

TP1 : Création d'un simulateur

Objectifs:	Créer la structure d'un simulateur et sa signature, construire et compiler pour une utilisation avec le moteur
Pré-requis:	TP0

Les développements s'appuient sur la suite de compilation/construction/test/packaging CMake. CMake est disponible sur http://www.cmake.org .

1 Créer un simulateur

Au fil de cette partie, pour mieux comprendre le fonctionnement d'un simulateur vous pouvez vous référer à la documentation : https://www.openfluid-project.org/resources/docs/manuals/ en/openfluid/2.2.0/main/html/dev srccode.html

Il existe 2 façons de créer le code source d'un simulateur:

- Ecrire le code source "à la main", à partir d'un code vierge : long, fastidieux, source d'erreurs, ...
- utiliser l'environnement de développement OpenFLUID : facile, assisté, intégré à l'environnement OpenFLUID, ...

Nous allons utiliser cette 2ème possibilité pour cet exercice.

1.1 Lancement de l'environnement de développement OpenFLUID

L'environnement de développement OpenFLUID est disponible sous deux formes:

- intégré dans l'interface OpenFLUID-Builder,
- en tant qu'outil indépendant : OpenFLUID-DevStudio.

Dans un premier temps, nous utiliserons l'interface d'OpenFLUID-DevStudio puis dans la suite des TP l'interface intégrée à OpenFLUID-Builder. Lancer OpenFLUID-DevStudio à partir de l'iĉone sur votre bureau ou à partir de la liste des programmes de l'ordinateur.



1.2 Création d'un nouveau simulateur

Nous allons tout d'abord générer le code source "vide" d'un simulateur. Dans OpenFLUID-DevStudio, cliquer sur l'icône "Nouveau ware" dans la barre latérale gauche.

Dans la fenêtre qui s'ouvre, dans *ID du Ware* indiquer le nom du simulateur training.signal.prod. Dans *Nom de la classe C++* indiquer SignalSimulator comme nom de classe C++ qui contiendra le simulateur. Désactiver la case à cocher d'édition de signature et cliquer sur OK.

Nouveau ware		
Créer un nouveau simulateur		
ID du ware training.signal.prod		
Sources		
Nom de la classe C++ SignalSimulator		
Générer les fichier de l'interface de paramétrage		
Nom de la classe C++ d'interface de paramétrage ParamsUiWidge		
✓ Open signature editor		
	⊗ Cancel	Ок

Vous devriez obtenir un dossier training.signal.prod contenant 3 dossiers et 3 fichiers :

- Le dossier *doc* : Permet de contenir la documentation du simulateur,
- Le dossier *src* : Contient le fichier source principal du simulateur WareMain.cpp et est prévu pour contenir tous les fichiers de code,
- Le dossier *tests* : Permet l'ajout de tests pour le simulateur.

Chaque dossier (ainsi que le dossier du simulateur lui-même) contient un fichier CMakeLists.txt qui correspond au système de construction du dossier

- Le fichier openfluid-ware.json : Contient la signature du simulateur (À ne pas modifier directement).
- Le fichier README.md : Fichier (facultatif) donnant des informations générales sur simulateur

Fichier Edition Construire Espace de travail Fer	être <u>A</u> ide	
Si Ob Extensi	training.signal.prod 🔇	
Nouveau ware	Signature Nouveau fichier Ouvrir un fichier Enre	egistrer le fichier Enregistrer sous Rechercher/Remplacer
Ouvrir un ware Ouvrir un ware C MakeLists C MakeLists.txt Importer README.md	<pre>1 /** 2 @file WareMain.cpp 3 */ 4 5 6 #include <openfluid pluggablesimulator.hpp="" ware=""> 7 8 9 /**</openfluid></pre>	
Options de build	<pre>10 11 */ 12 class Simulator : public openfluid::ware::Pluggable! 13 { 14 private: 15 16 17 public:</pre>	Simulator
Manuel	<pre>18 ' 19 ' 20 Simulator(): PluggableSimulator() 21 { 22 23 23 24 }</pre>	Ŧ
	CMakeLists.txt WareMain.cpp Status du ware Messages Avertissements détectés Pas de nom complet fourni dans la signature de ware Pas d'auteur défini dans la signature de ware	
tout replier	 Pas de contact fourni dans la signature de ware Pas de licence définie dans la signature de ware Pas de description dans la signature de ware 	
Espace de travail courant : /tmp/workspace		release installation job unique

1.3 Mise à jour de la signature du simulateur

La signature du simulateur comporte des champs facultatifs qui permettent de renseigner sur la description, le statut, l'auteur... Dans DevStudio, cliquer sur le l'icône Editer la signature dans la barre de menu supérieure du ware. Dans l'onglet *Informations*, remplir le champ *Auteurs* en ajoutant un nom et un email. Vous pouvez également remplir les champs *Nom* et *Description* afin d'indiquer des informations sur le nom complet et la description du simulateur. Nous allons indiquer également que le simulateur fonctionnera avec le pas de temps par défaut de la simulation. Pour cela, aller dans l'onglet *Dynamique* et cocher *La planification utilise le DeltaT par défaut*. Toutes les informations indiquées dans le menu d'édition de signature vont être enregistrées dans le fichier openfluid-ware.json. Pour l'édition de signature, il est recommandé de passer par le menu d'édition et non de modifier le fichier openfluid-ware.json directement.

formations	Paramètres	Attributs	Variables	Dynamique	Tickets
: training.sig	gnal.prod				
Nom :					
Description :					
/ersion :					
État :	stable				*
Licence :					
Auteurs :	Nom		Er	mail	+
					_
Contacts :	Nom		Er	mail	+
					_

2 Configurer, construire et installer le simulateur

Pour configurer le simulateur, cliquer sur l'icône Configurer dans la barre de menu supérieure du ware. Le résultat de la configuration apparaît dans la partie *Messages* et un nouveau dossier build-release-2.2 est apparu dans le dossier du simulateur.

Fichier Edition Construire Espace de travail	Fenêtre <u>A</u> ide
Si Ob Extensi.	. training.signal.prod 😣
Nouveau ware	2 Signature Nouveau fichier Ouvrir un fichier Enregistrer le fichier Enregistrer sous Rechercher/Remplacer
Cuvrir un ware Cuvrir un ware	<pre> a a a b a b a b a b a b a b a b a b a</pre>
Manuel	22 23 24 } CMakeLists.txt WareMain.cpp S Status du ware Messages /usr/bin/cmake /tmp/workspace/wares-dev/simulators/training.signal.prod -DCMAKE_BUILD_TYPE=Release - DOPENFLUID_CURRENT_WORKSPACE_PATH=/tmp/workspace - The C compiler identification is GNU 11.4.0 - The CXX compiler ABI info - Detecting C compiler features - done - Detecting C Compiler ABI info - Detecting C C compiler AB
Espace de travail courant : /tmp/workspace	release installation inbunique

Cette étape ne sera à faire qu'une seule fois.

Cliquer sur l'icône Construire. La construction et l'installation du simulateur s'effectuent. Si des erreurs sont présentes dans le code, celles-ci seront notifiées lors de la construction.

- **Note:** A chaque modification du code du simulateur, il faut sauvegarder les modifications du fichier et construire pour appliquer les modifications.
- **Note:** Il est important d'avoir le mode *installation* activé si l'on souhaite utiliser le simulateur par la suite. Vous trouverez le bouton en bas à droite de l'interface DevStudio

Pour vérifier que le simulateur a été correctement construit et installé, exécuter la commande suivante dans un terminal:

openfluid report simulators --list Le simulateur devrait alors apparaître dans la liste.

Ħ	gerardin@lisah-rdepalma: ~	Q		×
gerardin@lisah-rdepalma:~\$ training.signal.prod gerardin@lisah-rdepalma:~\$	openfluid report simulators1	list		