Бесплатная электронная книга

учусь

vtk

Free unaffiliated eBook created from
Stack Overflow contributors.

#vtk

Около

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: vtk

It is an unofficial and free vtk ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official vtk.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

глава 1: Начало работы с vtk

замечания

В этом разделе представлен обзор того, что такое vtk, и почему разработчик может захотеть его использовать.

Следует также упомянуть любые крупные темы в vtk и ссылки на связанные темы. Поскольку Documentation for vtk является новым, вам может потребоваться создать начальные версии этих связанных тем.

Examples

Установка или настройка

Строительство и установка на Windows 7

Предпосылки

- Если вы хотите построить VTK из последних источников, вам нужно git from Here или вы можете загрузить моментальный снимок кода в виде zip и разархивировать на свой диск
- CMake
- Microsoft Visual Studio 2015
- Много свободного места - по крайней мере, пара GB должна быть в безопасности, действительно в зависимости от того, что вы хотите построить

Готовиться

- Мне нравится держать вещи в чистоте, поэтому я обычно создаю 3 папки:

```
c:\vtk              #
c:\vtk\src          # 'code base' folder
c:\vtk\build        # 'out of source' build folder
c:\vtk\install      # 'install folder' where the 'installed' files will reside
```

- Если вы используете метод git,
    - откройте командную строку
    - изменить рабочий каталог `cd c:\vtk\src`
    - клонировать git-репозиторий `git clone https://gitlab.kitware.com/vtk/vtk.git` . Это

может занять некоторое время в зависимости от скорости вашего интернет-соединения
    - Если вы работаете за прокси-сервером, вам нужно настроить git, чтобы использовать его. См. Этот вопрос о том, как это сделать.

- Если вы используете метод zip, распакуйте исходный код в `c:\vtk\src`

конфигурация

- Запустить графический интерфейс CMake
- Выберите `c:\vtk\src` для `Where is the source code:`
- Выберите `c:\vtk\build` для `Where to build the binaries:`
- Хит `Configure` и выбрать `Visual Studio 2015` как необходимый генератор
- Вам будет предложено несколько вариантов конфигурации
- Обычно я использую следующие настройки для минимальной сборки
    - `CMAKE_INSTALL_PREFIX` = `c:\vtk\install`
    - `BUILD_SHARED_LIBS` галочкой
    - `BUILD_DOCUMENTATION`
    - `BUILD_TESTING`
    - `CMAKE_CXX_MP_FLAG` . Это будет использовать все ядра ЦП (в многоядерных / многопроцессорных системах), чтобы ускорить сборку
- Удерживание удара `Configure` корректировки любых ошибок до тех пор, пока все КРАСНЫЕ записи не станут БЕЛАЯ
- Hit `Generate`
- Закрыть CMake GUI

Строительство

- Если генерация прошла успешно,
    - Решение Visual Studio:

      ```
      c:\vtk\build\vtk.sln
      ```

    - Куча файлов проекта -

      ```
      ALL_BUILD.vcxproj
      INSTALL.vcxproj
      vtkCompileTools.vcxproj
      VTKData.vcxproj
      ZERO_CHECK.vcxproj
      ```

- Вы можете построить это, используя либо из командной строки, либо используя IDE
- Я предпочитаю командную строку, поскольку она, как правило, быстрее и использует меньше ОЗУ
- Использование командной строки
    - Запустить `Developer Command Prompt For Visual Studio 2015`

        - Изменить рабочий каталог: `cd c:\vtk\build`
        - Запустить msbuild:
            - для отладочных сборников
                - `msbuild /p:Configuration=Debug ALL_BUILD.vcxproj`
                - `msbuild /p:Configuration=Debug INSTALL.vcxproj`
            - для выпуска
                - `msbuild /p:Configuration=Release ALL_BUILD.vcxproj`
                - `msbuild /p:Configuration=Release INSTALL.vcxproj`
- Использование среды IDE
    - Откройте `VTK.sln` с помощью Visual Studio 2015 и создайте `INSTALL.vcxproj`
    - Этот метод обычно медленнее, так как IDE начнет строить intellisense для каждого из проектов, перечисленных в решении
- `c:\vtk\install` теперь должны иметь несколько новых папок
    - `bin` # содержит файлы DLL
    - `lib` # содержит файлы lib
    - `cmake`
    - `share`
    - `include` # содержит файлы заголовков

Использование сборки

- Чтобы использовать VTK в проекте Visual C ++, нужно
    - Настройте путь поиска файла заголовка компилятора, чтобы включить `c:\vtk\include\vtk-<version>`
    - Настройте путь поиска файла библиотеки компоновщика, чтобы включить `c:\vtk\lib`
    - Настройте компоновщик для ссылки на требуемые `.lib` файлы
    - Скопируйте необходимые библиотеки DLL в папку вывода
- Я собрал небольшой файл реквизита для обработки всех четырех задач `c:\vtk\vtk.vsprops`

```
<?xml version="1.0" encoding="UTF-8"?>
<Project ToolsVersion="4.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
    <PropertyGroup>
        <VTK_ROOT_DIR>$(MSBuildThisFileDirectory)</VTK_ROOT_DIR>
        <VTK_BIN_DIR>$(VTK_ROOT_DIR)\bin</VTK_BIN_DIR>
        <VTK_INC_DIR>$(VTK_ROOT_DIR)\include\vtk-7.0</VTK_INC_DIR>
        <VTK_LIB_DIR>$(VTK_ROOT_DIR)\lib</VTK_LIB_DIR>
    </PropertyGroup>

    <PropertyGroup>
        <BuildDependsOn>CopyVTKBinariesList;$(BuildDependsOn);</BuildDependsOn>
    </PropertyGroup>

    <Target Name="CopyVTKBinariesList">
        <ItemGroup>
            <VtkBinaries Include="$(VTK_BIN_DIR)\*.dll" />
        </ItemGroup>
        <Copy SourceFiles="@(VtkBinaries)"
              DestinationFiles="@(VtkBinaries-
>'$(OutDir)\%(RecursiveDir)%(Filename)%(Extension)')"
```

```
              SkipUnchangedFiles="true" />
    </Target>

    <ItemDefinitionGroup>
      <ClCompile>

<AdditionalIncludeDirectories>$(VTK_INC_DIR);%(AdditionalIncludeDirectories)</AdditionalIncludeDirecto

      </ClCompile>
      <Link>

<AdditionalLibraryDirectories>$(VTK_LIB_DIR);%(AdditionalLibraryDirectories)</AdditionalLibraryDirecto

        <AdditionalDependencies>vtkalglib-7.0.lib;vtkChartsCore-7.0.lib;vtkCommonColor-
7.0.lib;vtkCommonComputationalGeometry-7.0.lib;vtkCommonCore-7.0.lib;vtkCommonDataModel-
7.0.lib;vtkCommonExecutionModel-7.0.lib;vtkCommonMath-7.0.lib;vtkCommonMisc-
7.0.lib;vtkCommonSystem-7.0.lib;vtkCommonTransforms-7.0.lib;vtkDICOMParser-
7.0.lib;vtkDomainsChemistry-7.0.lib;vtkDomainsChemistryOpenGL2-7.0.lib;vtkexoIIc-
7.0.lib;vtkexpat-7.0.lib;vtkFiltersAMR-7.0.lib;vtkFiltersCore-7.0.lib;vtkFiltersExtraction-
7.0.lib;vtkFiltersFlowPaths-7.0.lib;vtkFiltersGeneral-7.0.lib;vtkFiltersGeneric-
7.0.lib;vtkFiltersGeometry-7.0.lib;vtkFiltersHybrid-7.0.lib;vtkFiltersHyperTree-
7.0.lib;vtkFiltersImaging-7.0.lib;vtkFiltersModeling-7.0.lib;vtkFiltersParallel-
7.0.lib;vtkFiltersParallelImaging-7.0.lib;vtkFiltersProgrammable-7.0.lib;vtkFiltersSelection-
7.0.lib;vtkFiltersSMP-7.0.lib;vtkFiltersSources-7.0.lib;vtkFiltersStatistics-
7.0.lib;vtkFiltersTexture-7.0.lib;vtkFiltersVerdict-7.0.lib;vtkfreetype-7.0.lib;vtkGeovisCore-
7.0.lib;vtkglew-7.0.lib;vtkhdf5-7.0.lib;vtkhdf5_hl-7.0.lib;vtkImagingColor-
7.0.lib;vtkImagingCore-7.0.lib;vtkImagingFourier-7.0.lib;vtkImagingGeneral-
7.0.lib;vtkImagingHybrid-7.0.lib;vtkImagingMath-7.0.lib;vtkImagingMorphological-
7.0.lib;vtkImagingSources-7.0.lib;vtkImagingStatistics-7.0.lib;vtkImagingStencil-
7.0.lib;vtkInfovisCore-7.0.lib;vtkInfovisLayout-7.0.lib;vtkInteractionImage-
7.0.lib;vtkInteractionStyle-7.0.lib;vtkInteractionWidgets-7.0.lib;vtkIOAMR-7.0.lib;vtkIOCore-
7.0.lib;vtkIOEnSight-7.0.lib;vtkIOExodus-7.0.lib;vtkIOExport-7.0.lib;vtkIOGeometry-
7.0.lib;vtkIOImage-7.0.lib;vtkIOImport-7.0.lib;vtkIOInfovis-7.0.lib;vtkIOLegacy-
7.0.lib;vtkIOLSDyna-7.0.lib;vtkIOMINC-7.0.lib;vtkIOMovie-7.0.lib;vtkIONetCDF-
7.0.lib;vtkIOParallel-7.0.lib;vtkIOParallelXML-7.0.lib;vtkIOPLY-7.0.lib;vtkIOSQL-
7.0.lib;vtkIOVideo-7.0.lib;vtkIOXML-7.0.lib;vtkIOXMLParser-7.0.lib;vtkjpeg-7.0.lib;vtkjsoncpp-
7.0.lib;vtklibxml2-7.0.lib;vtkmetaio-7.0.lib;vtkNetCDF-7.0.lib;vtkNetCDF_cxx-
7.0.lib;vtkoggtheora-7.0.lib;vtkParallelCore-7.0.lib;vtkpng-7.0.lib;vtkproj4-
7.0.lib;vtkRenderingAnnotation-7.0.lib;vtkRenderingContext2D-
7.0.lib;vtkRenderingContextOpenGL2-7.0.lib;vtkRenderingCore-7.0.lib;vtkRenderingFreeType-
7.0.lib;vtkRenderingImage-7.0.lib;vtkRenderingLabel-7.0.lib;vtkRenderingLOD-
7.0.lib;vtkRenderingOpenGL2-7.0.lib;vtkRenderingVolume-7.0.lib;vtkRenderingVolumeOpenGL2-
7.0.lib;vtksqlite-7.0.lib;vtksys-7.0.lib;vtktiff-7.0.lib;vtkverdict-7.0.lib;vtkViewsContext2D-
7.0.lib;vtkViewsCore-7.0.lib;vtkViewsGeovis-7.0.lib;vtkViewsInfovis-7.0.lib;vtkzlib-
7.0.lib;%(AdditionalDependencies)</AdditionalDependencies>
      </Link>
    </ItemDefinitionGroup>
    <ItemGroup />

</Project>
```

- Вышеупомянутый файл vsprops копирует все доступные DLL в папку `c:\vtk\bin` .
- Альтернативный способ убедиться, что DLL могут быть расположены, - это использовать переменную среды PATH для сеанса отладки и поместить путь двоичных файлов VTK в качестве первого каталога для поиска при загрузке зависимостей. Следующий фрагмент может быть вместо задачи `CopyVTKBinariesList` для этого.

```
<PropertyGroup>

<LocalDebuggerEnvironment>PATH=$(VTK_BIN_DIR);%PATH%;$(LocalDebuggerEnvironment)</LocalDebuggerEn

</PropertyGroup>
```

- Для окончательного развертывания вы можете использовать инструмент, например Dependency Walker, для отслеживания того, какие DLL и их зависимости используются, и только для пакетов только для перераспределения.
- Чтобы использовать файл реквизита в проекте Visual C ++, вы можете использовать инструмент Property Manager в Visual Studio (Menu: View => Property Manager) или отредактировать vcxproj с помощью текстового редактора и добавить следующую строку `<Import Project="C:\vtk\vtk.vsprops" />` ниже других импортных проектов.

Убираться

- Если вы хотите восстановить некоторое дисковое пространство, вы можете удалить папку `c:\vtk\build` но недостатком является то, что вы не можете отлаживать vtk

Пробные

- Просто удалите папку `c:\vtk` если вы больше не хотите VTK

MacOSX и Unix:

1. Установите последнюю версию CMake, доступную здесь.
2. Загрузите последнюю версию VTK здесь .
3. Создать каталог сборки для VTK `mkdir <path_to_build_directory`
4. Настроить с помощью `ccmake <path_to_VTK_directory -G "UNIX Makefiles" \ -DVTK_USE_QVTK:BOOL=ON \ -DVTK_USE_CARBON:BOOL=ON \ -DCMAKE_INSTALL_PREFIX=/usr/local \ -DVTK_USE_GUISUPPORT:BOOL=ON` или использовать графический интерфейс для этого с помощью `ccmake <path_to_VTK_directory`
5. Войдите в каталог сборки и используйте `make -j` (вам не нужно использовать `-j` но компиляция действительно длинная.
6. Наконец, используйте `make install`

Прочитайте Начало работы с vtk онлайн: https://riptutorial.com/ru/vtk/topic/5182/начало-работы-с-vtk

глава 2: Привет, мир

Examples

Привет, мир

```
#include <vtkAutoInit.h>

VTK_MODULE_INIT(vtkRenderingOpenGL2);
VTK_MODULE_INIT(vtkRenderingFreeType);
VTK_MODULE_INIT(vtkInteractionStyle);

#include <vtkSmartPointer.h>
#include <vtkTextActor.h>
#include <vtkRenderer.h>
#include <vtkRenderWindow.h>
#include <vtkRenderWindowInteractor.h>

int main(int /*argc*/, char ** /*argv*/)
{
    auto textActor = vtkSmartPointer<vtkTextActor>::New();
    textActor->SetInput("Hello World");

    auto renderer = vtkSmartPointer<vtkRenderer>::New();
    renderer->AddActor(textActor);
    renderer->ResetCamera();

    auto interactor = vtkSmartPointer<vtkRenderWindowInteractor>::New();

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

    interactor->Start();

    return 0;
}
```

Сломать:

```
#include <vtkAutoInit.h>

VTK_MODULE_INIT(vtkRenderingOpenGL2);
VTK_MODULE_INIT(vtkRenderingFreeType);
VTK_MODULE_INIT(vtkInteractionStyle);
```

Конструкция VTK использует шаблон проектирования фабричного метода для создания новых экземпляров производных классов vtkObject с использованием метода `<ClassName>::New()` . Это позволяет выбрать конкретную реализацию платформы во время выполнения, чтобы удовлетворить требуемый интерфейс.

Чтобы этот механизм работал, фабричные классы должны «регистрироваться» самостоятельно, чтобы их можно было выбрать инфраструктурой vtk. Подробности по этой теме доступны здесь .

`VTK_MODULE_INIT` - это макрос, используемый для автоматического инициализации необходимых модулей / библиотек (`vtkRenderingOpenGL2`) (`vtkRenderingOpenGL2` , `vtkRenderingFreeType` , `vtkInteractionStyle` в этом примере). Невозможность инициализации модулей приведет к `<ClassName>::New()` для возврата `NULL` и, следовательно, ошибок времени выполнения.

```
#include <vtkSmartPointer.h>
#include <vtkTextActor.h>
#include <vtkRenderer.h>
#include <vtkRenderWindow.h>
#include <vtkRenderWindowInteractor.h>
```

Роль `vtkSmartPointer` аналогична роли `std::unique_ptr` в том, что она управляет счетчиком ссылок, который управляет временем жизни `vtkObject` производного класса `vtkObject` .

`vtkTextActor` - это простой класс, который можно использовать для отображения строк на экране.

`vtkRenderer` - это класс, ответственный за управление содержимым сцены. В частности, он управляет сбором

- 2D-актеры, полученные из `vtkActor2D`
- 3D-актеры, полученные из `vtkProp3D`
- Объемы: `vtkVolume`
- Камера: `vtkCamera`
- Свет: `vtkLight`

`vtkRenderWindow` - это класс, который обеспечивает платформенный независимый интерфейс для

- управление коллекцией рендеринга.
- обрабатывая ввод пользователя и перенаправляя его на `vtkRenderWindowInteractor` для дальнейшей обработки

`vtkRenderWindowInteractor` - это класс, отвечающий за сопоставление событий ввода (мыши / клавиатуры / времени) пользователя с соответствующим действием. Внутри он использует `vtkInteractorStyle` для обеспечения различных `vtkInteractorStyle` отображения.

```
auto textActor = vtkSmartPointer<vtkTextActor>::New();
textActor->SetInput("Hello World");
```

Создайте текстовый актер и установите строку для отображения

```
auto renderer = vtkSmartPointer<vtkRenderer>::New();
renderer->AddActor(textActor);
renderer->ResetCamera();
```

- Создание рендеринга
- Добавьте к нему текстового актера
- Сбрасывает положение камеры, чтобы убедиться, что актер отображается на экране.

```
auto interactor = vtkSmartPointer<vtkRenderWindowInteractor>::New();
```

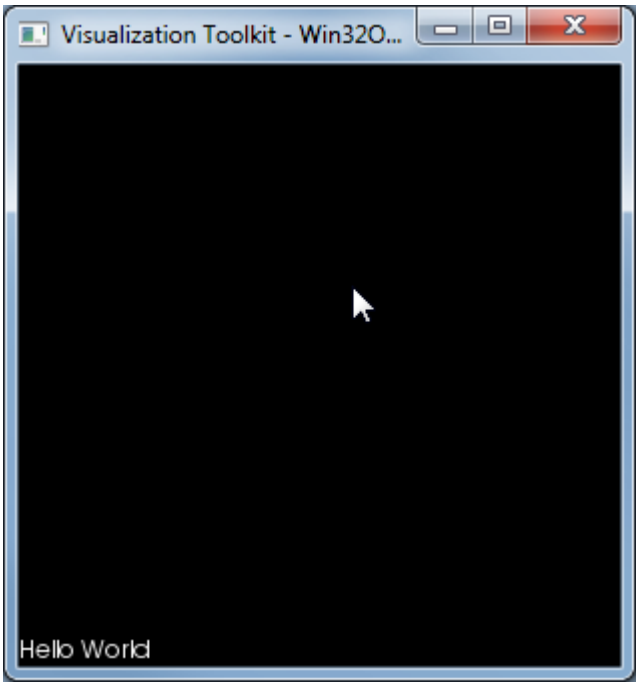
```
auto renderWindow = vtkSmartPointer<vtkRenderWindow>::New();
renderWindow->AddRenderer(renderer);
renderWindow->SetInteractor(interactor);
```

Создайте окно для рендеринга, добавьте рендер к нему и установите интерактор. Заводская функция автоматически выбирает подходящую реализацию на основе доступных / зарегистрированных заводских классов

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

Это блокирующий вызов, который возвращается только тогда, когда пользователь запрашивает quit (ключ `q`) или закрывает окно. Запускает цикл сообщений и отправляет сообщения.

Запуск этого должен создать окно, которое выглядит так:



Заметки

Этот список DLL, который использовался этим exe:

```
VTKCommonCore-7.0.DLL

VTKInteractionStyle-7.0.DLL

VTKRenderingCore-7.0.DLL

VTKRenderingFreeType-7.0.DLL

VTKRenderingOpenGL2-7.0.DLL
```

Прочитайте Привет, мир онлайн: https://riptutorial.com/ru/vtk/topic/5974/привет--мир

кредиты

S. No	Главы	Contributors
1	Начало работы с vtk	Community, LBes, Shreyas Murali
2	Привет, мир	Shreyas Murali