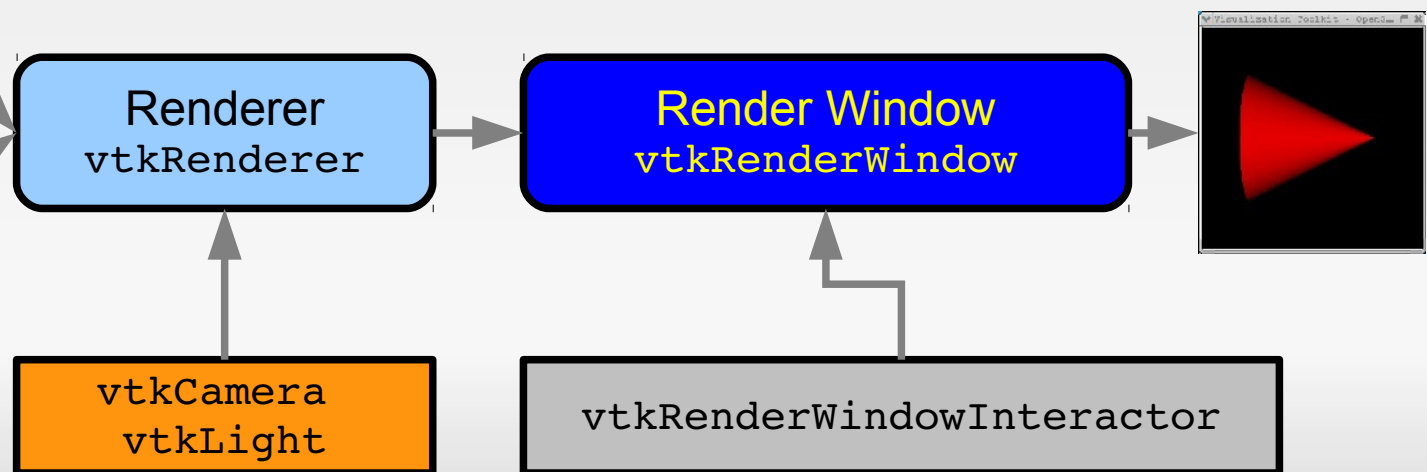
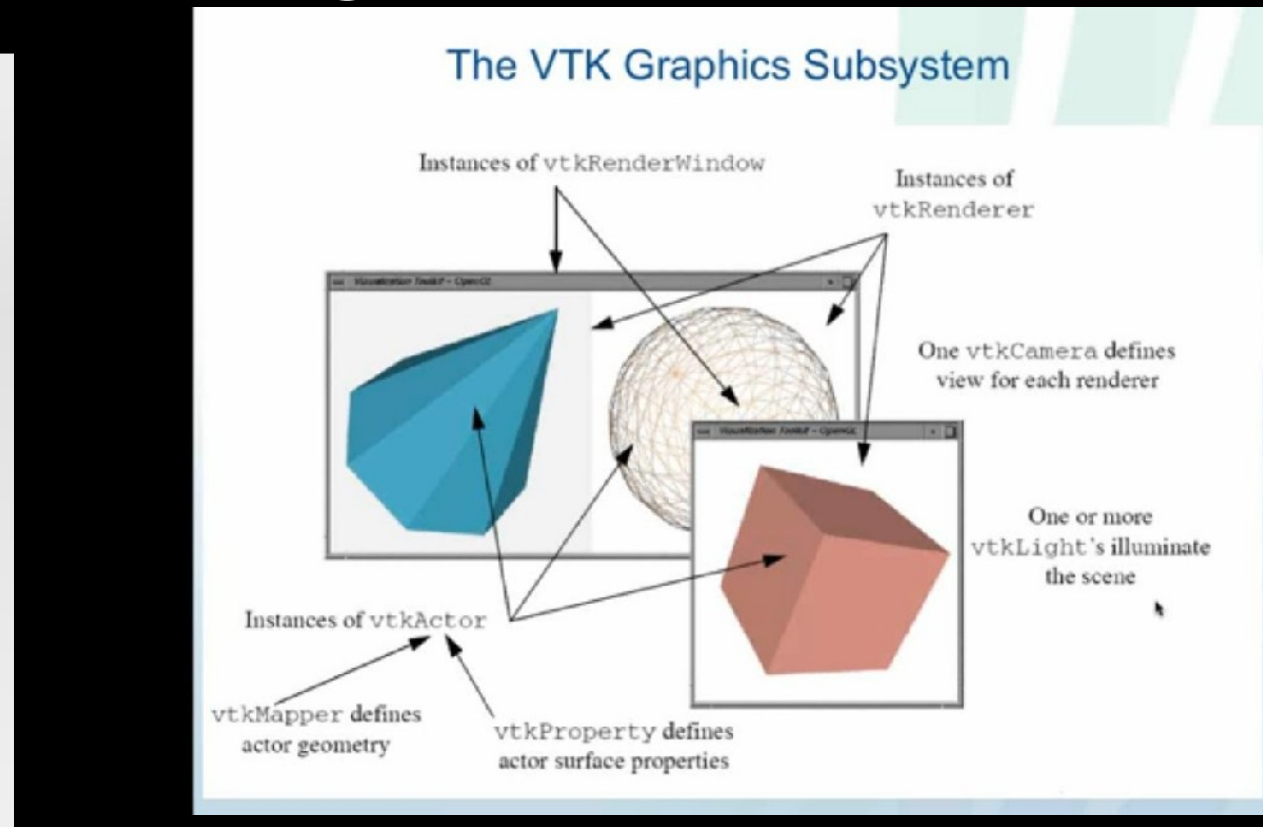
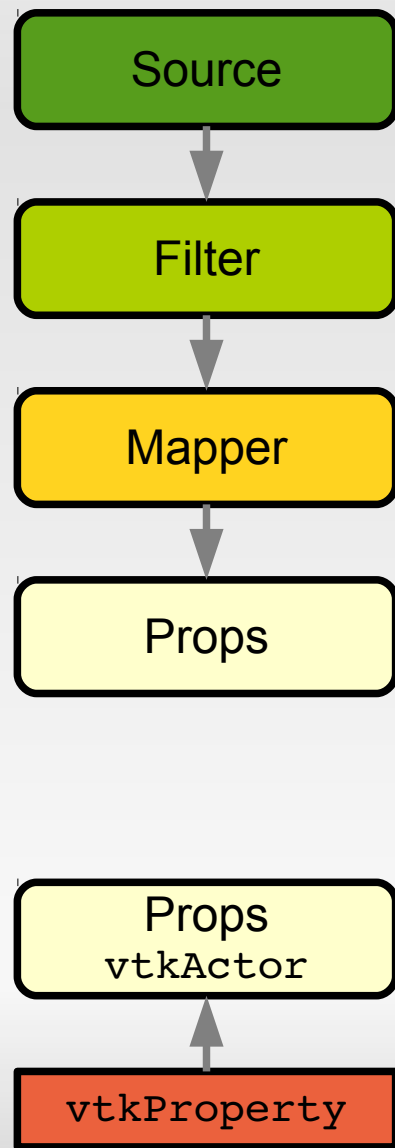
*każdy przykład  
występuje w kilku  
implementacjach  
np. w Pythonie*

## Zadanie:

Zmień prędkość i kierunek obrotu stożka bazując na dokumentacji klasy `vtkCamera` :)

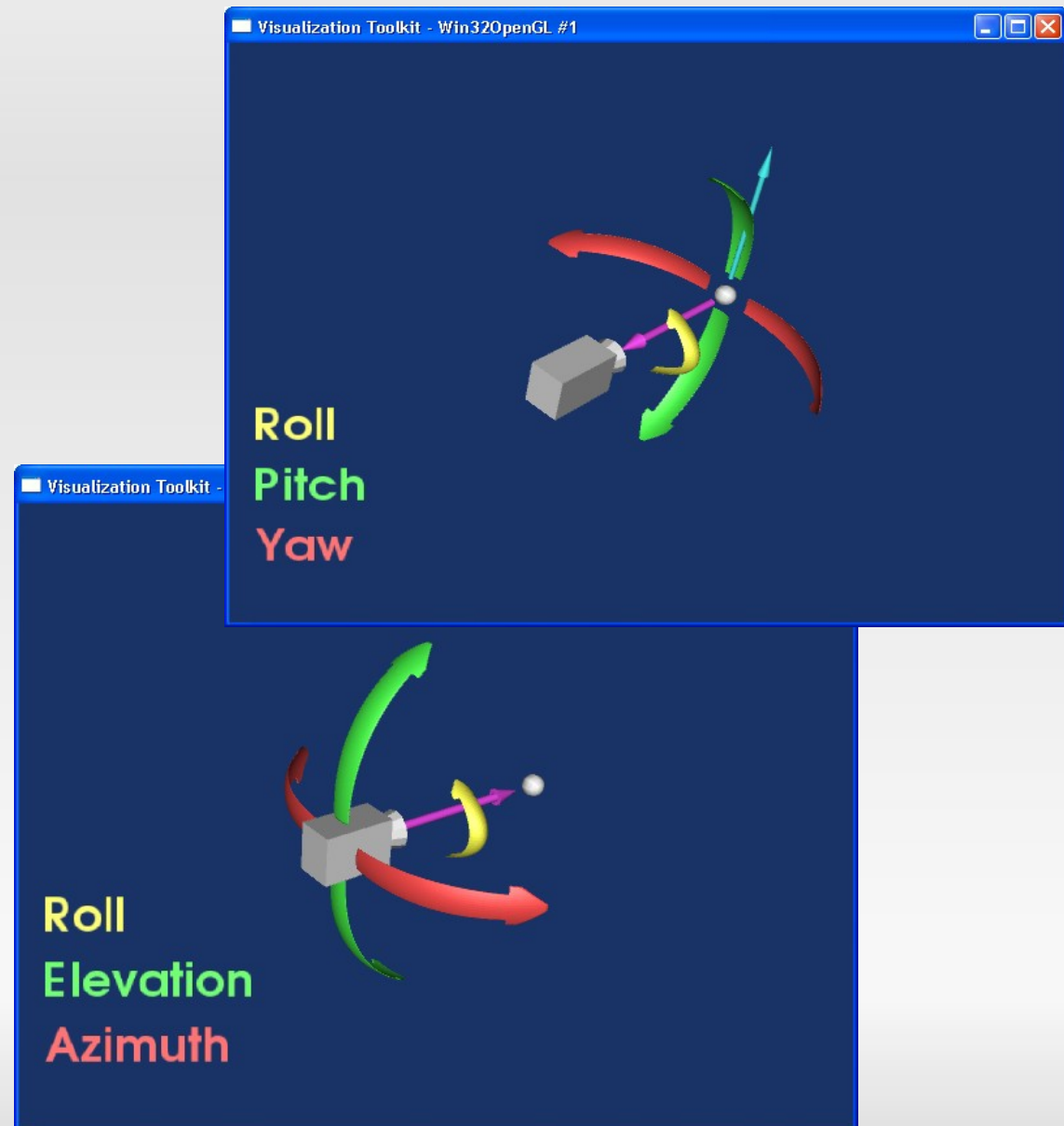
# System graficzny VTK

Strumień wizualizacji VTK:



# vtkCamera

- **dokumentacja** jest głównym źródłem wiedzy o klasach VTK :)
- co przedstawiają diagramy?
- jakie klasa ma metody?
- **Przykładowy kod:**
  - **Examples:**  
zobacz Medical\*
  - **Tests**



# Ciekawe przykłady:

Najlepiej odpalać z konsoli:

- `/usr/share/vtk/Medical/Python/`
- `/usr/share/vtk/VolumeRendering/Python`
- `/usr/share/vtk/Modelling/Python`
- `/usr/share/vtk/VisualizationAlgorithms/Python`

Przykłady do poprawnej pracy wymagają danych, czyli instalacji pakietu `vtk-data`.

Jak uruchomić przykład VTK w Pythonie z sieci? Np. z wiki lub dokumentacji:

`http://www.vtk.org/Wiki/VTK/Examples/Python`

1. tworzymy pusty plik,
2. kopiujemy zawartość przykładu, zapisujemy,
3. nadajemy prawa do uruchomienia (np. w konsoli komenda `chmod` lub prawy klawisz myszki na pliku, a potem: *Właściwości/Uprawnienia...* )
4. uruchamiamy :)



Jak skompilować  
przykład VTK w C++?

# Przykład w C++ ?

- Kod C++ biblioteki VTK (przykłady jak i sama biblioteka) jest niezależny od platformy (architektury CPU, kompilatora itp.) i przed kompilacją wymaga specjalnego przygotowania!
- Zadanie to ułatwia program `cmake` i jego graficzna nakładka `cmake-gui`:
  - do pobrania w postaci paczki w Synaptic lub w konsoli komenda:

```
sudo apt-get install cmake-qt-gui
```
  - lub ze strony [www.cmake.org](http://www.cmake.org)

W wyniku jego działania powstaje gotowy do skompilowania projekt :)

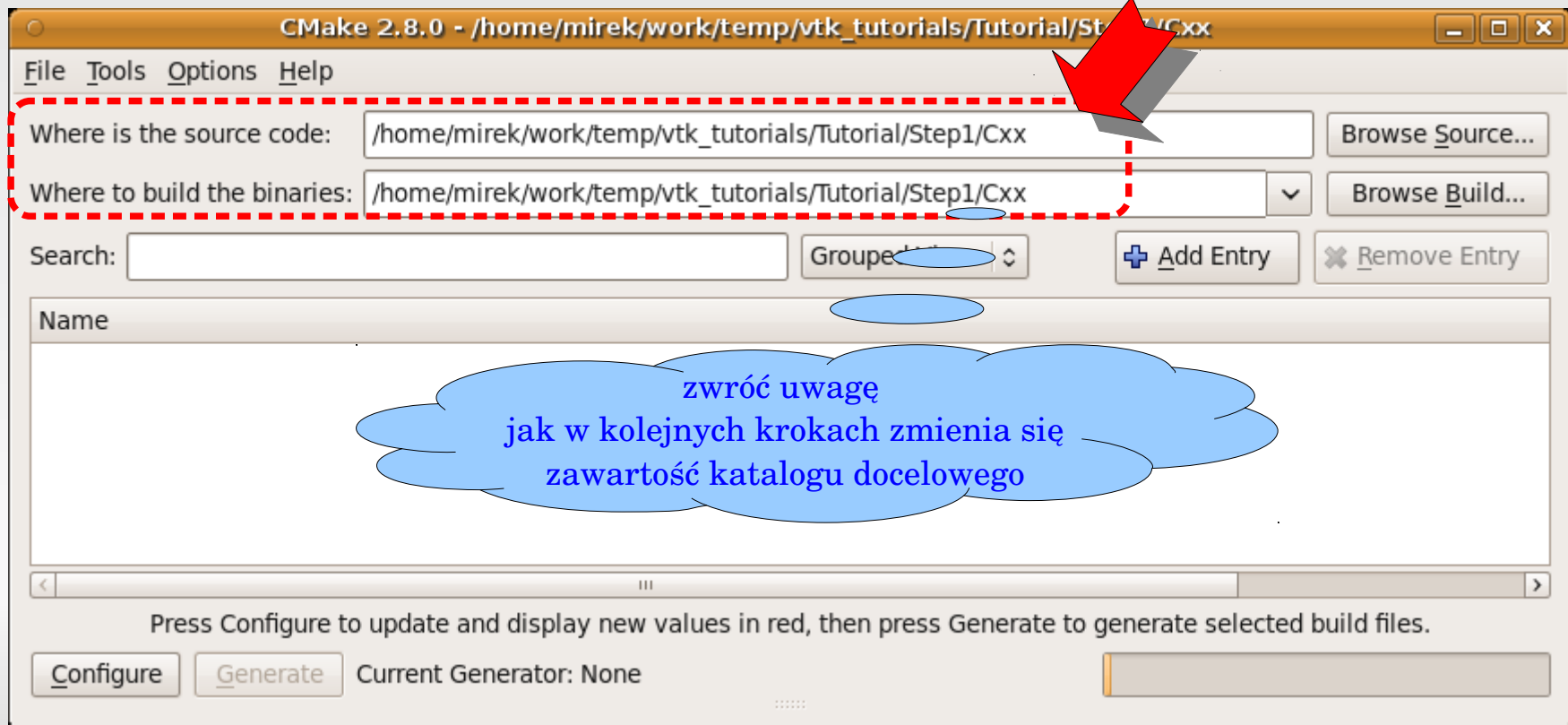
# Przygotowanie kodu C++

1

## ZADANIE: skompilować program z katalogu Step1:

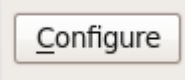
- Uruchamiamy cmake w konsoli komendą: `cmake-gui` .
- Podajemy ścieżkę dostępu do „surowego” kodu (np. katalog *Tutorial* projekt *Step1*, wersja w języku C++) oraz do katalogu z przyszłym projektem (mogą być różne):

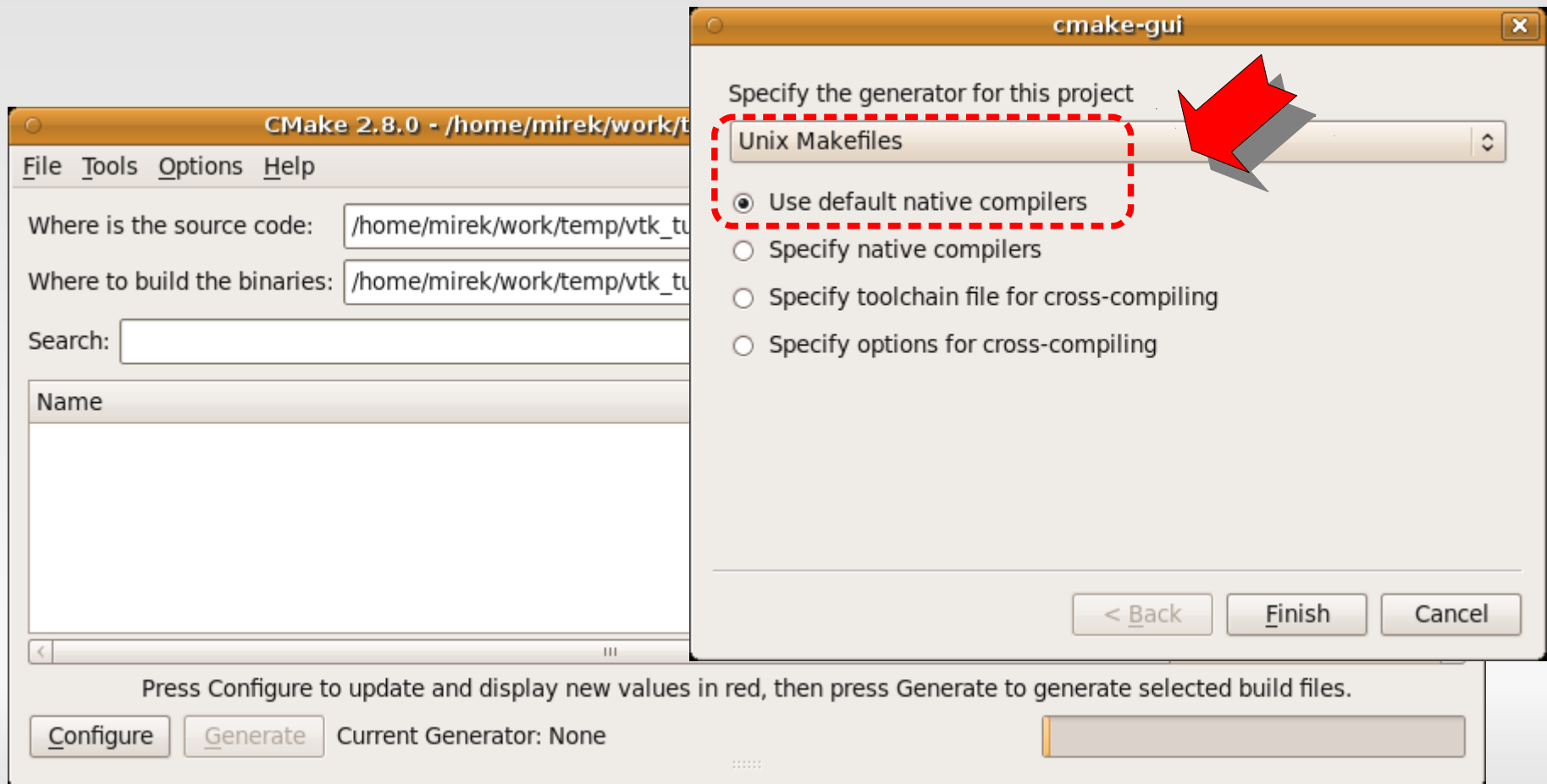
ta kropka wskazuje na katalog bieżący!



# Przygotowanie kodu C++

2

- Uruchamiamy konfigurację: 
- Wybieramy typu projektu i kompilator:

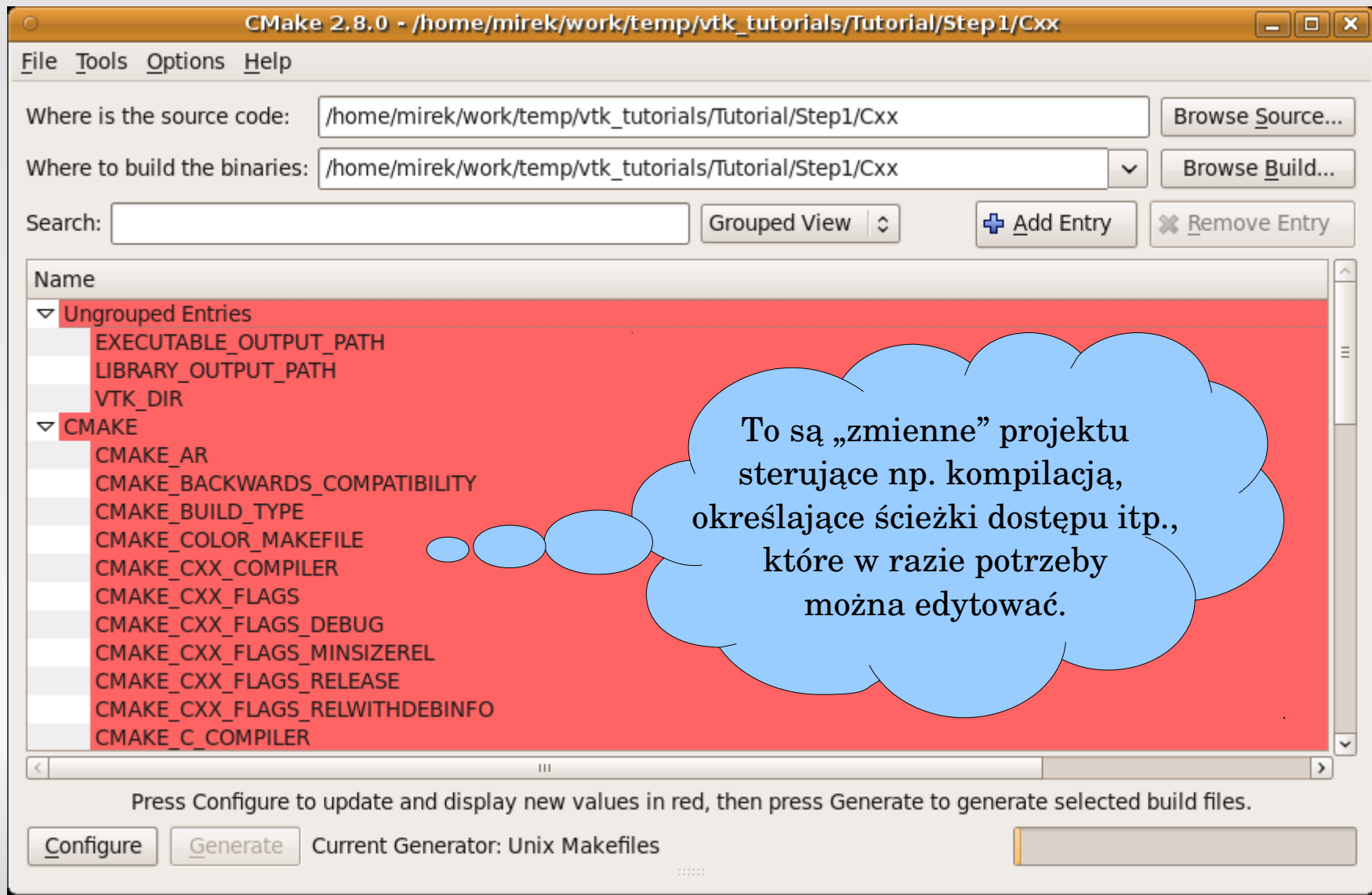




# Przygotowanie kodu C++

3

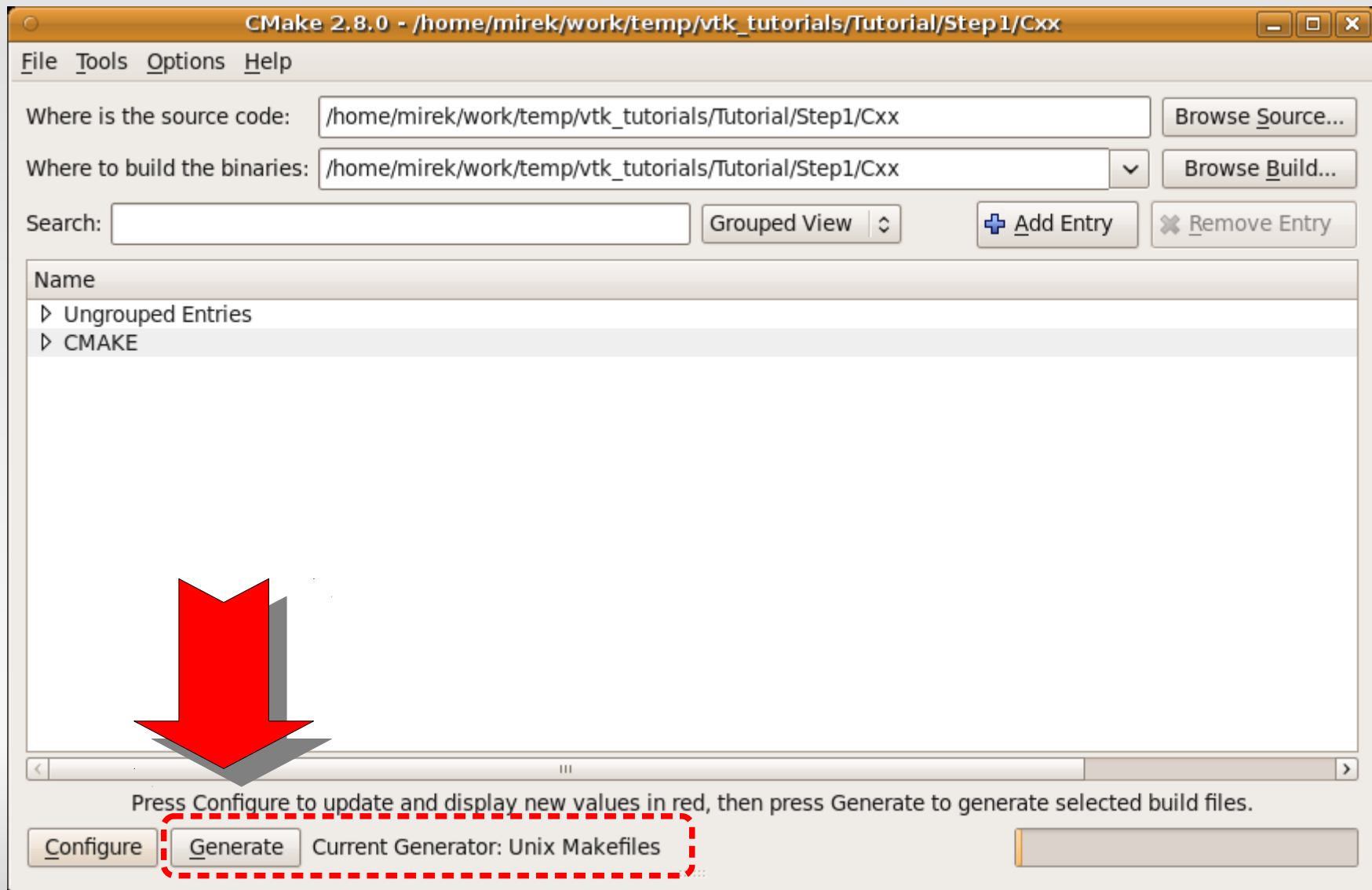
- Po zatwierdzeniu, cmake analizuje kod i pokazuje „co znalazł”:



# Przygotowanie kodu C++

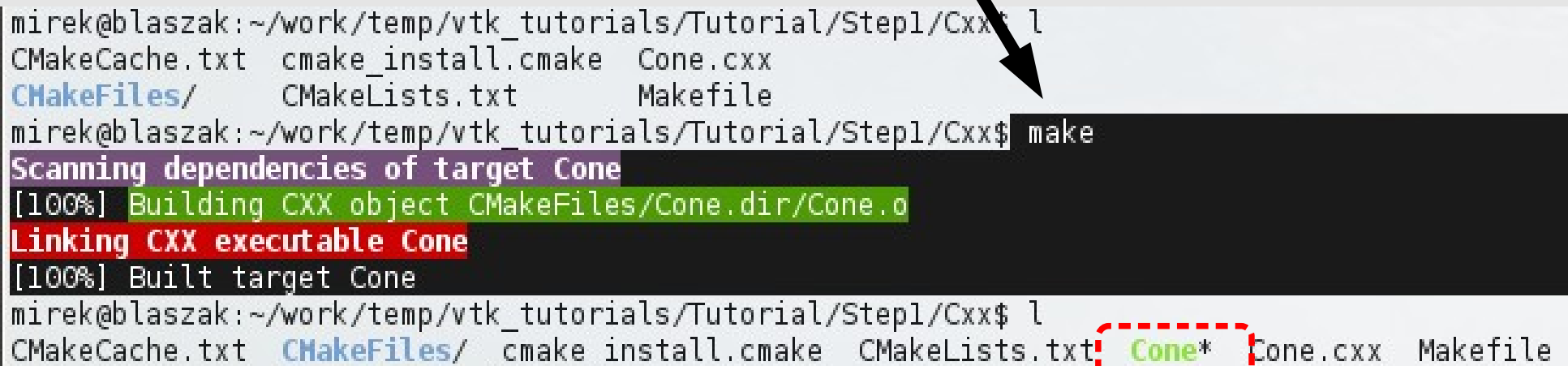
4

- Zatwierdzamy klawiszem , aż znikną „czerwone” pola, i można będzie wygenerować projekt (klawiszem *Generate*):



# Kompilacja kodu C++

- Po wygenerowaniu projektu można go skompilować! :)
- W Ubuntu kompilujemy poleceniem *make*:



```
mirek@blaszak:~/work/temp/vtk_tutorials/Tutorial/Step1/Cxx$ l
CMakeCache.txt  cmake_install.cmake  Cone.cxx
CMakeFiles/      CMakeLists.txt          Makefile
mirek@blaszak:~/work/temp/vtk_tutorials/Tutorial/Step1/Cxx$ make
Scanning dependencies of target Cone
[100%] Building CXX object CMakeFiles/Cone.dir/Cone.o
Linking CXX executable Cone
[100%] Built target Cone
mirek@blaszak:~/work/temp/vtk_tutorials/Tutorial/Step1/Cxx$ l
CMakeCache.txt  CMakeFiles/  cmake_install.cmake  CMakeLists.txt  Cone*  Cone.cxx  Makefile
```

- W wyniku kompilacji otrzymujemy program, który uruchamiamy komendą:

**./Cone**

# Szybkie przygotowanie i kompilacja kodu C++

Jeżeli program jest sprawdzony (np. jest to przykład z dokumentacji) i nie chcemy wprowadzać zmian w sposobie kompilacji, możliwe jest pominięcie interfejsu graficznego i szybkie przygotowanie i kompilacja takiego kodu C++.

Wymaga to wydania następujących komend:

- w katalogu projektu wydajemy komendę:

```
cmake .
```

- jeżeli nie wystąpią błędy, kompilujemy przygotowany kod:

```
make
```

- w wyniku kompilacji otrzymujemy program, który uruchamiamy, np.:

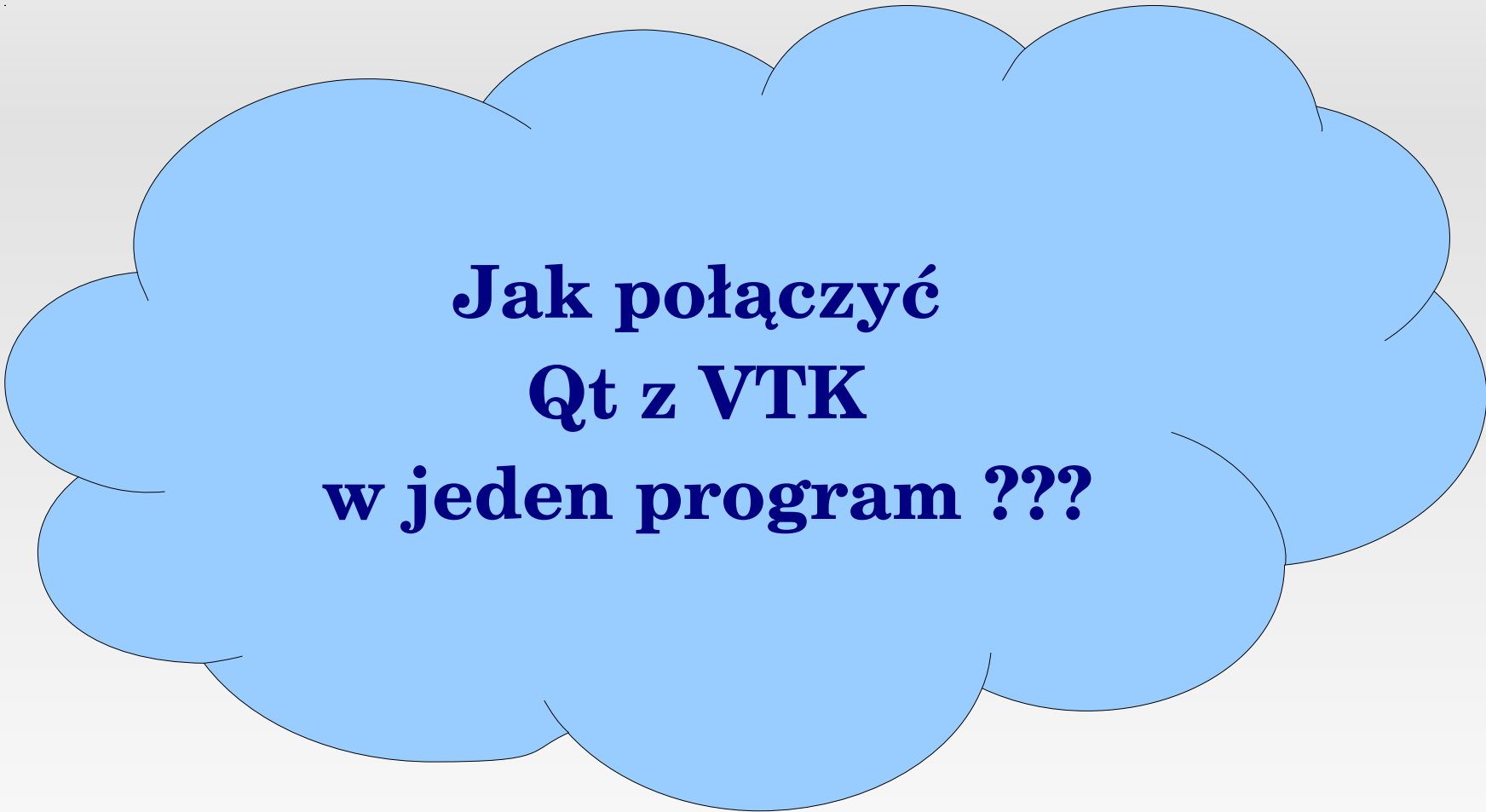
```
./Cone1
```

# Modyfikacja kodu

Po modyfikacji kodu C++ zazwyczaj wystarcza jego kompilacja, czyli wywołanie programu *make*.

## ZADANIA:

- Bazując na doświadczeniach zdobytych przy modyfikacji skryptu w języku Pythona zmień kierunek i prędkość obrotu stożka w kodzie C++ (kod z przykładu *Tutorial/Step1*)
- Zmień kolor stożka używając klasy *vtkProperty* (przykład użycia można zobaczyć w *Step4*)
- Zastąp animację (pętlę *for*) możliwością interakcji z użytkownikiem (klasa *vtkRenderWindowInteractor*, przykład *Step5*)
- Przetestuj działanie kontrolki graficznej (*widgetu*) użytej w przykładzie *Step6*, przeanalizuj kod źródłowy.



**Jak połączyć  
Qt z VTK  
w jeden program ???**

**Część nieobowiązkowa, ale bardzo ciekawa ;)**

- uruchamiamy *Qt Creator*
- tworzymy pusty projekt: „*Aplikacja GUI Qt*”
- do pliku projektu (\*.pro) dodajemy biblioteki:

```
LIBS      += -L/usr/lib/vtk-5.8 -lvtkCommon\  
-lvtksys -lQVTK -lvtkViews -lvtkWidgets\  
-lvtkInfoviz -lvtkRendering -lvtkGraphics\  
-lvtkImaging -lvtkIO -lvtkFiltering\  
-lvtkDICOMParser -lvtkHybrid
```

oraz ścieżkę dostępu:

```
INCLUDEPATH += /usr/include/vtk-5.8
```

Przy kopiowaniu  
tekstu z PDF'a  
pojawiają się  
niewidoczne znaki!  
Myślnik zastępowany  
jest kilkoma znakami,  
trzeba to poprawić  
ręcznie.

# Konfiguracja Qt Creator

2

Zawartość pliku projektu (\*.pro)

```
12
13 SOURCES += main.cpp\
14          mainwindow.cpp
15
16 HEADERS  += mainwindow.h
17
18 FORMS    += mainwindow.ui
19
20 LIBS     += -L/usr/lib/vtk-5.8 -lvtkCommon\
21            -lvtksys -lQVTK -lvtkViews -lvtkWidgets\
22            -lvtkInfovis -lvtkRendering -lvtkGraphics\
23            -lvtkImaging -lvtkIO -lvtkFiltering\
24            -lvtkDICOMParser -lvtkHybrid
25
26 INCLUDEPATH += /usr/include/vtk-5.8
27 |
```

jedna linia!!!  
lub łamanie  
z użyciem \

Sprawdzenie położenia biblioteki vtk (w konsoli):

- whereis vtk
- locate vtk



# Konfiguracja Qt Creator

3

The screenshot shows the Qt Creator IDE with a new Qt VTK widget being configured. The central canvas displays a black rectangle with the text "Wpisz tutaj" (Write here) above it. A red arrow points from the "QVTKWidget" widget in the left sidebar to the black rectangle in the canvas. The right sidebar shows the "Obiekt" (Object) and "Klasa" (Class) hierarchy. The "centralWidget" is highlighted, and its class is "QVTKWidget". A blue cloud bubble points to the "qvtkWidget" entry in the hierarchy, containing the text: "widget ten zastępuje w strumieniu VTK klasę okna vtkRenderWindow" (this widget replaces the vtkRenderWindow class in the VTK stream). The bottom of the interface shows the "Edytor akcji" (Action Editor) and "Edytor sygnałów / slotów" (Signal/Slot Editor) tabs.

mainwindow.ui

<Filtr>

Double...in BOX

Time Edit

Date Edit

Date/Time Edit

Dial

Horizo...ll Bar

Vertica...oll Bar

Horizo...Slider

Vertical Slider

Display Widgets

Container (KDE)

Graphics (KDE)

Input (KDE)

Buttons (KDE)

Display (KDE)

Views (KDE)

Plot (KDE)

Sonnet (KDE)

Arthur ... [Demo]

Display...amples]

Phonon

Qt 3 Support

QVTK

QVTKWidget

Wpisz tutaj

Obiekt

Klasa

MainWindow QMainWindow

centralWidget QWidget

qvtkWidget QVTKWidget

menuBar QMenuBar

mainToolBar QToolBar

statusBar QStatusBar

widget ten zastępuje w strumieniu VTK klasę okna vtkRenderWindow

objectName qvtkWidget

QVTKWidget

enabled ☒

geometry [(9, 9), 255 x 231]

sizePolicy [Preferred, Preferred]

minimumSize 0 x 0

maximumSize 16777215 x 16777215

sizeIncrement 0 x 0

baseSize 0 x 0

Nadajnik Sygnał Odbiornik Slo

Edytor akcji Edytor sygnałów / slotów

- Podłączamy renderer'a do widgetu:

```
mainwindow.cpp
1 #include "mainwindow.h"
2 #include "ui_mainwindow.h"
3
4 MainWindow::MainWindow(QWidget *parent) :
5     QMainWindow(parent),
6     ui(new Ui::MainWindow)
7 {
8     ui->setupUi(this);
9
10    vtkSmartPointer<vtkRenderer> renderer =
11        vtkSmartPointer<vtkRenderer>::New();
12
13    ui->qvtkWidget->GetRenderWindow()->AddRenderer(renderer);
14
15 }
16
17 MainWindow::~MainWindow()
18 {
19     delete ui;
20 }
21
```

Niezbędny dopisek!

```
mainwindow.h
1 #ifndef MAINWINDOW_H
2 #define MAINWINDOW_H
3
4 #include <QMainWindow>
5 #include <vtkSmartPointer.h>
6 #include <vtkRenderWindow.h>
7 #include <vtkRenderer.h>
8
9 namespace Ui {
10     class MainWindow;
11 }
12
13 class MainWindow : public QMainWindow
14 {
15     Q_OBJECT
16
17 public:
18     explicit MainWindow(QWidget *parent = 0);
19     ~MainWindow();
20
21 private:
22     Ui::MainWindow *ui;
23 };
24
25 #endif // MAINWINDOW_H
26
```

# Rozbudowa przykładu...

- Tak przygotowany projekt po kompilacji i uruchomieniu powinien wyświetlić intrygujący „czarny prostokąt” :)

Po linii 13. można umieścić dowolny kod z VTK, np. przerobiony przykład.

## ■ **Zadanie:**

- „przeszczepić” kod programu *Cone.cxx* z *Tutorial/Step1* bez pętli *for* obracającej stożkiem. Uwaga: należy pamiętać o „inkludach”!

# Przykład implementacji

```
mainwindow.cpp
1 #include "mainwindow.h"
2 #include "ui_mainwindow.h"
3
4 MainWindow::MainWindow(QWidget *parent) :
5     QMainWindow(parent),
6     ui(new Ui::MainWindow)
7 {
8     ui->setupUi(this);
9
10    // przygotowanie widgetu VTK do działania
11    vtkSmartPointer<vtkRenderer> renderer =
12        vtkSmartPointer<vtkRenderer>::New();
13
14    ui->qvtkWidget->GetRenderWindow()->AddRenderer(renderer);
15
16    // przykładowy kod VTK
17    vtkConeSource *cone = vtkConeSource::New();
18    cone->SetHeight( 3.0 );
19    cone->SetRadius( 1.0 );
20    cone->SetResolution( 10 );
21
22    vtkPolyDataMapper *coneMapper = vtkPolyDataMapper::New();
23    coneMapper->SetInputConnection( cone->GetOutputPort() );
24
25    vtkActor *coneActor = vtkActor::New();
26    coneActor->SetMapper( coneMapper );
27
28    renderer->AddActor(coneActor);
29    renderer->SetBackground(0.1, 0.2, 0.4);
30
31    // sprzątanie
32    cone->Delete();
33    coneMapper->Delete();
34    coneActor->Delete();
35 }
36
37 MainWindow::~MainWindow()
38 {
39     delete ui;
40 }
41
```

```
mainwindow.h
1 #ifndef MAINWINDOW_H
2 #define MAINWINDOW_H
3
4 #include <QMainWindow>
5 #include <vtkSmartPointer.h>
6 #include <vtkRenderWindow.h>
7 #include <vtkRenderer.h>
8
9 #include <vtkConeSource.h>
10 #include <vtkPolyDataMapper.h>
11
12 namespace Ui {
13     class MainWindow;
14 }
15
16 class MainWindow : public QMainWindow
17 {
18     Q_OBJECT
19
20 public:
21     explicit MainWindow(QWidget *parent = 0);
22     ~MainWindow();
23
24 private:
25     Ui::MainWindow *ui;
26 };
27
28 #endif // MAINWINDOW_H
29
30
```

## ZADANIE:

- dodać „suwaki” zmieniające np. liczbę ścianek stożka, wysokość
- zmienić kolor stożka (klasa `vtkProperty`)

# Komunikacja z VTK do Qt

- `vtkSmartPointer<vtkEventQtSlotConnect> connections =  
vtkSmartPointer<vtkEventQtSlotConnect>::New();`
- `// przechwycenie zdarzenia vtk:  
connections->Connect(ui->qvtkWidget->GetInteractor(),  
vtkCommand::LeftButtonPressEvent,  
this,  
SLOT(clicked()));`
  - slot Qt
  - „eventy” VTK
- `//przykład podpiecia sie pod postepy w obliczeniach,  
connections->Connect( jpgReader, vtkCommand::ProgressEvent,  
this,  
SLOT( updateProgressValue( vtkObject*,  
unsigned long, void*, void*))));`
- `//debug poprawnosci polaczen na konsole:  
connections->PrintSelf(cout, vtkIndent());`

Każdy obiekt VTK  
ma tą metodę!!!  
Przydaje się do debugowania :)



# Callbacks – wymiana informacji

[www.vtk.org/Wiki/VTK/Tutorials/Callbacks](http://www.vtk.org/Wiki/VTK/Tutorials/Callbacks)

Mechanizm oparty na „obserwatorach”, podobne do *sygnałów i slotów* ale inaczej zrealizowane!

- **vtkCommand** – obsługa zdarzeń w VTK
  - ```
void func (vtkObject*, unsigned long eid, void* clientdata,  
          void *calldata)  
{...};
```
  - ```
vtkCallbackCommand keyPressCallback();
```
  - ```
keyPressCallback->SetCallback ( func );
```
  - ```
***->AddObserver (vtkCommand::KeyPressEvent, keyPressCallback());
```