LOBACHEVSKY UNIVERSITY

Научная визуализация с помощью VTK

Васильев Евгений

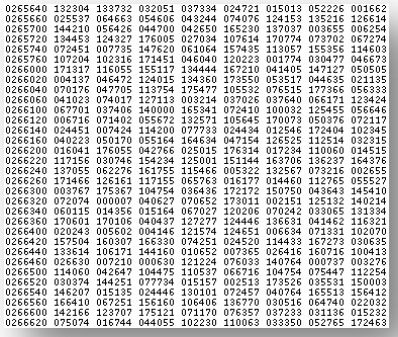
ИИТММ ННГУ

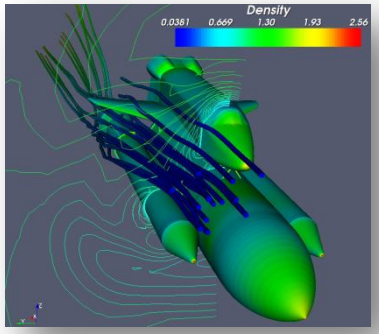
Содержание

- **Введение**
- **Обзор системы VTK**
- **Представление данных**
- **Настройка визуализации**
- **Взаимодействие с пользователем**

Введение

- Научная визуализация – это компьютерная визуализация научных данных. Исходным анализируемым данным ставится в соответствие некоторая их статическая или динамическая графическая интерпретация.





* The ParaView Tutorial

VTK

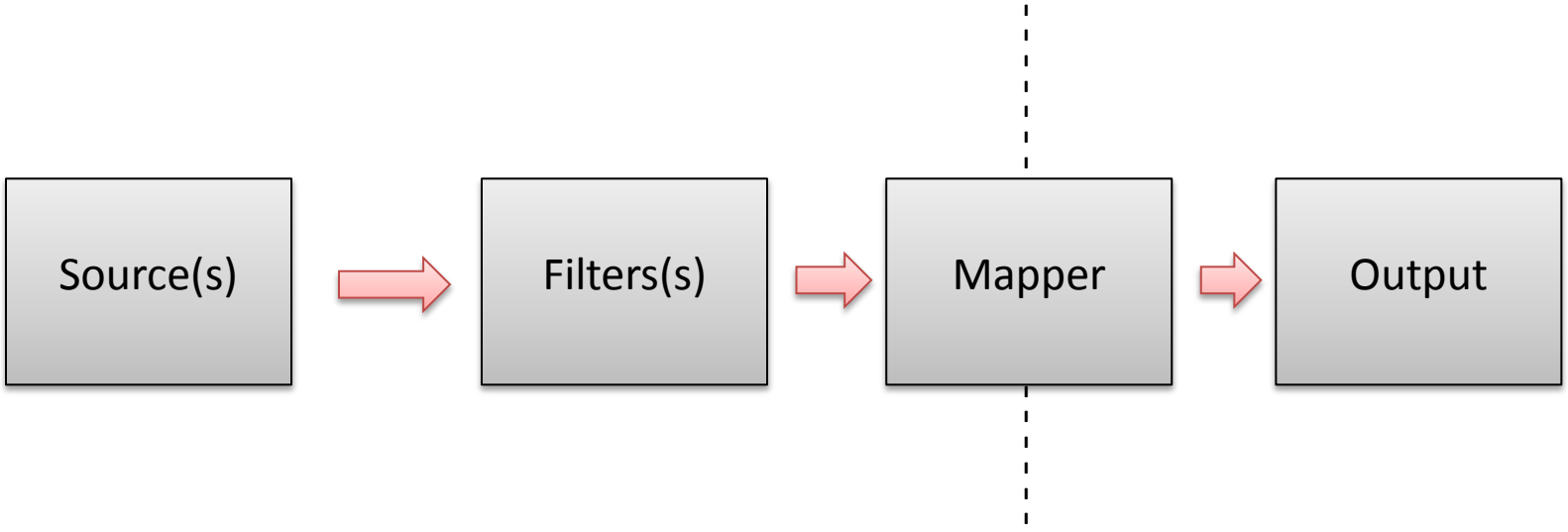
- Объектно-ориентированная библиотека для визуализации и анализа данных;
- Открытый исходный код, кросплатформенность;
- Написана на C++, интерфейсы Python, Tcl, Java;
- Актуальная документация, огромная библиотека примеров;

Обобщенный графический конвейер

сырые данные
геометрия
топология

графика

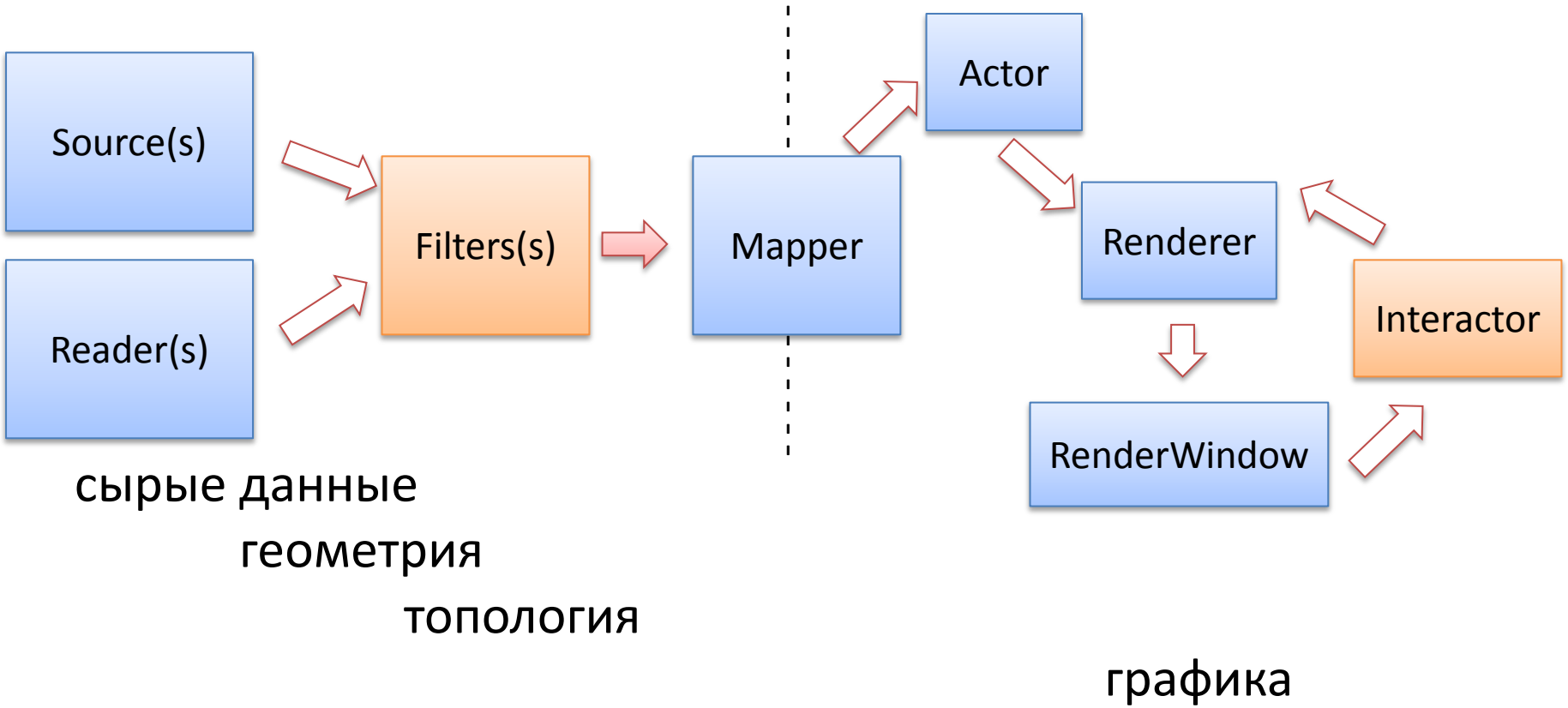
Конвейер VTK



сырые данные
геометрия
топология

графика

Конвейер VTK



Разбор программы

```
vtkSmartPointer<vtkXMLImageDataReader> reader =
    vtkSmartPointer<vtkXMLImageDataReader>::New();
reader->SetFileName(argv[1]);
reader->Update();
```

Reader

```
vtkSmartPointer<vtkContourFilter> contours =
    vtkSmartPointer<vtkContourFilter>::New();
contours->SetInputConnection(reader->GetOutputPort());
contours->GenerateValues(10, 0.0, 5.0);
```

Filter

```
vtkSmartPointer<vtkPolyDataMapper> mapper =
    vtkSmartPointer<vtkPolyDataMapper>::New();
mapper->SetInputConnection(contours->GetOutputPort());
```

Mapper

```
vtkSmartPointer<vtkActor> actor =
    vtkSmartPointer<vtkActor>::New();
actor->SetMapper(mapper);

vtkSmartPointer<vtkRenderer> renderer =
    vtkSmartPointer<vtkRenderer>::New();
renderer->AddActor(actor);

vtkSmartPointer<vtkRenderWindow> renderWindow =
    vtkSmartPointer<vtkRenderWindow>::New();
renderWindow->AddRenderer(renderer);

vtkSmartPointer<vtkRenderWindowInteractor> interactor =
    vtkSmartPointer<vtkRenderWindowInteractor>::New();
interactor->SetRenderWindow(renderWindow);

renderWindow->Render();
interactor->Start();
```

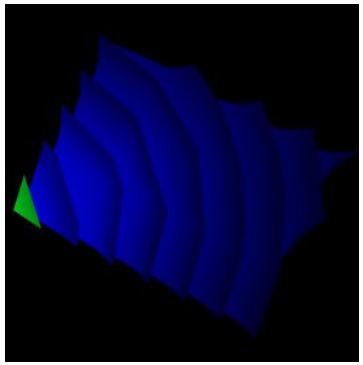
Actor

Render

Window

Interactor

Run!



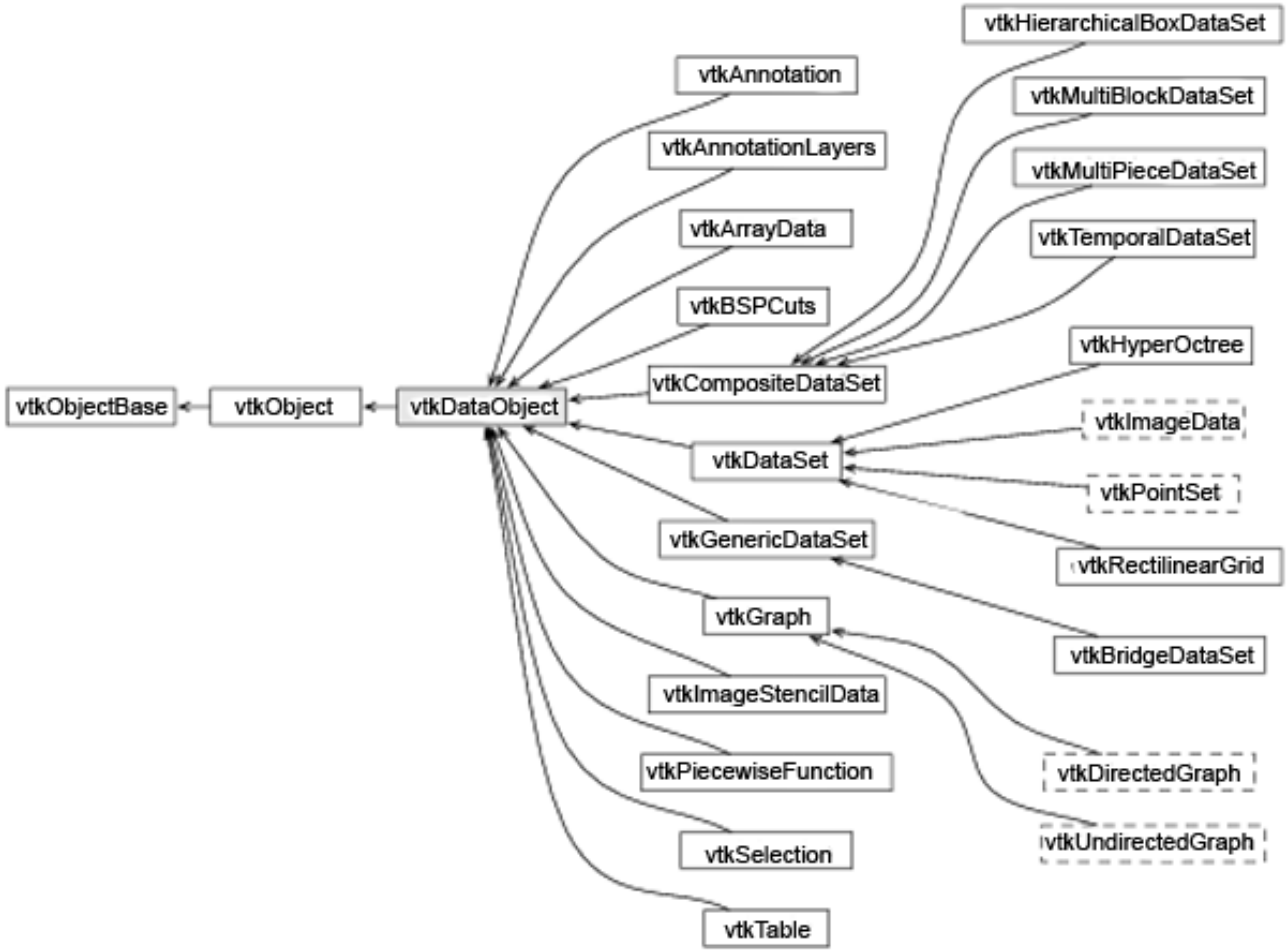
Source: [example_first.cpp]

Сборка программы в CMake

Пример CMake файла:

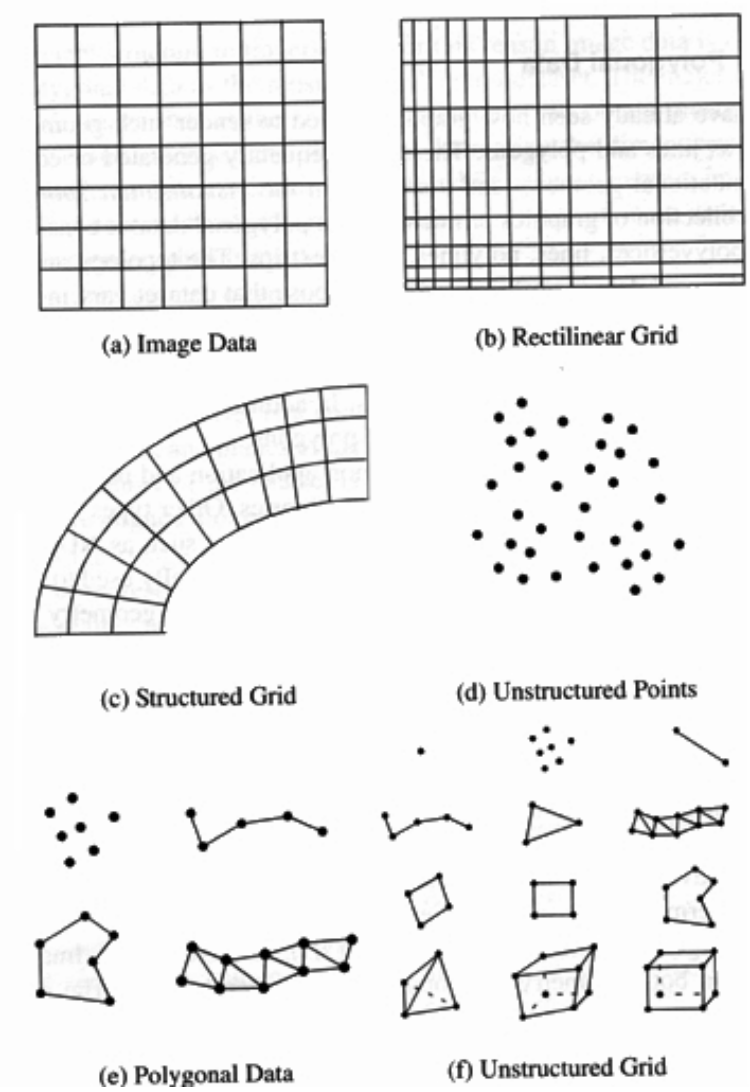
```
cmake_minimum_required(VERSION 2.8)

PROJECT(Example) # Name of the project

find_package(VTK REQUIRED) # Find the VTK library
include(${VTK_USE_FILE}) # Include the VTK library

set(SOURCE_EXE main.cpp) # Add sources

add_executable(Example ${SOURCE_EXE}) # Create executable file

target_link_libraries(Example ${VTK_LIBRARIES}) # Link VTK libraries
```

Представление данных



Структуры данных VTK

- Image Data;
- Rectilinear Grid;
- Structured Grid;
- Unstructured Points;
- Polygonal Data;
- Unstructured Grid;

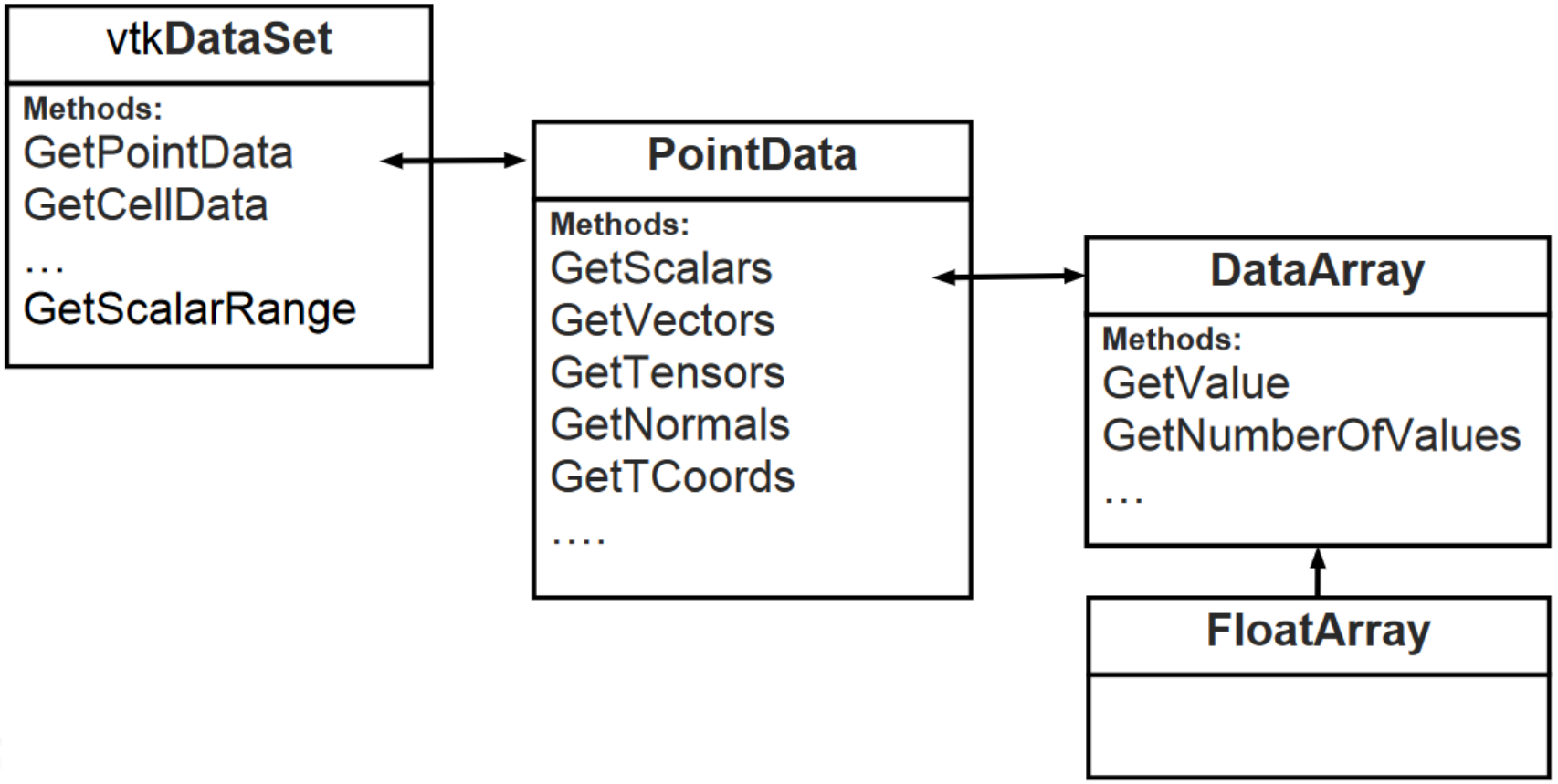


Атрибуты данных

- **Scalars**
  - Одно значение (температура, давление, плотность, высота)
- **Vectors**
  - Трехмерный вектор (направление и скорость)
- **Normals**
  - Трехмерный вектор нормали (для освещения)
- **Texture Coordinates**
  - Текстурные координаты в декартовых с/к
- **Tensors**
  - Матрица 3x3 (напряжение)

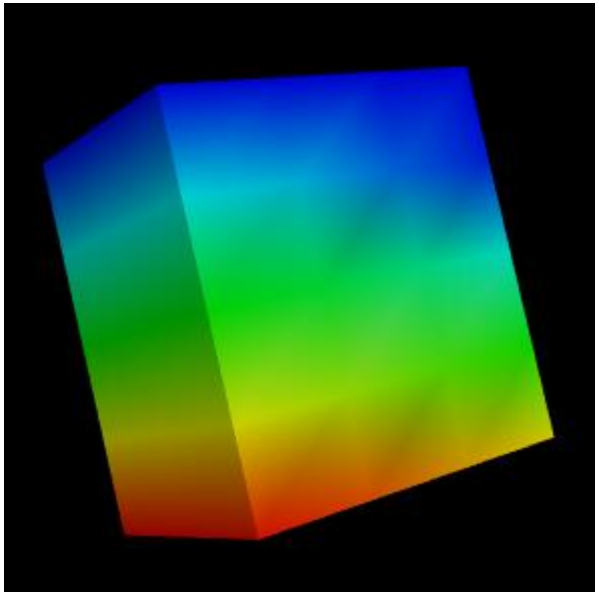
Иерархия данных

```
Data->GetPointData()->GetScalars()->GetValue(1);
```



vtkStructuredPoints

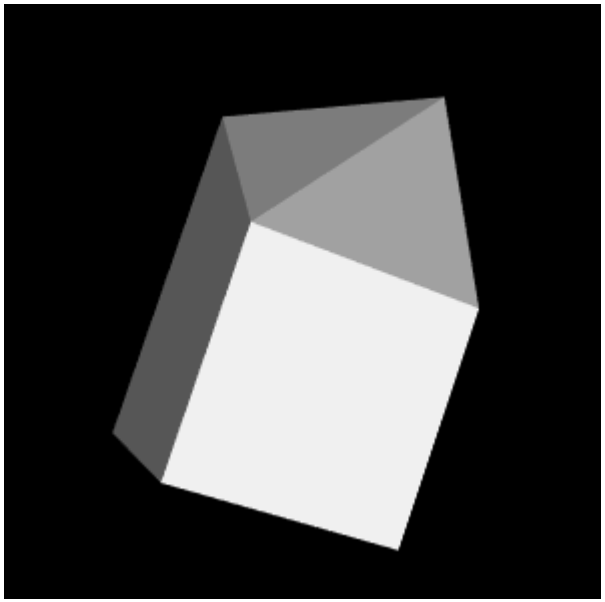
```
vtkSmartPointer<vtkStructuredPoints> SP =
vtkSmartPointer<vtkStructuredPoints>::New();
SP->SetOrigin(0, 0, 0);
SP->SetDimensions(3, 4, 5);
SP->SetSpacing(1.33, 1.25, 1);

vtkSmartPointer<vtkFloatArray> FA =
vtkSmartPointer<vtkFloatArray>::New();
for (int i = 0; i<3 * 4 * 5; i++)
FA->InsertValue(i, i * 0.02);
SP->GetPointData()->SetScalars(FA);
```



Source: example_imagedata.cpp

vtkStructuredGrid

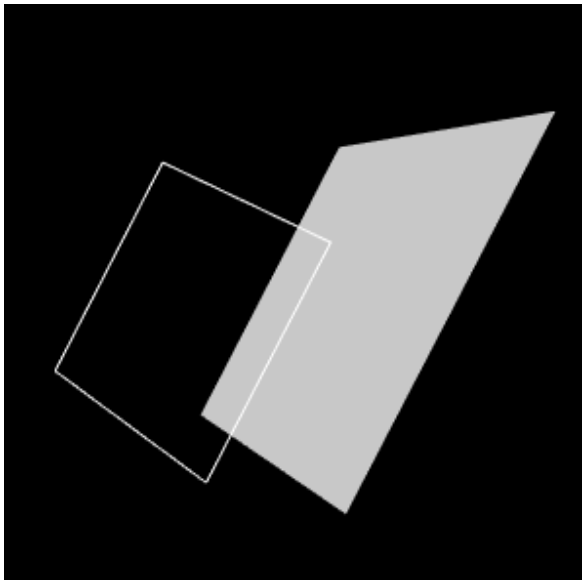
```
vtkSmartPointer<vtkPoints> P =
vtkSmartPointer<vtkPoints>::New();
P->InsertNextPoint(0, 0, 0);    P->InsertNextPoint(1, 0, 0);
P->InsertNextPoint(0, 1, 0);    P->InsertNextPoint(1, 1, 0);
P->InsertNextPoint(0, 0, 1);    P->InsertNextPoint(1, 0, 1);
P->InsertNextPoint(0, 1, 1.5); P->InsertNextPoint(1, 1, 2);

vtkSmartPointer<vtkStructuredGrid> SG =
vtkSmartPointer<vtkStructuredGrid>::New();
SG->SetDimensions(2, 2, 2);
SG->SetPoints(P);
```



Source: example_structuredgrid.cpp

vtkPolyData

```
vtkSmartPointer<vtkCellArray> CA =
vtkSmartPointer<vtkCellArray>::New();
CA->InsertNextCell(4);
CA->InsertCellPoint(3);
CA->InsertCellPoint(2);
CA->InsertCellPoint(6);
CA->InsertCellPoint(7);

vtkSmartPointer<vtkPolyData> PD =
vtkSmartPointer<vtkPolyData>::New();
PD->SetPoints(P);
PD->SetPolys(CA);
```



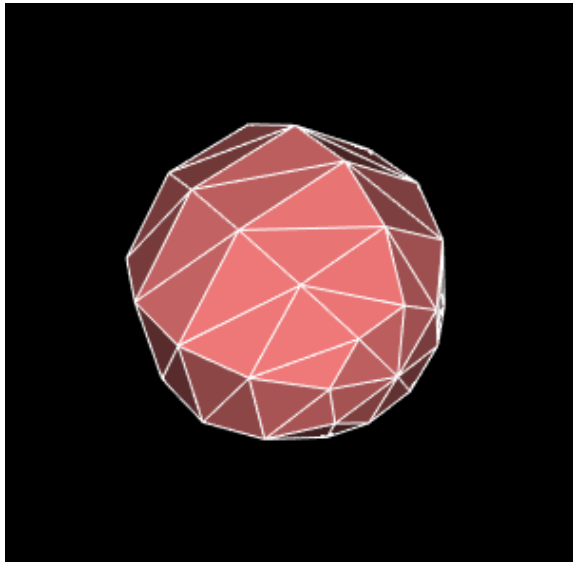
Source: [example_polydata.cpp]

Подсистема визуализации

- vtkActor – объект с визуализируемыми данными
- vtkRenderer – объект «сцены» рендеринга
- vtkCamera – камера
- vtkLight – источник света

Настройка vtkActor

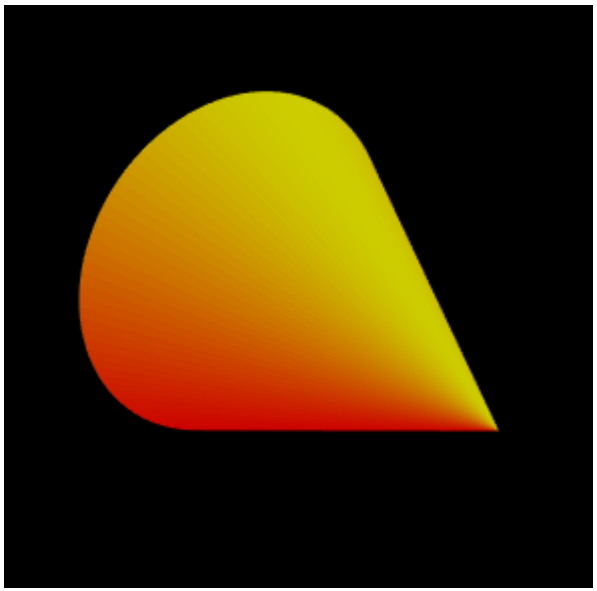
```
vtkSmartPointer<vtkActor> actor =
    vtkSmartPointer<vtkActor>::New();
actor->SetMapper(mapper);

actor->GetProperty()->SetColor(1.0, 0.5, 0.5);
actor->GetProperty()->SetInterpolationToFlat();
actor->GetProperty()->SetEdgeVisibility(1);
actor->GetProperty()->SetEdgeColor(1.0, 1.0, 1.0);
```



Source: example_actorproperties.cpp

Настройка освещения

```
vtkSmartPointer<vtkLight> light =
    vtkSmartPointer<vtkLight>::New();
light->SetColor(1.0, 1.0, 0.0);
light->SetPosition(2.0, 1.0, 0.0);

vtkSmartPointer<vtkLight> light2 =
    vtkSmartPointer<vtkLight>::New();
light2->SetColor(1.0, 0.0, 0.0);
light2->SetPosition(2.0, -1.0, 0.0);

renderer->AddLight(light);
renderer->AddLight(light2);
```



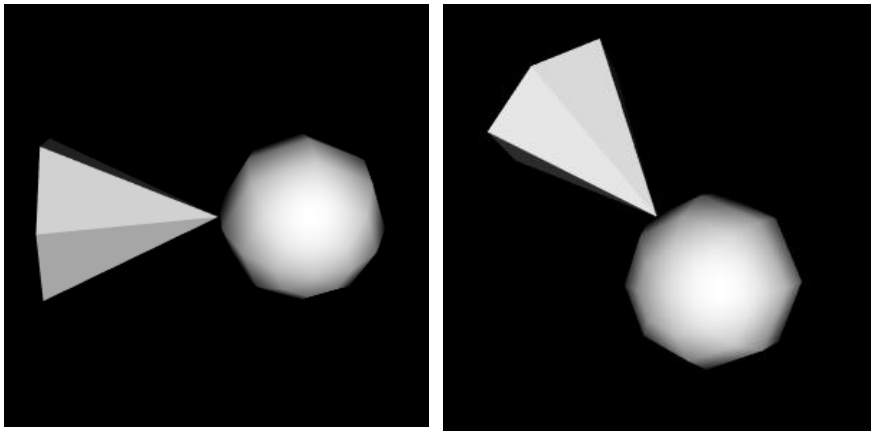
Source: [example_conelight.cpp]

Подсистема взаимодействия

- **vtkRenderWindowInteractor** – перехватывают события мыши и клавиатуры и транслируют их в манипуляции с камерой или в другие действия.
- **vtkInteractorStyle** – определяют каким образом происходят манипуляции камерой
  - **vtkInteractorStyleTrackballCamera**
  - **vtkInteractorStyleJoystickCamera**

Настройка работы с мышью

```
vtkSmartPointer<vtkInteractorStyleTrackballCamera> style =
vtkSmartPointer<vtkInteractorStyleTrackballCamera>::New();

renderWindowInteractor->SetInteractorStyle(style);
```



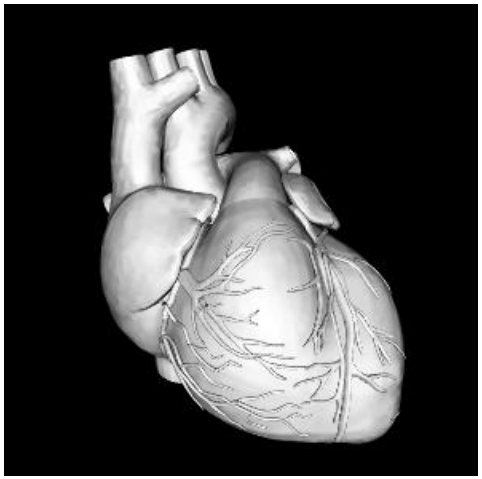
Source: [example_trackballcamera.cpp]

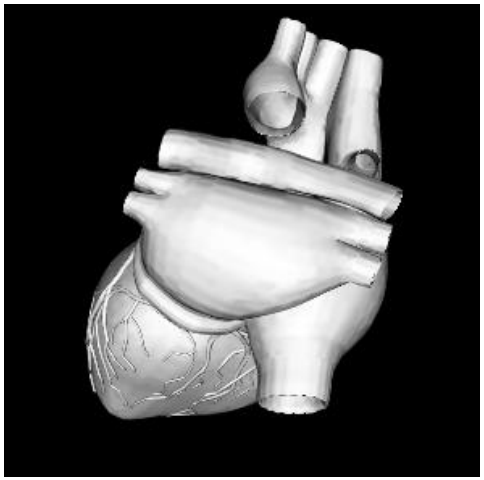
Подсистема ввода-вывода

- **Legasy formats (.vtk) –** текстовые форматы данных, удобно читать, загружаются долго
- **XML formats (.vti, .vtu, etc) –** XML-подобные форматы, неудобно читать, загружаются быстрее
- Подробное описание форматов: www.vtk.org/VTK/img/file-formats.pdf

Чтение obj файла

```
vtkSmartPointer<vtkOBJReader> reader =
vtkSmartPointer<vtkOBJReader>::New();
reader->SetFileName(argv[1]);
reader->Update();
```

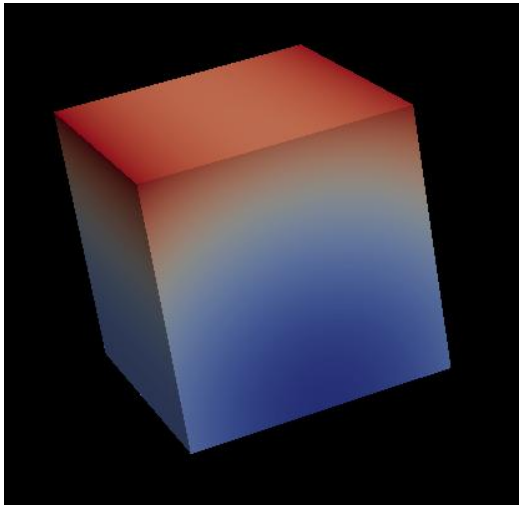




Source: [example_objreader.cpp]

Сохранение массива в файл

```
// Legacy format
vtkSmartPointer<vtkStructuredPointsWriter> writer =
vtkSmartPointer<vtkStructuredPointsWriter>::New();
writer->SetFileName("3darray.vtk”);
writer->SetInputData(dataImage);
writer->Write();

// XML format
vtkSmartPointer<vtkXMLImageDataWriter> writerXml =
vtkSmartPointer<vtkXMLImageDataWriter>::New();
writerXml->SetFileName ("3darray.vti");
writerXml->SetInputData(dataImage);
writerXml->Write();
```



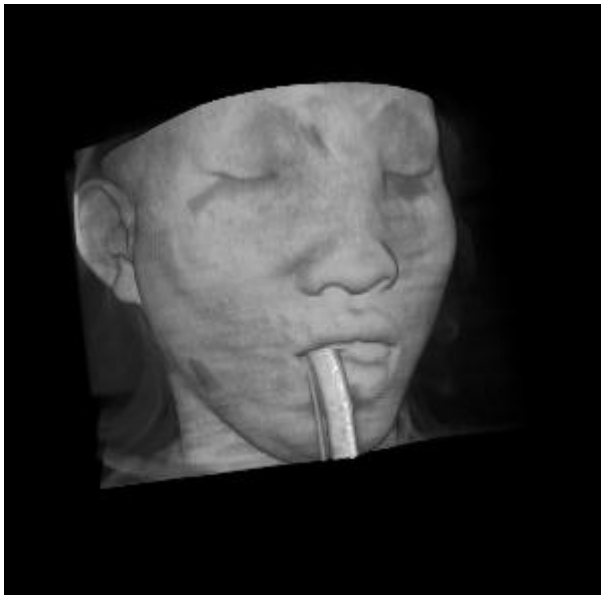
Source: [example_save3darray.cpp]

Расширенные примеры

- Объемный рендеринг;

Объемный рендеринг - 1

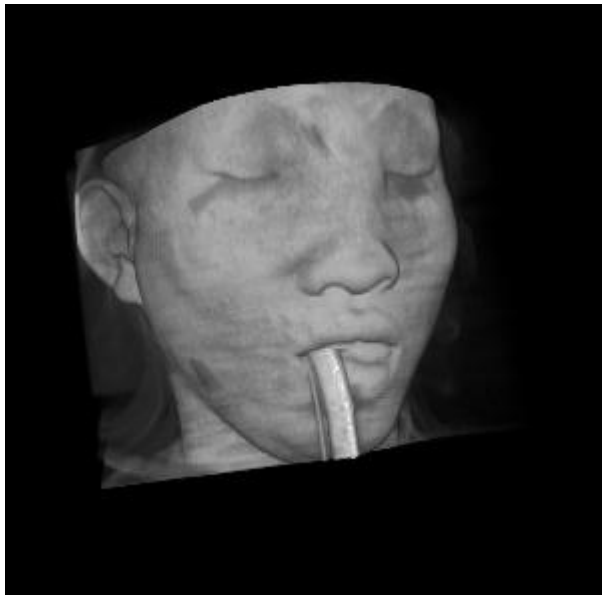
```
// Volume mapping
vtkSmartPointer<vtkSmartVolumeMapper> volumeMapper =
    vtkSmartPointer<vtkSmartVolumeMapper>::New();
volumeMapper->SetInputConnection(reader->GetOutputPort());
volumeMapper->SetRequestedRenderModeToGPU();

vtkSmartPointer<vtkVolumeProperty> volumeProperty =
    vtkSmartPointer<vtkVolumeProperty>::New();
volumeProperty->ShadeOff();
volumeProperty->
    SetInterpolationType(VTK_CUBIC_INTERPOLATION);
```



Объемный рендеринг - 2

```
// Transfer function
vtkSmartPointer<vtkPiecewiseFunction> tfOpacity =
    vtkSmartPointer<vtkPiecewiseFunction>::New();
tfOpacity->AddPoint(0.0, 0.0);
tfOpacity->AddPoint(100.0, 0.0);
tfOpacity->AddPoint(1000.0, 1.0);
volumeProperty->SetScalarOpacity(tfOpacity);

vtkSmartPointer<vtkColorTransferFunction> tfColor =
    vtkSmartPointer<vtkColorTransferFunction>::New();
tfColor->AddRGBPoint(0.0, 0.0, 0.0, 0.0);
tfColor->AddRGBPoint(1000.0, 1.0, 1.0, 1.0);
volumeProperty->SetColor(tfColor);
```

Source : example_volumerender.cpp



Литература

- Система VTK. Глава книги "Архитектура приложений с открытым исходным кодом". Ссылка (на русском)
- The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics. Ссылка(English)
- Официальный сайт www.vtk.org/
- Примеры на GitHub