EBook Gratuito

APPRENDIMENTO vtk

Free unaffiliated eBook created from
Stack Overflow contributors.

#vtk

Sommario

Di

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

Capitolo 1: Iniziare con vtk

Osservazioni

Questa sezione fornisce una panoramica su cosa sia il vtk e sul motivo per cui uno sviluppatore potrebbe volerlo utilizzare.

Dovrebbe anche menzionare qualsiasi argomento di grandi dimensioni all'interno di vtk e collegarsi agli argomenti correlati. Poiché la documentazione di vtk è nuova, potrebbe essere necessario creare versioni iniziali di tali argomenti correlati.

Examples

Installazione o configurazione

Costruzione e installazione su Windows 7

Prerequisiti

- Se vuoi creare VTK dalle ultime fonti hai bisogno di git da Qui o puoi scaricare un'istantanea del codice come zip e decomprimerlo sul tuo disco
- CMake
- Microsoft Visual Studio 2015
- Un sacco di spazio libero - almeno un paio di GB per essere al sicuro, in realtà dipende da ciò che tutto ciò che si vuole costruire

Prepararsi

- Mi piace mantenere le cose pulite quindi di solito creo 3 cartelle in questo modo:

```
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
```

- Se usi il metodo git,
    - apri un prompt dei comandi
    - cambia la directory di lavoro `cd c:\vtk\src`
    - clonare il repository `git clone https://gitlab.kitware.com/vtk/vtk.git` . Questo potrebbe richiedere del tempo a seconda della velocità della tua connessione internet
    - Se stai lavorando dietro un proxy, dovrai installare git per usarlo. Vedi questa domanda su come farlo.

- Se si utilizza il metodo zip, decomprimere il codice sorgente in `c:\vtk\src`

Configurazione

- Avvia la GUI di CMake
- Selezionare `c:\vtk\src` per `Where is the source code:`
- Selezionare `c:\vtk\build` per `Where to build the binaries:`
- Premi `Configure` e seleziona `Visual Studio 2015` come generatore richiesto
- Ti verrà presentato un numero di opzioni di configurazione
- In genere utilizzo le seguenti impostazioni per una build minima
  - `CMAKE_INSTALL_PREFIX` = `c:\vtk\install`
  - `BUILD_SHARED_LIBS` spuntato
  - `BUILD_DOCUMENTATION`
  - `BUILD_TESTING`
  - `CMAKE_CXX_MP_FLAG` . Questo utilizzerà tutti i core della CPU (su sistemi multicore / multiprocessore) per accelerare la build
- Tenere premuto `Configure` correggere eventuali errori finché tutte le voci RED diventano BIANCHE
- Hit `Generate`
- Chiudi la GUI di CMake

Costruzione

- Se la generazione ha avuto successo, dovrebbe esserci
  - Una soluzione di Visual Studio:

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

  - Un mucchio di file di progetto -

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

- È possibile creare questo utilizzando da una riga di comando o utilizzando l'IDE
- Preferisco la riga di comando in quanto è generalmente più veloce e utilizza meno RAM
- Usando la riga di comando
  - Avviare `Developer Command Prompt For Visual Studio 2015`
  - Cambia directory di lavoro: `cd c:\vtk\build`
  - Avvia msbuild:
    - per le build di debug
      - `msbuild /p:Configuration=Debug ALL_BUILD.vcxproj`
      - `msbuild /p:Configuration=Debug INSTALL.vcxproj`
    - per le versioni di rilascio
      - `msbuild /p:Configuration=Release ALL_BUILD.vcxproj`
      - `msbuild /p:Configuration=Release INSTALL.vcxproj`

- Utilizzando l'IDE
  - Aprire `VTK.sln` con Visual Studio 2015 e `VTK.sln` il `VTK.sln INSTALL.vcxproj`
  - Questa tecnica di solito è più lenta in quanto l'IDE inizierà a costruire intellisense per ciascuno dei progetti elencati nella soluzione
- `c:\vtk\install` dovrebbe ora avere alcune nuove cartelle
  - `bin` # contiene i file dll
  - `lib` # contiene i file lib
  - `cmake`
  - `share`
  - `include` # contiene i file di intestazione

Usando la build

- Per utilizzare VTK in un progetto Visual C ++, è necessario
  - Configurare il percorso di ricerca del file di intestazione del compilatore per includere `c:\vtk\include\vtk-<version>`
  - Configurare il percorso di ricerca del file della libreria linker per includere `c:\vtk\lib`
  - Configura il linker per collegarsi ai file `.lib` richiesti
  - Copia le DLL richieste nella cartella di output
- Ho messo insieme un piccolo file di oggetti di scena per gestire tutte le quattro attività `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>
```

- Il file vsprops sopra copia tutte le DLL disponibili nella cartella `c:\vtk\bin` .
- Un modo alternativo per assicurarsi che le DLL possano essere individuate consiste nell'utilizzare modificare la variabile di ambiente PATH per la sessione di debug e inserire il percorso dei binari VTK come prima directory da cercare quando si caricano le dipendenze. Il seguente frammento può essere invece del compito `CopyVTKBinariesList` per farlo.

  ```
  <PropertyGroup>

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

  </PropertyGroup>
  ```

- Per la distribuzione finale è possibile utilizzare uno strumento come Dependency Walker per rintracciare quali DLL e relative dipendenze vengono utilizzate e raggruppare solo quelle per la ridistribuzione.
- Per utilizzare il file prop in un progetto Visual C ++, puoi utilizzare lo strumento Property Manager in Visual Studio (Menu: View => Property Manager) oppure modificare il vcxproj

utilizzando un editor di testo e aggiungere la seguente riga `<Import Project="C:\vtk\vtk.vsprops" />` sotto le altre importazioni del progetto.

Pulire

- Se desideri recuperare spazio su disco, puoi eliminare la cartella `c:\vtk\build` ma il lato negativo è che non puoi eseguire il debug in vtk

disinstallazione

- Basta eliminare la cartella `c:\vtk` se non si desidera più VTK

MacOSX e Unix:

1. Installa l'ultima versione di CMake disponibile qui
2. Scarica l'ultimo VTK qui .
3. Creare una directory di build per VTK `mkdir <path_to_build_directory`
4. Configura 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` o usa la GUI per farlo con `ccmake <path_to_VTK_directory`
5. Entra nella directory build e usa `make -j` (non devi usare `-j` ma la compilazione è molto lunga.
6. Finalmente usa `make install`

Leggi Iniziare con vtk online: https://riptutorial.com/it/vtk/topic/5182/iniziare-con-vtk

Capitolo 2: Ciao mondo

Examples

Ciao esempio del mondo

```
#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;
}
```

Abbattersi:

```
#include <vtkAutoInit.h>

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

La progettazione VTK utilizza un modello di progettazione del metodo factory per creare nuove istanze di classi derivate vtkObject utilizzando il metodo `<ClassName>::New()` . Ciò consente di selezionare un'implementazione specifica della piattaforma durante il runtime per soddisfare un'interfaccia richiesta.

Affinché questo meccanismo funzioni, le classi factory devono "registrarsi" in modo che possano essere selezionate dall'infrastruttura vtk. I dettagli su questo argomento sono disponibili qui .

`VTK_MODULE_INIT` è una macro utilizzata per inizializzare automaticamente i moduli / le librerie richiesti (`vtkRenderingOpenGL2` , `vtkRenderingFreeType` , `vtkInteractionStyle` in questo esempio). La mancata inizializzazione dei moduli comporterà chiamate `<ClassName>::New()` per restituire `NULL` e quindi errori di runtime.

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

`vtkSmartPointer` ruolo `vtkSmartPointer` è simile a quello di `std::unique_ptr` in quanto gestisce il conteggio dei riferimenti che controlla la durata delle istanze di classe derivate `vtkObject` .

`vtkTextActor` è una classe semplice che può essere utilizzata per visualizzare le stringhe sullo schermo.

`vtkRenderer` è una classe responsabile della gestione dei contenuti di una scena. In particolare gestisce la raccolta di

- Attori 2D derivati da `vtkActor2D`
- Attori 3D derivati da `vtkProp3D`
- Volumi: `vtkVolume`
- Fotocamera: `vtkCamera`
- Luci: `vtkLight`

`vtkRenderWindow` è una classe che fornisce un'interfaccia indipendente dalla piattaforma

- gestire una collezione di riproduttori.
- gestire l'input dell'utente e inoltrarlo a `vtkRenderWindowInteractor` per un'ulteriore elaborazione

`vtkRenderWindowInteractor` è una classe responsabile della mappatura degli eventi di input dell'utente (mouse / tastiera / timing) ad un'azione corrispondente. Internamente utilizza un `vtkInteractorStyle` per fornire diversi comportamenti di mappatura.

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

Crea un attore di testo e imposta la stringa da visualizzare

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

- Crea un renderer
- Aggiungi l'attore di testo ad esso
- Ripristina la posizione della telecamera per assicurarsi che l'attore sia visibile sullo schermo.

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

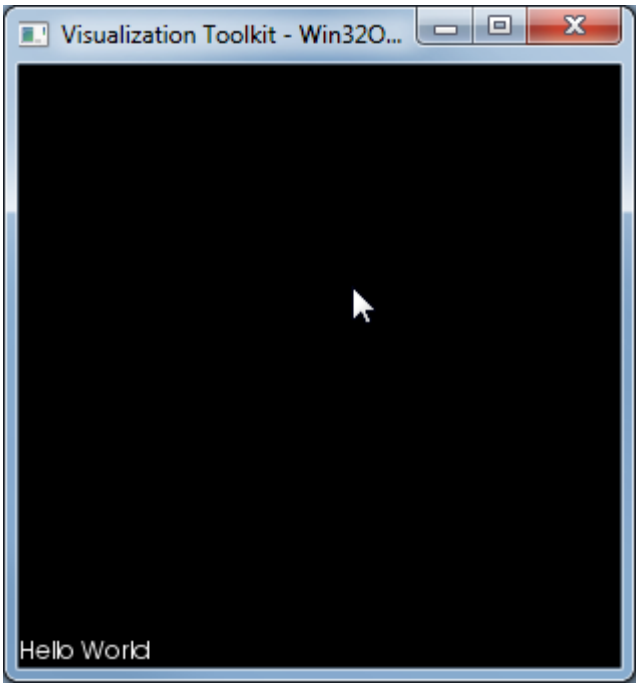
```
auto renderWindow = vtkSmartPointer<vtkRenderWindow>::New();
renderWindow->AddRenderer(renderer);
renderWindow->SetInteractor(interactor);
```

Crea una finestra per eseguire il rendering, aggiungi il renderer e imposta l'interactor. La funzione di fabbrica sceglierà automaticamente un'implementazione adatta in base alle classi di fabbrica disponibili / registrate

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

Questa è una chiamata di blocco che viene restituita solo quando l'utente richiede una chiusura (tasto `q`) o chiude la finestra. Esegue un ciclo di messaggi e invia i messaggi.

L'esecuzione di questo dovrebbe creare una finestra simile a questa



Gli appunti

Questo elenco di DLL che sono stati utilizzati da questo exe sono:

`VTKCommonCore-7.0.DLL`

`VTKInteractionStyle-7.0.DLL`

`VTKRenderingCore-7.0.DLL`

`VTKRenderingFreeType-7.0.DLL`

`VTKRenderingOpenGL2-7.0.DLL`

Leggi Ciao mondo online: https://riptutorial.com/it/vtk/topic/5974/ciao-mondo

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con vtk	Community, LBes, Shreyas Murali
2	Ciao mondo	Shreyas Murali