EBook Gratis

APRENDIZAJE vtk

Free unaffiliated eBook created from
Stack Overflow contributors.

#vtk

Tabla de contenido

Acerca de

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

Capítulo 1: Empezando con vtk

Observaciones

Esta sección proporciona una descripción general de qué es vtk y por qué un desarrollador puede querer usarlo.

También debe mencionar cualquier tema grande dentro de vtk y vincular a los temas relacionados. Dado que la Documentación para vtk es nueva, es posible que deba crear versiones iniciales de esos temas relacionados.

Examples

Instalación o configuración

Construcción e instalación en Windows 7

Prerrequisitos

- Si desea compilar VTK a partir de las fuentes más recientes, necesita git desde Aquí o puede descargar una instantánea del código como un zip y descomprimirlo en su unidad de disco
- CMake
- Microsoft Visual Studio 2015
- Un montón de espacio libre - al menos un par de GB para estar seguro, dependiendo realmente de lo que quieras construir

Preparandose

- Me gusta mantener las cosas limpias, así que normalmente creo 3 carpetas así:

```
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
```

- Si usa el método git,
    - abrir un indicador de comando
    - cambiar el directorio de trabajo `cd c:\vtk\src`
    - clonar el repositorio `git clone https://gitlab.kitware.com/vtk/vtk.git` . Esto podría tomar un tiempo dependiendo de la velocidad de su conexión a internet
    - Si está trabajando detrás de un proxy, necesitará configurar git para usarlo. Vea esta

pregunta sobre cómo hacer eso.

- Si usa el método zip, descomprima el código fuente en `c:\vtk\src`

Configuración

- Iniciar la interfaz gráfica de usuario de CMake
- Seleccione `c:\vtk\src` para `Where is the source code:`
- Seleccione `c:\vtk\build` para `Where to build the binaries:`
- Presiona `Configure` y selecciona `Visual Studio 2015` como el generador requerido
- Se le presentará una serie de opciones de configuración
- Generalmente uso la siguiente configuración para una construcción mínima
  - `CMAKE_INSTALL_PREFIX` = `c:\vtk\install`
  - `BUILD_SHARED_LIBS` marcado
  - `BUILD_DOCUMENTATION` marcar
  - `BUILD_TESTING` marcar
  - `CMAKE_CXX_MP_FLAG` marcado. Esto utilizará todos los núcleos de la CPU (en sistemas multinúcleo / multiprocesador) para acelerar la compilación
- Mantener pulsando `Configure` corregir cualquier error hasta que todas las entradas ROJAS se vuelvan BLANCAS
- Hit `Generate`
- Cerrar CMake GUI

edificio

- Si la generación fue exitosa debería haber
  - Una solución de Visual Studio:

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

  - Un montón de archivos de proyecto -

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

- Puede compilar esto usando una línea de comandos o usando el IDE
- Prefiero la línea de comandos ya que generalmente es más rápida y usa menos RAM
- Usando la línea de comando
  - Iniciar la `Developer Command Prompt For Visual Studio 2015`
  - Cambiar directorio de trabajo: `cd c:\vtk\build`
  - Ejecutar msbuild:
    - para construcciones de depuración
      - `msbuild /p:Configuration=Debug ALL_BUILD.vcxproj`
      - `msbuild /p:Configuration=Debug INSTALL.vcxproj`
    - para compilaciones de lanzamiento

        - `msbuild /p:Configuration=Release ALL_BUILD.vcxproj`
        - `msbuild /p:Configuration=Release INSTALL.vcxproj`
- Usando el IDE
    - Abra `VTK.sln` con Visual Studio 2015 y `VTK.sln` el `INSTALL.vcxproj`
    - Esta técnica suele ser más lenta, ya que el IDE comenzará a desarrollar una inteligencia para cada uno de los proyectos enumerados en la solución.
- `c:\vtk\install` ahora debería tener algunas carpetas nuevas
    - `bin` # contiene los archivos dll
    - `lib` # contiene los archivos lib
    - `cmake`
    - `share`
    - `include` # contiene los archivos de encabezado

Usando la compilación

- Para usar VTK en un proyecto de Visual C ++, uno tiene que
    - Configure la ruta de búsqueda del archivo del encabezado del compilador para incluir `c:\vtk\include\vtk-<version>`
    - Configure la ruta de búsqueda del archivo de la biblioteca del vinculador para incluir `c:\vtk\lib`
    - Configure el enlazador para vincular a los archivos `.lib` requeridos
    - Copie las DLL requeridas a la carpeta de salida
- He reunido un pequeño archivo de `c:\vtk\vtk.vsprops` para manejar las cuatro tareas `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)</AdditionalLibraryDirector

        <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>
```

- El archivo vsprops anterior copia todas las dll disponibles en la carpeta `c:\vtk\bin` .
- Una forma alternativa de asegurarse de que se puedan ubicar las DLL es usar la variable de entorno PATH para la sesión de depuración y colocar la ruta de los binarios VTK como el primer directorio que se buscará al cargar las dependencias. El siguiente fragmento puede ser en lugar de la tarea `CopyVTKBinariesList` para hacer esto.

  ```
  <PropertyGroup>

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

  </PropertyGroup>
  ```

- Para la implementación final, es posible que desee utilizar una herramienta como Dependency Walker para rastrear qué archivos DLL y sus dependencias se utilizan y agrupar solo aquellos para redistribuir.

- Para usar el archivo de propiedades en un proyecto de Visual C ++, puede usar la herramienta Administrador de propiedades dentro de Visual Studio (Menú: Ver => Administrador de propiedades) o editar el vcxproj usando un editor de texto y agregar la siguiente línea `<Import Project="C:\vtk\vtk.vsprops" />` debajo de las otras importaciones del proyecto.

Limpiar

- Si desea recuperar algo de espacio en el disco, puede eliminar la carpeta `c:\vtk\build` pero la desventaja es que no puede depurar en vtk

Desinstalación

- Simplemente elimine la carpeta `c:\vtk` si ya no desea VTK

MacOSX y Unix:

1. Instala la última versión de CMake disponible aquí
2. Descarga el último VTK aquí .
3. Cree un directorio de compilación para VTK `mkdir <path_to_build_directory`
4. Configure con `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 ccmake <path_to_VTK_directory`
5. Ingrese al directorio de compilación y use `make -j` (no tiene que usar `-j` pero la compilación es realmente larga.
6. Finalmente usar `make install`

Lea Empezando con vtk en línea: https://riptutorial.com/es/vtk/topic/5182/empezando-con-vtk

Capítulo 2: Hola Mundo

Examples

Hola mundo ejemplo

```
#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;
}
```

Descompostura:

```
#include <vtkAutoInit.h>

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

El diseño de VTK utiliza un patrón de diseño de método de fábrica para crear nuevas instancias de clases derivadas de vtkObject usando el método `<ClassName>::New()` . Esto permite seleccionar una implementación específica de plataforma durante el tiempo de ejecución para satisfacer una interfaz requerida.

Para que este mecanismo funcione, las clases de fábrica deben "registrarse" para que puedan ser seleccionadas por la infraestructura vtk. Los detalles sobre este tema están disponibles aquí .

`VTK_MODULE_INIT` es una macro utilizada para inicializar automáticamente los módulos / bibliotecas requeridos (`vtkRenderingOpenGL2` , `vtkRenderingFreeType` , `vtkInteractionStyle` en este ejemplo). Si no se inicializan los módulos, se producirán llamadas a `<ClassName>::New()` para devolver `NULL` y, por lo tanto, errores de tiempo de ejecución.

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

`vtkSmartPointer` función `vtkSmartPointer` es similar a la de un `std::unique_ptr` en que administra el recuento de referencia que controla el tiempo de vida de las instancias de la clase derivada `vtkObject` .

`vtkTextActor` es una clase simple que se puede usar para mostrar cadenas en la pantalla.

`vtkRenderer` es una clase responsable de administrar los contenidos de una escena. En concreto gestiona la recogida de

- Actores 2D derivados de `vtkActor2D`
- Actores 3D derivados de `vtkProp3D`
- Volúmenes: `vtkVolume`
- Cámara: `vtkCamera`
- Luces: `vtkLight`

`vtkRenderWindow` es una clase que proporciona una interfaz independiente de la plataforma para

- gestionando una colección de renderizadores.
- el manejo de la entrada del usuario y el reenvío a `vtkRenderWindowInteractor` para su posterior procesamiento

`vtkRenderWindowInteractor` es una clase responsable de asignar los eventos de entrada del usuario (mouse / teclado / tiempo) a una acción correspondiente. Internamente utiliza un `vtkInteractorStyle` para proporcionar diferentes comportamientos de mapeo.

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

Crear actor de texto y establecer la cadena para mostrar

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

- Crear un renderizador
- Añade el actor de texto a él.

- Restablece la posición de la cámara para asegurarse de que el actor esté visible en la pantalla.

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

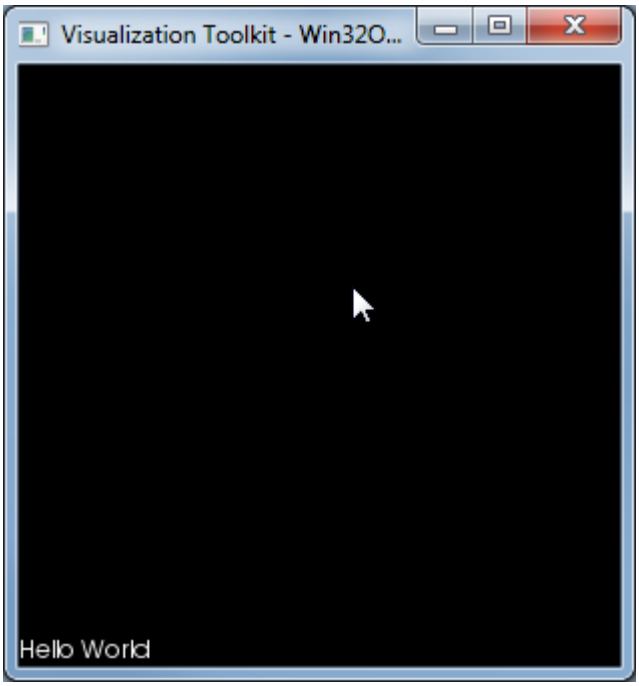
```
auto renderWindow = vtkSmartPointer<vtkRenderWindow>::New();
renderWindow->AddRenderer(renderer);
renderWindow->SetInteractor(interactor);
```

Cree una ventana para renderizar, agregue el renderizador y configure el interactor. La función de fábrica seleccionará automáticamente una implementación adecuada en función de las clases de fábrica disponibles / registradas

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

Esta es una llamada de bloqueo que se devuelve solo cuando el usuario solicita un cierre (tecla `q`) o cierra la ventana. Ejecuta un bucle de mensajes y despacha los mensajes.

Ejecutar esto debería crear una ventana como esta



Notas

Esta lista de DLL que fue utilizada por este exe son:

`VTKCommonCore-7.0.DLL`

`VTKInteractionStyle-7.0.DLL`

`VTKRenderingCore-7.0.DLL`

`VTKRenderingFreeType-7.0.DLL`

```
VTKRenderingOpenGL2-7.0.DLL
```

Lea Hola Mundo en línea: https://riptutorial.com/es/vtk/topic/5974/hola-mundo

Creditos

S. No	Capítulos	Contributors
1	Empezando con vtk	Community, LBes, Shreyas Murali
2	Hola Mundo	Shreyas Murali