eBook Gratuit

APPRENEZ vtk

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#vtk

Table des matières

À propos

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

Chapitre 1: Commencer avec vtk

Remarques

Cette section fournit une vue d'ensemble de ce que vtk est et pourquoi un développeur peut vouloir l'utiliser.

Il devrait également mentionner tous les grands sujets au sein de vtk, et établir un lien avec les sujets connexes. La documentation de vtk étant nouvelle, vous devrez peut-être créer des versions initiales de ces rubriques connexes.

Examples

Installation ou configuration

Construction et installation sur Windows 7

Conditions préalables

- Si vous souhaitez créer VTK à partir des dernières sources, vous devez télécharger Here ou vous pouvez télécharger un instantané du code sous forme de fichier zip et le décompresser sur votre disque dur.
- CMake
- Microsoft Visual Studio 2015
- Beaucoup d'espace libre - au moins deux Go pour être sûr, tout dépend de ce que vous voulez construire

Se préparer

- J'aime garder les choses propres, donc je crée généralement 3 dossiers comme ceci:

```
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 vous utilisez la méthode git,
    - ouvrir une invite de commande
    - changer le répertoire de travail `cd c:\vtk\src`
    - cloner le clone git du dépôt `git clone https://gitlab.kitware.com/vtk/vtk.git` . Cela peut prendre un certain temps en fonction de la vitesse de votre connexion Internet
    - Si vous travaillez derrière un proxy, vous devrez configurer git pour l'utiliser. Voir cette

question sur la façon de le faire.

- Si vous utilisez la méthode zip, décompressez le code source dans `c:\vtk\src`

Configuration

- Lancer l'interface graphique CMake
- Sélectionnez `c:\vtk\src` pour `Where is the source code:`
- Sélectionnez `c:\vtk\build` pour `Where to build the binaries:`
- Appuyez sur `Configure` et sélectionnez `Visual Studio 2015` comme générateur requis
- Vous serez présenté avec un certain nombre d'options de configuration
- J'utilise généralement les paramètres suivants pour une construction minimale
  - `CMAKE_INSTALL_PREFIX` = `c:\vtk\install`
  - `BUILD_SHARED_LIBS` coché
  - `BUILD_DOCUMENTATION` décoché
  - `BUILD_TESTING` unticked
  - `CMAKE_CXX_MP_FLAG` coché. Cela va utiliser tous les cœurs du processeur (sur les systèmes multicœur / multiprocesseur) pour accélérer la construction
- Garder la frappe `Configure` correction des erreurs jusqu'à ce que toutes les entrées ROUGES deviennent BLANCES
- Hit `Generate`
- Fermer l'interface graphique de CMake

Bâtiment

- Si la génération a réussi, il devrait y avoir
  - Une solution Visual Studio:

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

  - Un tas de fichiers de projet -

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

- Vous pouvez créer ceci en utilisant une ligne de commande ou en utilisant l'EDI
- Je préfère la ligne de commande car elle est généralement plus rapide et utilise moins de RAM
- Utiliser la ligne de commande
  - Lancer l' `Developer Command Prompt For Visual Studio 2015`
  - Changer le répertoire de travail: `cd c:\vtk\build`
  - Lancer msbuild:
    - pour les versions de débogage
      - `msbuild /p:Configuration=Debug ALL_BUILD.vcxproj`
      - `msbuild /p:Configuration=Debug INSTALL.vcxproj`

        - pour les versions release
            - `msbuild /p:Configuration=Release ALL_BUILD.vcxproj`
            - `msbuild /p:Configuration=Release INSTALL.vcxproj`
- Utiliser l'IDE
    - Ouvrez le `VTK.sln` avec Visual Studio 2015 et `INSTALL.vcxproj` le `INSTALL.vcxproj`
    - Cette technique est généralement plus lente car l'EDI commence à construire intellisense pour chacun des projets répertoriés dans la solution.
- `c:\vtk\install` devrait maintenant avoir quelques nouveaux dossiers
    - `bin` # contient les fichiers dll
    - `lib` # contient les fichiers lib
    - `cmake`
    - `share`
    - `include` # contient les fichiers d'en-tête

Utiliser la construction

- Pour utiliser VTK dans un projet Visual C ++, il faut
    - Configurez le chemin de recherche du fichier d'en-tête du compilateur pour inclure `c:\vtk\include\vtk-<version>`
    - Configurez le chemin de recherche du fichier de la bibliothèque de l'éditeur de liens pour inclure `c:\vtk\lib`
    - Configurez l'éditeur de liens pour créer un lien vers les fichiers `.lib` requis
    - Copiez les DLL requises dans le dossier de sortie
- J'ai mis en place un petit fichier d'accessoires pour gérer les quatre tâches `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>
```

- Le fichier vsprops ci-dessus copie toutes les DLL disponibles dans le dossier `c:\vtk\bin` .
- Une autre façon de vous assurer que les DLL peuvent être localisées consiste à modifier la variable d'environnement PATH pour la session de débogage et à placer le chemin des fichiers binaires VTK en tant que premier répertoire à rechercher lors du chargement des dépendances. Le fragment ci-dessous peut être à la place de la tâche `CopyVTKBinariesList` pour cela.

```
<PropertyGroup>

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

</PropertyGroup>
```

- Pour le déploiement final, vous souhaiterez peut-être utiliser un outil tel que Dependency

Walker pour identifier les DLL et leurs dépendances utilisées et ne regrouper que celles qui seront redistribuées.

- Pour utiliser le fichier props dans un projet Visual C ++, vous pouvez utiliser l'outil Property Manager dans Visual Studio (Menu: View => Property Manager) ou modifier le vcxproj à l'aide d'un éditeur de texte et ajouter cette ligne `<Import Project="C:\vtk\vtk.vsprops" />` sous les autres importations du projet.

Nettoyer

- Si vous souhaitez récupérer de l'espace disque, vous pouvez supprimer le dossier `c:\vtk\build` mais l'inconvénient est que vous ne pouvez pas déboguer dans vtk

Désinstallation

- Supprimez simplement le dossier `c:\vtk` si vous ne voulez plus utiliser VTK

MacOSX et Unix:

1. Installez la dernière version de CMake disponible ici
2. Téléchargez la dernière version de VTK ici .
3. Créez un répertoire de génération pour VTK `mkdir <path_to_build_directory`
4. Configurez avec `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` ou utilisez l'interface graphique pour le faire avec `ccmake <path_to_VTK_directory`
5. Entrez dans le répertoire de construction et utilisez `make -j` (vous n'avez pas à utiliser `-j` mais la compilation est très longue.
6. Enfin utiliser `make install`

Lire Commencer avec vtk en ligne: https://riptutorial.com/fr/vtk/topic/5182/commencer-avec-vtk

Chapitre 2: Bonjour le monde

Examples

Bonjour Monde Exemple

```
#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;
}
```

Panne:

```
#include <vtkAutoInit.h>

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

La conception de VTK utilise un modèle de conception de méthode d'usine pour créer de nouvelles instances de classes dérivées de vtkObject à l'aide de la méthode `<ClassName>::New()` . Cela permet à une implémentation spécifique à une plate-forme d'être sélectionnée pendant l'exécution pour satisfaire une interface requise.

Pour que ce mécanisme fonctionne, les classes d'usine doivent "s'inscrire" elles-mêmes afin qu'elles puissent être sélectionnées par l'infrastructure vtk. Les détails sur ce sujet sont disponibles ici .

`VTK_MODULE_INIT` est une macro utilisée pour initialiser automatiquement les modules / bibliothèques requis (`vtkRenderingOpenGL2` , `vtkRenderingFreeType` , `vtkInteractionStyle` dans cet exemple). `<ClassName>::New()` pas à initialiser les modules, les `<ClassName>::New()` des erreurs `NULL` et donc à l'exécution.

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

`vtkSmartPointer` rôle `vtkSmartPointer` est similaire à celui d'un `std::unique_ptr` en ce qu'il gère le compteur de références qui contrôle la durée de vie des `vtkObject` de `vtkObject` dérivées de `vtkObject` .

`vtkTextActor` est une classe simple qui peut être utilisée pour afficher des chaînes à l'écran.

`vtkRenderer` est une classe chargée de gérer le contenu d'une scène. Plus précisément, il gère la collecte de

- Acteurs 2D dérivés de `vtkActor2D`
- Acteurs 3D dérivés de `vtkProp3D`
- Volumes: `vtkVolume`
- Caméra: `vtkCamera`
- Lumières: `vtkLight`

`vtkRenderWindow` est une classe qui fournit une interface indépendante de la plate-forme pour

- gérer une collection de rendus.
- gérer les entrées de l'utilisateur et les transmettre à `vtkRenderWindowInteractor` pour un traitement ultérieur

`vtkRenderWindowInteractor` est une classe chargée de mapper les événements d'entrée utilisateur (souris / clavier / synchronisation) sur une action correspondante. En interne, il utilise un `vtkInteractorStyle` pour fournir différents comportements de mappage.

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

Créer un acteur de texte et définir la chaîne à afficher

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

- Créer un moteur de rendu

- Ajoutez-y l'acteur de texte
- Réinitialise la position de la caméra pour vous assurer que l’acteur est visible sur l’écran.

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

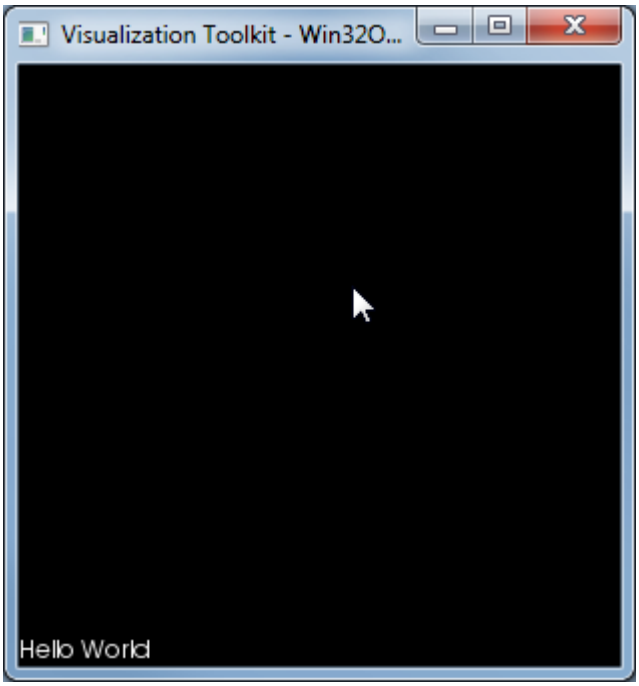
```
auto renderWindow = vtkSmartPointer<vtkRenderWindow>::New();
renderWindow->AddRenderer(renderer);
renderWindow->SetInteractor(interactor);
```

Créez une fenêtre dans laquelle effectuer le rendu, ajoutez-y le moteur de rendu et définissez l'interacteur. La fonction d'usine choisira automatiquement une implémentation appropriée en fonction des classes d'usine disponibles / enregistrées

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

Il s'agit d'un appel bloquant qui ne renvoie que lorsque l'utilisateur demande une fermeture (touche `q`) ou ferme la fenêtre. Exécute une boucle de message et distribue les messages.

En cours d'exécution cela devrait créer une fenêtre qui ressemble à ceci



Remarques

Cette liste de DLL utilisées par cet exe est la suivante:

`VTKCommonCore-7.0.DLL`

`VTKInteractionStyle-7.0.DLL`

`VTKRenderingCore-7.0.DLL`

`VTKRenderingFreeType-7.0.DLL`

```
VTKRenderingOpenGL2-7.0.DLL
```

Lire Bonjour le monde en ligne: https://riptutorial.com/fr/vtk/topic/5974/bonjour-le-monde

Crédits

S. No	Chapitres	Contributeurs
1	Commencer avec vtk	Community, LBes, Shreyas Murali
2	Bonjour le monde	Shreyas Murali