

ISEN

ALL IS DIGITAL!

LILLE



yncréa



Programmation Orienté Objet / C++

Tutorial : quelques outils/informations additionnelles pour faciliter la programmation du projet

pascal.mosbah@yncrea.fr

Nacim IHADDADEN, Pascal MOSBAH

Projet

Modélisation d'une épreuve de course à pied

Présentation de la bibliothèque SFML et d'outils pour l'aide à la programmation

- Git
- Cmake
- SFML

Git installation

Prérequis

Disposer d'un accès à un dépôt Git (par exemple un compte sur GitHub pour créer ses propres dépôts).

Pour Visual Studio

Source d'information sur l'utilisation de Git sous Visual Studio

<https://github.com/github/VisualStudio/blob/master/docs/using/index.md>

Installer l'extension GitHub Extension for Visual Studio :

Dans Tools/Get tools and Features...

Install Individual Components :

- Git for Windows

Pour Linux Ubuntu ou distribution dérivée de Debian:

- `sudo apt install git`

Pour MacOS

- `brew install git` (devrait être inclus dans l'environnement de développement Xcode)

Cmake installation

Prérequis

Pour Visual Studio

Source d'information principale sur l'utilisation de cmake sous Visual Studio:

<https://docs.microsoft.com/en-us/cpp/build/cmake-projects-in-visual-studio?view=msvc-160>

Dans Tools/Get tools and Features...

Install Individual Components :

- Installer l'extension C++ CMake tools for Windows
- Optionnel mais très utile : <https://cmake.org/download>

Pour Linux : Ubuntu ou distribution dérivée de Debian:

- `sudo apt install cmake`

Pour MacOS

- <https://cmake.org/download> (devrait être inclus dans l'environnement de développement Xcode)

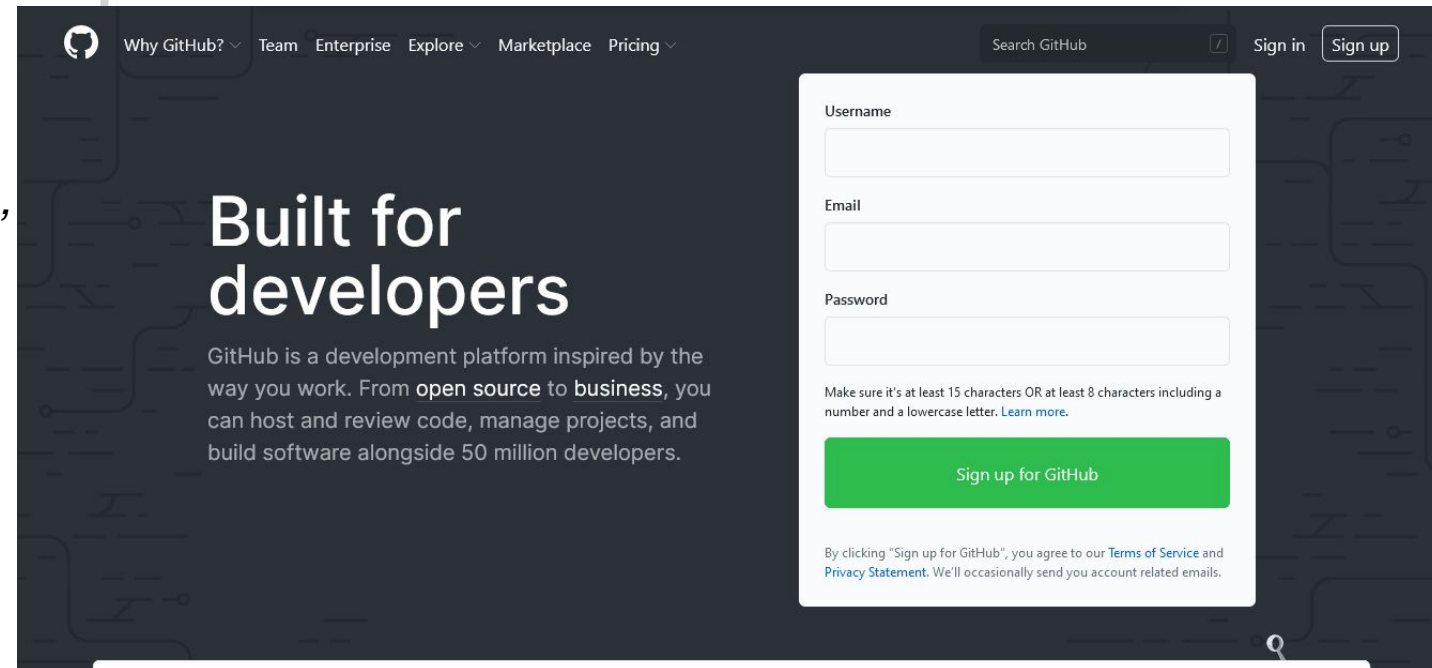
<https://fr.wikipedia.org/wiki/Git>

“**Git** est un [logiciel de gestion de versions décentralisé](#). C'est un [logiciel libre](#) créé par [Linus Torvalds](#), auteur du [noyau Linux](#), et distribué selon les termes de la [licence publique générale GNU](#) version 2. En 2016, il s'agit du [logiciel de gestion de versions](#) le plus populaire qui est utilisé par plus de douze millions de personnes³. ”

<https://fr.wikipedia.org/wiki/GitHub>

GitHub ([/git hʌb/](#); exploité sous le nom de *GitHub, Inc.*) est un service web d'[hébergement](#) et de gestion de développement de logiciels, utilisant le [logiciel de gestion de versions Git](#).

Création d'un compte sous GitHub

The image shows the GitHub website's sign-up page. The background is dark with a subtle pattern of code and icons. On the left, the text "Built for developers" is prominently displayed in white, followed by a description of GitHub as a development platform. On the right, there is a white sign-up form with fields for "Username", "Email", and "Password". Below the password field, there is a note about password requirements and a green "Sign up for GitHub" button. At the bottom of the form, there is a small line of text about agreeing to terms of service and privacy policy.

Why GitHub? Team Enterprise Explore Marketplace Pricing Search GitHub Sign in Sign up

Built for developers

GitHub is a development platform inspired by the way you work. From **open source** to **business**, you can host and review code, manage projects, and build software alongside 50 million developers.

Username

Email

Password

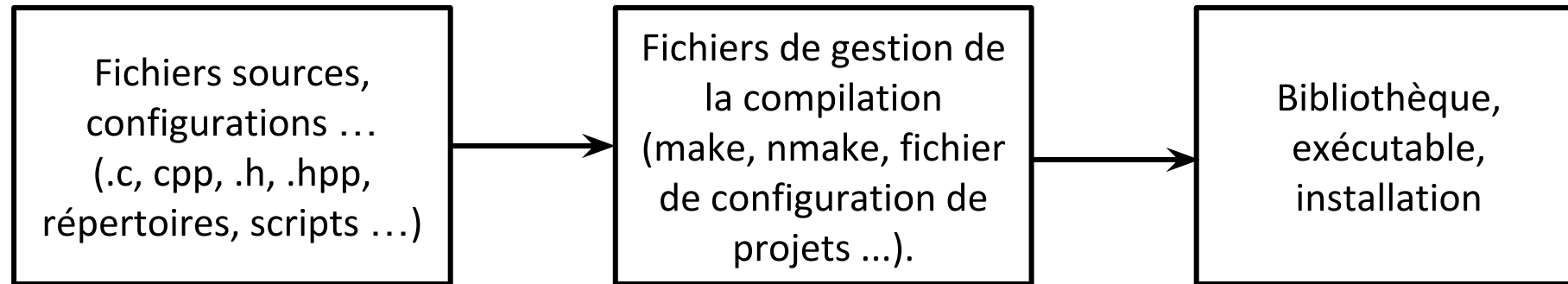
Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)

Sign up for GitHub

By clicking "Sign up for GitHub", you agree to our [Terms of Service](#) and [Privacy Statement](#). We'll occasionally send you account related emails.

Intérêt pour l'utilisation de Git pour l'ensemble de vos projets :

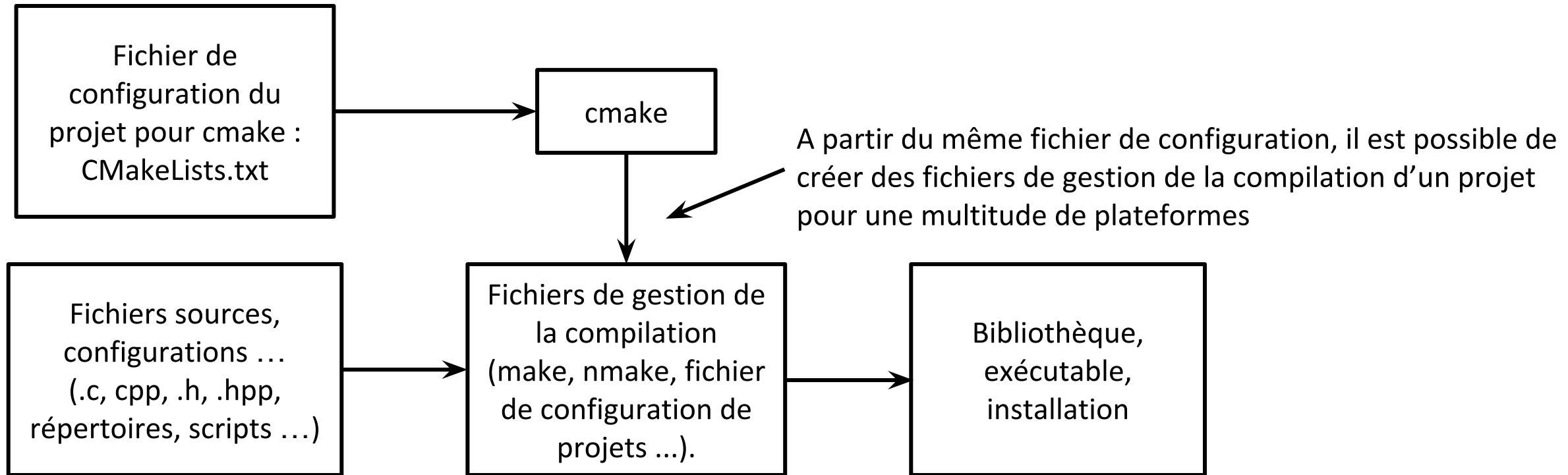
- *“Mon disque dur a “planté”, j’ai tout perdu et ne peux pas rendre mon travail !”*
Les fichiers sources sont à la fois sauvegardés sur l'ordinateur et un serveur Git distant.
- *“J’ai effectué des modifications dans mon programme et depuis plus rien ne fonctionne. Je n’arrive pas à retrouver ce qui a été modifié et qui pose problème”*
Git sauvegarde les modifications entre deux versions d'un programme.
- *“J’aimerais bien essayer une nouvelle solution mais j’ai peur de tout casser...”*
Git peut gérer différentes branches de développement.
- *“Nous sommes plusieurs à travailler sur un projet et nous n’arrivons pas à rassembler notre travail!”*
Git peut aider à la détection et la gestion des conflits en cas de modifications parallèles de fichiers sources.



Processus général de génération de “cibles” (bibliothèque, exécutable, sauvegarde...)

La gestion de la compilation est généralement propre à chaque système suivant des directives précisées dans des fichiers de configuration (fichier “Makefile” (Unix), projet “.sln” ou NMakefile (Visual C++), ...

cmake est un outil de gestion de la compilation multisource et multilangage d'un projet



Quelques dépôts git “publics” (sans mots de passe) comprenant des exemples d’utilisation de cmake et SFML:

https://github.com/pascal59/test_cmake

Simple test de l’utilisation de cmake

https://github.com/pascal59/test_SFML2

Démonstration de l’utilisation de SFML pour générer des diagrammes à barre

https://github.com/pascal59/test_SFML

Démonstration de l’utilisation de SFML pour afficher une voiture sur un circuit conduite par des touches au clavier, tirée d’un exemple trouvé sur l’internet (référence à venir)



https://github.com/pascal59/test_SFML1

Démonstration de l’utilisation de SFML pour afficher une voiture sur un circuit suivi par des personnages animées tirée de différents exemples trouvés sur l’internet (références à venir)




Visual Studio 2019

Open recent

Today


-  **example.sln** 11/5/2020 3:59 PM
C:\Users\pmo\source\repos\example
-  **test_cmake** 11/5/2020 8:57 AM
C:\Users\pmo\Source\Repos

Yesterday


-  **fisrt_test.sln** 11/4/2020 7:20 PM
C:\Users\pmo\source\repos\test_cmake\Build
-  **fisrt_test.sln** 11/4/2020 6:04 PM
C:\Users\pmo\source\repos\my_functions\test_cmake\b...
-  **SFML_Test2.sln** 11/4/2020 3:17 PM
C:\Users\pmo\source\repos\test_SFML2\build

Get started

 **Clone a repository**
Get code from an online repository like GitHub or Azure DevOps

 **Open a project or solution**
Open a local Visual Studio project or .sln file

 **Open a local folder**
Navigate and edit code within any folder

 **Create a new project**
Choose a project template with code scaffolding to get started

[Continue without code →](#)

Clone a repository

Enter a Git repository URL

Repository location

Local path

https://github.com/pascal59/test_cmake

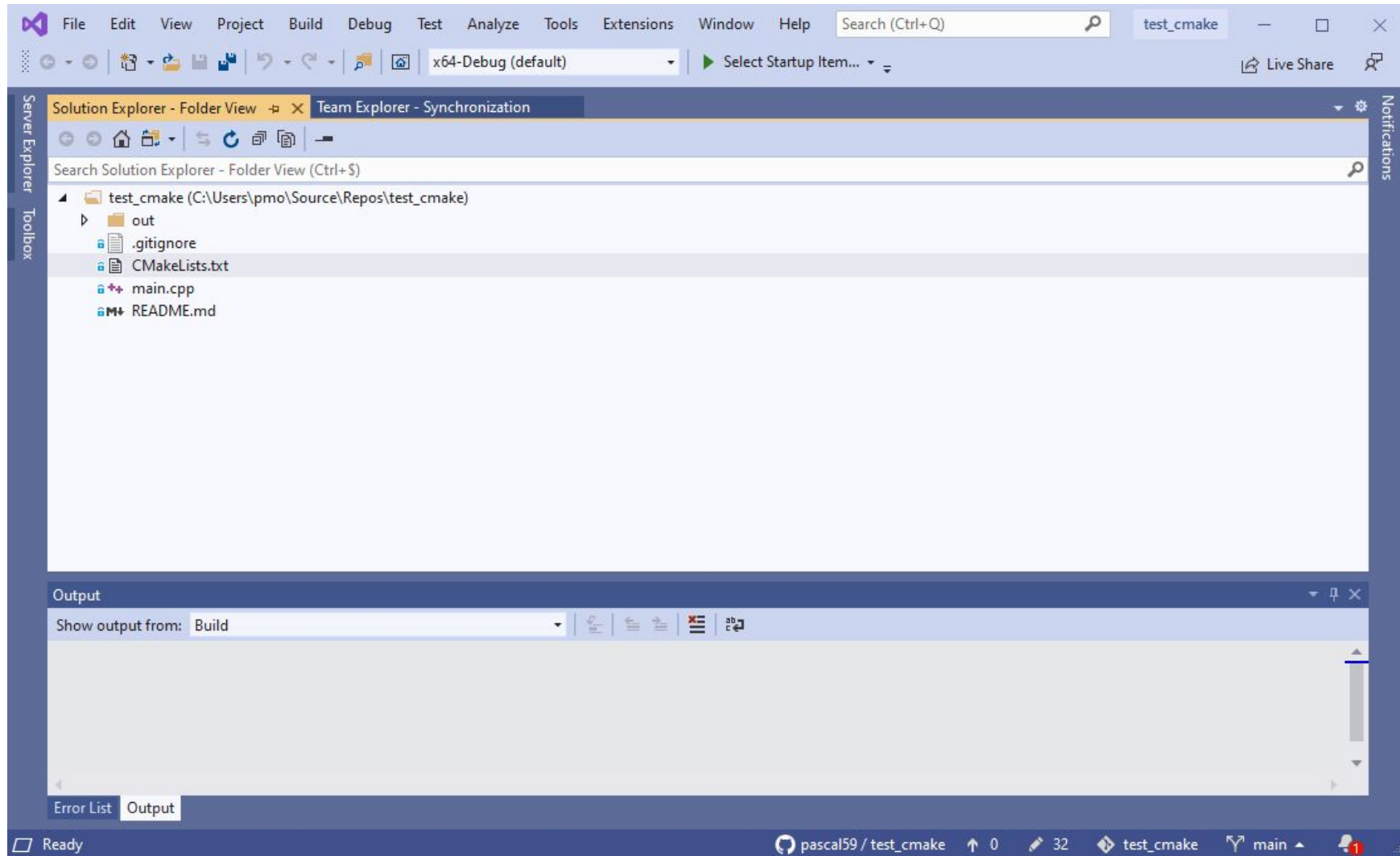
Browse a repository

 Azure DevOps

 GitHub

Back

Clone



Un très simple exemple d'utilisation de cmake

(le fichier CMakeLists.txt peut être directement utilisé sous Visual Studio ou avec l'outil CMake)

{ CMakeLists.txt
main.cpp }

Deux fichiers :

Fichier source : main.cpp,

Fichier de configuration de cmake : CMakeLists.txt

build

Si utilisation avec cmake directement, il est préférable de créer un répertoire de travail (build)

main.cpp :

```
#include <cstdlib>
#include <iostream>
```

```
int main(){
    std::cout << "Hello world\n";
}
```

CMakeLists.txt :

```
project(first_test)
```

```
set(SRCS "main.cpp")
```

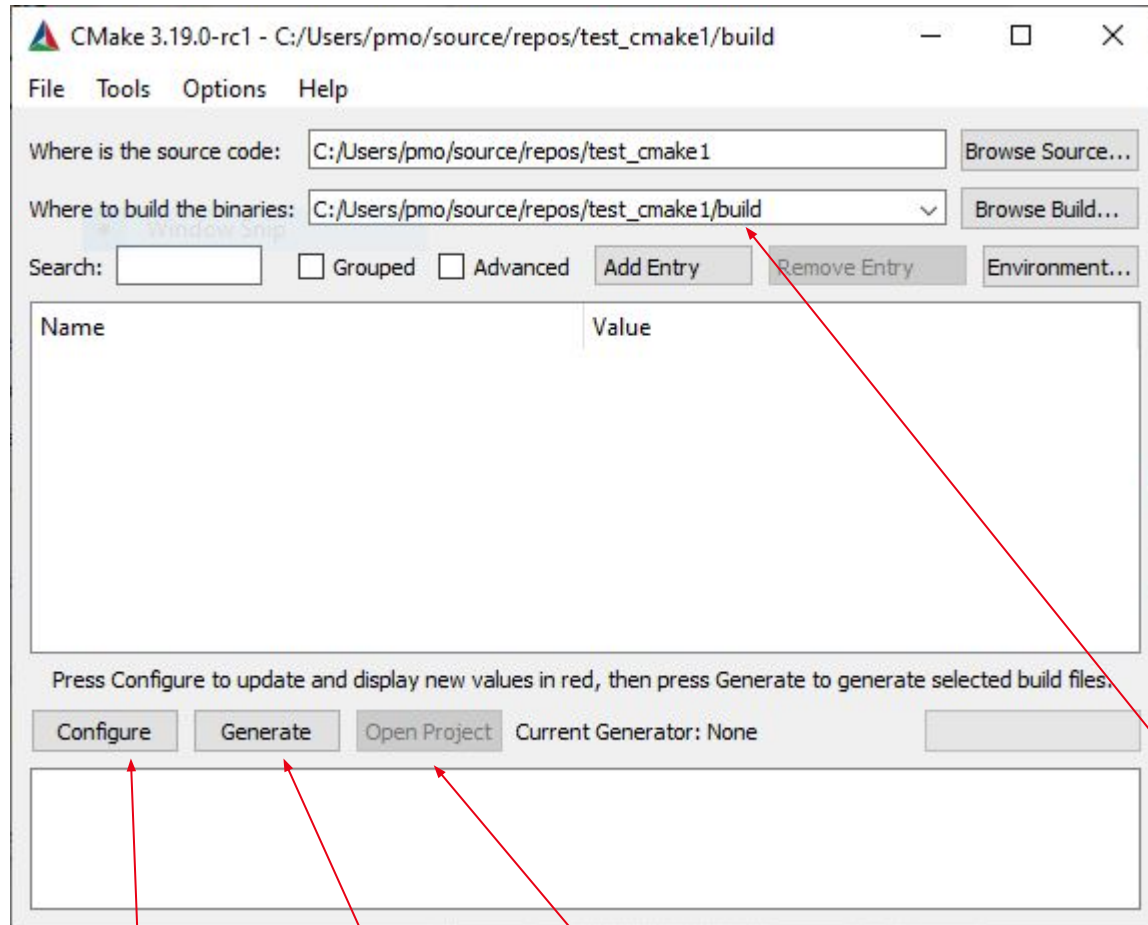
```
add_executable (first_test ${SRCS})
```

Nom du projet

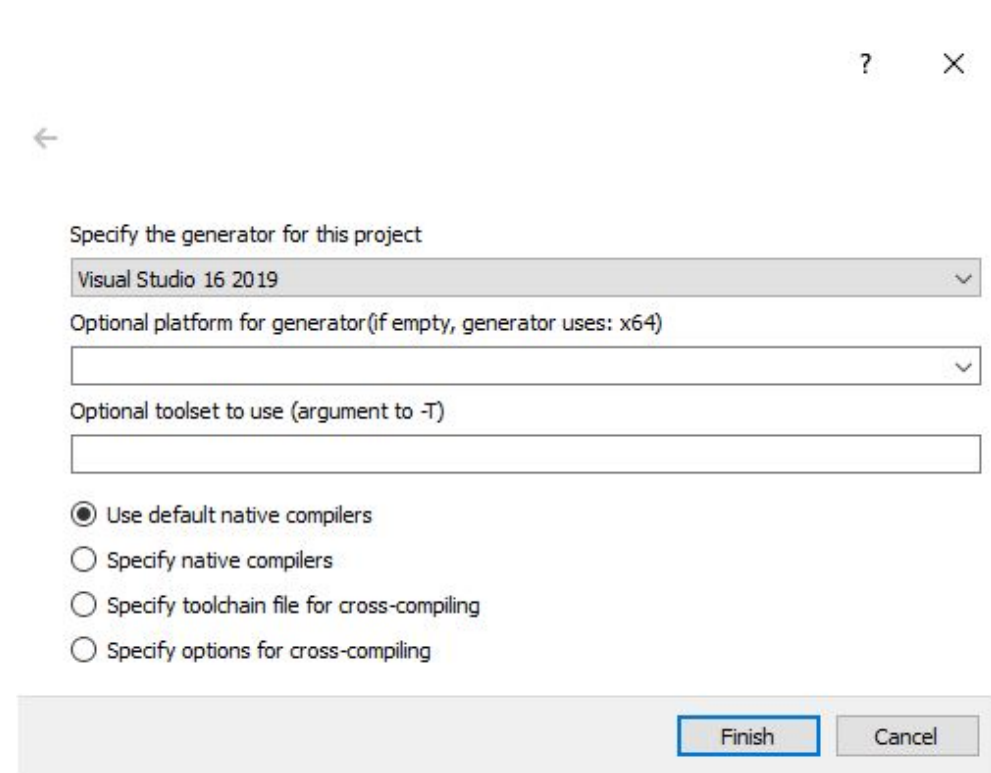
Liste de fichiers sources
pour cet exemple
main.cpp

Association de la liste des fichiers source à l'exécutable tous les autres
paramètres sont générés par défaut par cmake

Génération d'une application



Configure + Generate + Open Project



Répertoire "build" à créer

Sous Linux (MacOS ?)

```
pmo@DESKTOP-IE723GF:~/test$ git clone https://github.com/pascal59/test_cmake
Cloning into 'test_cmake'...
remote: Enumerating objects: 16, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 16 (delta 4), reused 5 (delta 1), pack-reused 0
Unpacking objects: 100% (16/16), 5.80 KiB | 742.00 KiB/s, done.
pmo@DESKTOP-IE723GF:~/test$ cd test_cmake/
pmo@DESKTOP-IE723GF:~/test/test_cmake$ ls
CMakeLists.txt README.md main.cpp
pmo@DESKTOP-IE723GF:~/test/test_cmake$ mkdir build
pmo@DESKTOP-IE723GF:~/test/test_cmake$ cd build/
```

```
pmo@DESKTOP-IE723GF:~/test/test_cmake/build$ cmake ..
-- The C compiler identification is GNU 9.3.0
-- The CXX compiler identification is GNU 9.3.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/pmo/test/test_cmake/build
pmo@DESKTOP-IE723GF:~/test/test_cmake/build$ make
Scanning dependencies of target first_test
[ 50%] Building CXX object CMakeFiles/first_test.dir/main.cpp.o
[100%] Linking CXX executable first_test
[100%] Built target first_test
pmo@DESKTOP-IE723GF:~/test/test_cmake/build$
```

Sous Linux (MacOS ?)

```
pmo@DESKTOP-IE723GF:~/test/test_cmake$ vi main.cpp
pmo@DESKTOP-IE723GF:~/test/test_cmake$ git commit -a -m"My changes"
[main b1feafc] My changes
1 file changed, 1 insertion(+), 1 deletion(-)
pmo@DESKTOP-IE723GF:~/test/test_cmake$ git push
Username for 'https://github.com': pascal59
Password for 'https://pascal59@github.com':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 298 bytes | 298.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/pascal59/test_cmake
66bc397..b1feafc main -> main
pmo@DESKTOP-IE723GF:~/test/test_cmake$ git pull
Already up to date.
```

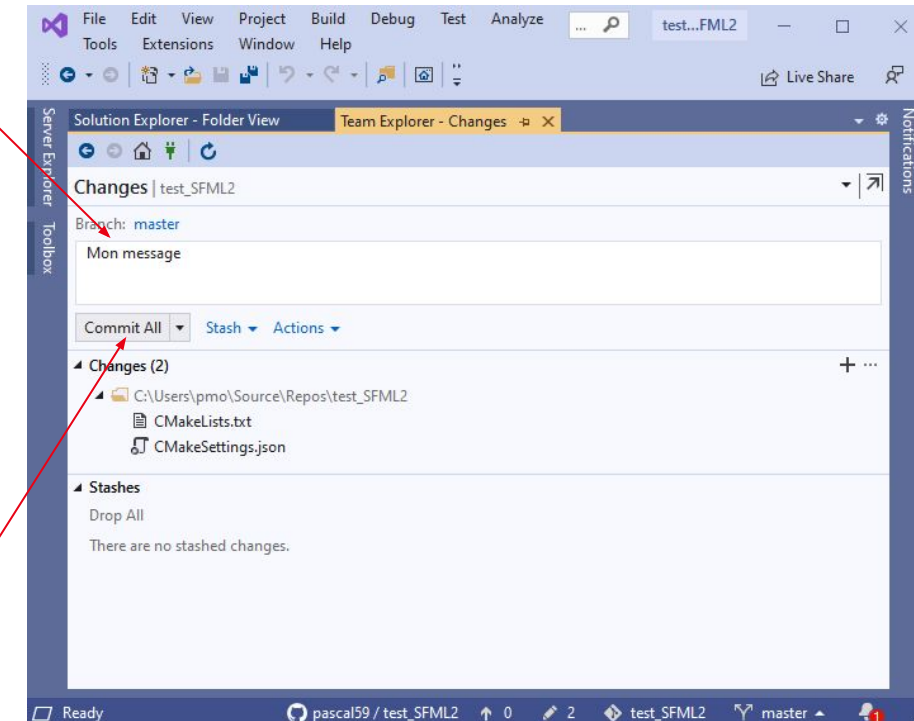
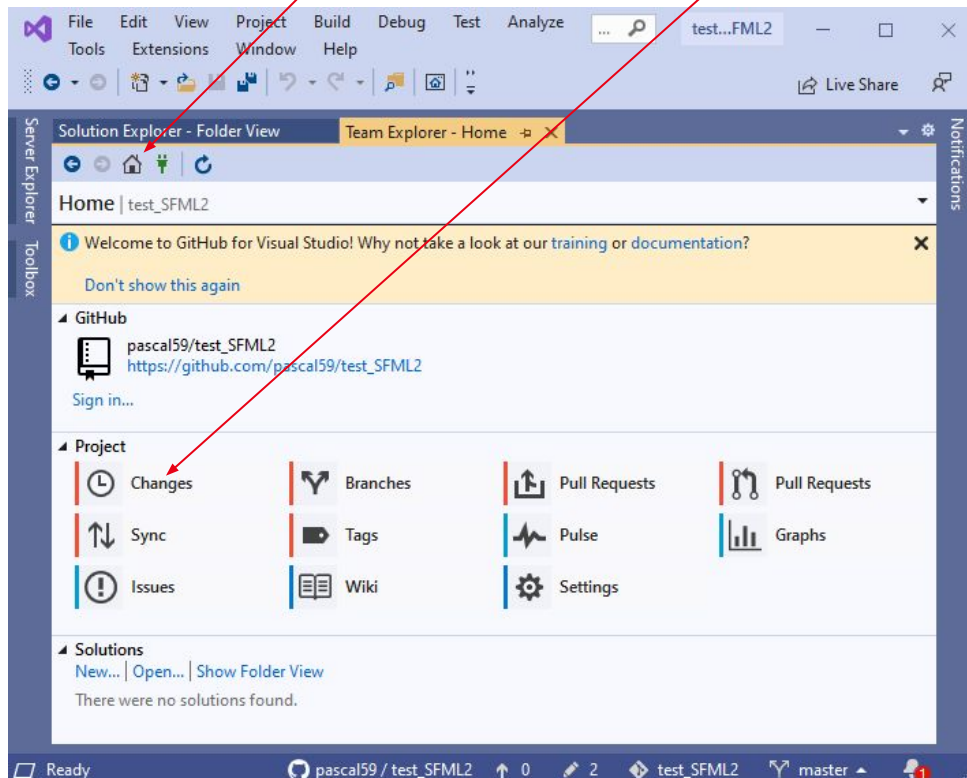

Compilation/utilisation de Git sous Visual C++

Afficher Team explorer (View/Team explorer)

Cliquer sur “home” et ensuite “changes”

Message obligatoire

Commit & push



<https://www.sfml-dev.org/>

“SFML provides a simple interface to the various components of your PC, to ease the development of games and multimedia applications. It is composed of five modules: system, window, graphics, audio and network.”

Bibliothèque relativement simple à mettre en oeuvre, peut être utilisé pour vos visualisations graphiques

Installation :

Windows :

Télécharger l'archive SFML-2.5.1_windows_VS15.6_static.zip
pour simplifier l'utilisation des fichiers cmake placer le répertoire sous :
c:\SFML-2.5.1_windows_VS15.6_static

Linux Ubuntu ou distribution dérivée de Debian:

apt-get install libsfml-dev

MacOS :

Voir : <https://www.sfml-dev.org/tutorials/2.5/start-osx.php>

```
cmake_minimum_required (VERSION 3.9)
project(SFML_Test2)
set(CompilerFlags
    CMAKE_CXX_FLAGS
    CMAKE_CXX_FLAGS_DEBUG
    CMAKE_CXX_FLAGS_RELEASE
    CMAKE_C_FLAGS
    CMAKE_C_FLAGS_DEBUG
    CMAKE_C_FLAGS_RELEASE
)
if(MSVC)
    foreach(CompilerFlag ${CompilerFlags})
        string(REPLACE "/MD" "/MT" ${CompilerFlag} "${${CompilerFlag}}")
        string(REPLACE "/W3" "/W4" ${CompilerFlag} "${${CompilerFlag}}")
        set(CompilerFlag "${CompilerFlag} /std:c++17")
    endforeach()
    set(SFML_STATIC_LIBRARIES TRUE)
    set(SFML_DIR "C:/SFML-2.5.1_windows_VS15.6_static/lib/cmake/SFML")
elseif (CMAKE_CXX_COMPILER_ID STREQUAL "GNU")
    set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -Wall -std=c++17")
endif()
SET(CMAKE_CONFIGURATION_TYPES "Debug;Release" CACHE STRING "Configs" FORCE)
find_package(SFML 2.5 COMPONENTS graphics window system REQUIRED)
set(TEST_SRCS "main.cpp")
add_executable (SFML_Test2 ${TEST_SRCS})
target_link_libraries(SFML_Test2 sfml-graphics sfml-window)
```

“Flags” de compilation par défaut

CMakeLists.txt du projet SFML_test2

Modification des directives de compilation
pour le compilateur inclus dans Visual C++

Configuration pour l'utilisation de la
bibliothèque SFML sous Windows

Modification des directives de compilation
pour le compilateur gcc (Unix)

Force les seules configurations : Debug et
Release

Trouve et charge la configuration de la
bibliothèque SFML

- Dans votre projet utilisez les concepts de la programmation orientée objet (POO) et le plus possible les algorithmes et les “containers” STL.
- Le rendu du projet contiendra:
 - Le code source
 - Un mini manuel d'utilisation
 - Des fichiers exemples à utiliser pour les tests
- La date limite sera communiqué ultérieurement.
- Les dernières séances seront dédiées à la démonstration de l'outil pour l'évaluation du projet.
- Le projet est personnel mais rien n'empêche de vous entraider.

Questions/réponses sur le projet

- *Comment définir le parcours? Avec des points (coordonnées en 3D)?*
- *Pour génération aléatoire, est ce qu'il faut créer un programme parallèlement ou l'inclure dans le programme principal ?*
- *Que veut dire Parcours existant sur internet ?*

Il est possible de trouver des traces gpx (ou autre format) de marathon sur l'internet (faire par exemple une recherche avec "marathon trace gpx"). A partir de ces fichiers, il est possible d'en tirer des informations comme la distance, l'altitude, les pentes/dénivelés, les trajectoires ... en utilisant un site comme [gpsvisualizer](https://www.gpsvisualizer.com/).

Dans le cas d'une génération aléatoire du parcours, je pense préférable d'utiliser un programme annexe qui sauvegardera le parcours dans un fichier qui sera ensuite lu par le programme de modélisation. Au besoin, il pourra être modifié s'il est trop compliqué/absurde.

- *Est-il possible d'inclure des paramètres/critères supplémentaires (sprint, pluie..)*

Le projet est assez ouvert et il est possible d'y ajouter des paramètres/critères supplémentaires. Notez qu'en conséquence, ces ajouts peuvent complexifier le projet. Le mieux, tout comme toute adaptation du projet, est d'en discuter avec vos moniteurs de TP

- *Doit-on simuler le parcours en temps réel?*

La simulation doit se faire en temps réel. Par exemple, les calculs peuvent s'effectuer suivant des pas de temps prédéfinis suffisamment courts pour capturer les changements de contextes qui peuvent se produire durant toute l'épreuve.