A primeira etapa é criar os arquivos necessários para o projeto.

1)O controle de versão e dependências

1.1) Controle de versão: Cria-se o projeto no github, com um readme.md e um gitignore. Depois faz-se a clonagem desse projeto para a máquina.

É inaceitável nos dias de hoje que se faça qualquer trabalho que programação que não seja trivial sem um controle de versão para poder reverter mudanças caso necessárias ou criar várias árvores de código, podendo avançar com a base de código mantendo a versão que está na produção intacta, com cada uma em uma árvore. Além de gerenciar múltiplas mãos mexendo no código.

1.2) As dependências: O projeto depende da ITK e da VTK. Elas devem ser baixadas de seus respectivos githubs e construídas. Abaixo será explicado como construir projetos com o Cmake.

2)O Cmake.

2.1) O que é: O Cmake é um sistema de criação de arquivos de projeto, como os makefiles e as soluções do visual studio, simplificando a linkagem das libs e o acesso aos headers. Cria-se um CMakeLists.txt no diretório do projeto e pelo menos um arquivo .cpp.

2.2) CmakeLists.txt do projeto: O CmakeLists.txt terá o seguinte código:

cmake\_minimum\_required(VERSION 2.8)

#nome do projeto

project(Segmentator)

#Seta os paths das coisas necessárias

set(ITK\_DIR "C:/libs/itk/ITK\_bin")#itk

set(VTK\_DIR "C:/libs/VTK\_BIN")#vtk

set(D3D\_LIBRARY "d3d9.lib")#necessário pra pegar o tamanho da memória da GPU.

set(WBEM\_LIBRARY "wbemuuid.lib")#necessário pra pegar o tamanho da memória da GPU.

# Find ITK.

find\_package(ITK REQUIRED)

include(${ITK\_USE\_FILE})

# Find VTK

find\_package(VTK REQUIRED)

include(${VTK\_USE\_FILE})

#OpenGL

find\_package(OpenGL REQUIRED)

#pega os arquivos

file(GLOB my\_source\_files "\*.h" "\*.cpp")

#Define um executável

add\_executable(segmentator ${my\_source\_files} )

#Liga as bibliotecas ao executável definido

target\_link\_libraries(segmentator ${OPENGL\_LIBRARIES})

target\_link\_libraries(segmentator ${ITK\_LIBRARIES} ${VTK\_LIBRARIES})

target\_link\_libraries(segmentator ${D3D\_LIBRARY} ${WBEM\_LIBRARY})

A 1ª linha especifica a versão mínima do cmake que entende o arquivo. Depois define-se o nome do projeto, no caso Segmentator. As próximas linhas determinam localização de bibliotecas que irão ser usadas. Algumas dessas bibliotecas já estão preparadas para uso com o cmake por seus criadores, como a ITK e a VTK, outras tem que especificar o caminho para o .lib, como as do directx.

Determinado os caminhos das bibliotecas, pede-se para o cmake achar os pacotes, algo que só funciona para bibliotecas que tenham sido feitas para funcionar como pacotes no cmake (como é o caso da ITK e VTK) e em bibliotecas que o CMake entenda como achar, no caso do OpenGL. O REQUIRED diz que se o pacote não for achado o projeto não pode ser construído.

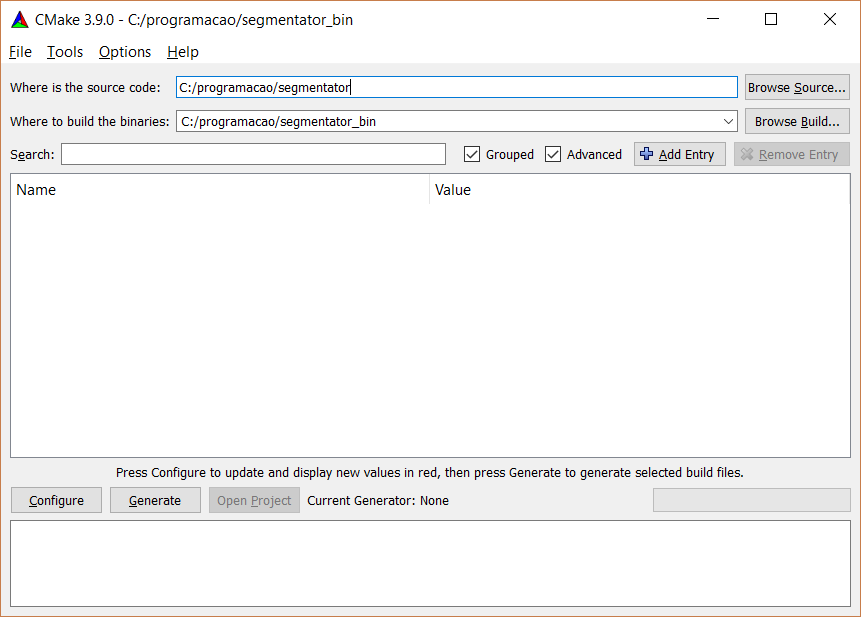
file(GLOB my\_source\_files "\*.h" "\*.cpp")busca os arquivos do código fonte e os agrupa em uma variável, que será usada mais na frente para definir um executável.

add\_executable(segmentator ${my\_source\_files} ) define um executável, de nome segmentador. O executável não precisa ter o mesmo nome que o projeto e um projeto pode ter mais de um executável (e bibliotecas, que não serão usadas no momento).

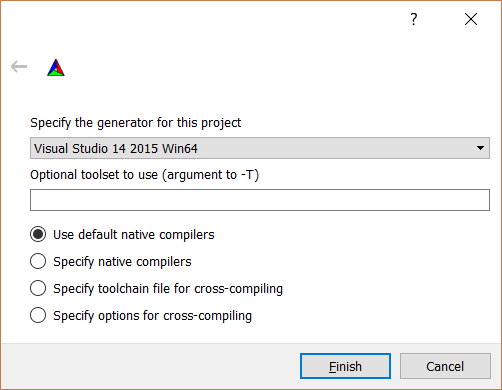
Definido o executável vamos fazer a linkagem entre o executável e as bibliotecas previamente definidas. Isso é feito com target\_link\_libraries, tendo como parâmetro o nome do executável e em seguida a variável que guarda as biblioteca sendo linkada.

3) Usando o CMake-GUI para gerar o makefile:

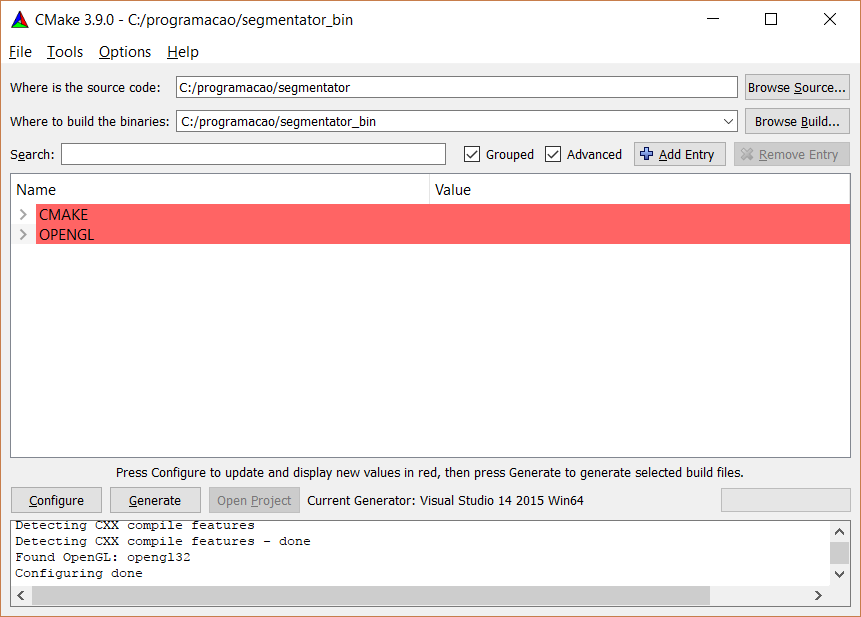
3.1) Informar onde está o código-fonte e onde se deseja gerar os binários. A boa prática diz que o código fonte e o binário devem ficar separados. Algumas bibliotecas se recusam a gerar os arquivos se os dois diretórios forem iguais. Eu vejo a vantagem dessa separação em manter o diretório do código fonte limpo de código compilado e arquivos de configuração, para não criar complicação para o controle de versão.



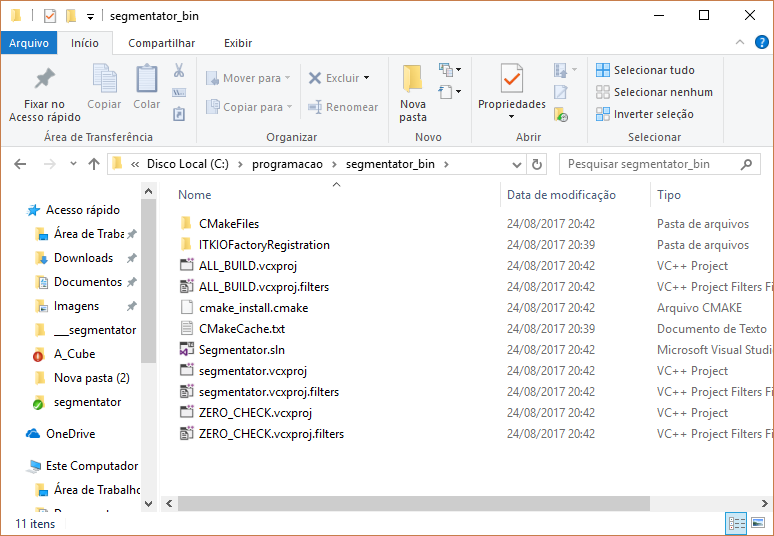
3.2) Configurar: Ao clicar no botão configurar será abeta uma tela para escolher o gerador. É nesse momento que se escolhe entre unix makefiles, soluções de visual studio e outras opções. Como vou trabalhar no Windows usando o visual studio, escolho a minha versão do visual studio e mando gerar. O cmake fará uma série de processamentos e verificações para gerar o arquivo de solução. Se houverem erros no cmakelists.txt eles aparecerão nessa hora assim como se houverem configurações adicionais necessárias elas serão pedidas agora. Por exemplo, se eu ao invés de especificar hardcoded onde está a itk e a vtk no meu sistema seria nessa hora que eu as especificaria pela tela do cmake-gui



3-3) Gerar o arquivo de solução: Se não houver pendências no cmake, basta clicar no botão “generate” para gerar o arquivo da solução. O cmake fará mais um processamento e o resultado final será os arquivos de projeto no diretório pedido.



Resultado final:



3.4) Gerando a ITK e a VTK: como dito anteriormente o projeto depende da ITK e da VTK. Para gerar os binários dessas bibliotecas deve-se fazer como foi dito acima sobre como usar o cmake-gui para gerar os binários do Segmentator. O processo completo de geração da ITK e da VTK demorará um pouco. Uma vez geradas, deve-se abrir o arquivo de projeto das bibliotecas como o visual studio em modo de administrador e ordenar a compilação dos projetos BUILD\_ALL e INSTALL. Se não estiver em modo de administrador INSTALL falhará porque ele escreve arquivos no diretório Program Files. A compilação das bibliotecas será lenta, ainda mais se a máquina tiver poucos núcleos e não tiver um SSD.

Conclusão: Com isso a preparação do projeto está concluída e podemos partir para a criação do programa. Os resultados se encontram em <https://github.com/geronimo-lisboa/segmentator> .

Caso vá usar o unix ou apple, as partes relativas à directx no cmakelists.txt terão que ser removidas. Essas partes são usadas para saber o tamanho da memória de vídeo no Windows, com os outros sistemas operacionais tendo mecanismos específicos. No futuro farei uma adaptação