

# **TP1 : Création d'un simulateur**

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

## 1 Créer un simulateur

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 le plug-in OpenFLUID pour Eclipse : facile, assisté, intégré à l'environnement de développement, ...

Nous allons utiliser cette 3ème possibilité pour cet exercice.

### 1.1 Lancement d'Eclipse

Pour lancer Eclipse, cliquez sur l'icône Eclipse présente sur le Bureau. Au lancement d'Eclipse, il peut vous être demandé de choisir le chemin du workspace que vous souhaitez utiliser. Il est conseillé d'utiliser celui proposé par défaut (sous Linux : /home/openfluid/formation/workspace). Le workspace est le répertoire qui contiendra l'ensemble des projets de simulateurs développés dans cet exercice ainsi que les suivants.

### 1.2 Création du squelette du simulateur

Nous allons utiliser le plug-in OpenFLUID pour Eclipse afin de générer le code source "vide" d'un simulateur. Aller dans le menu d'Eclipse *File > New > Other*.... Dans la fenêtre qui s'ouvre, choisir *OpenFLUID > Simulator project* et cliquer sur *Next*.

😣 New				
Select a wizard				
Wizards:				
type filter text				()
Source Fo CVS CVS CVS COpenFLUIE Simulator Remote System DMA	) project stem Explorer			
<ul> <li>SVN</li> <li>Tasks</li> </ul>				·
?	<back< td=""><td>Next &gt;</td><td>Cancel</td><td>Finish</td></back<>	Next >	Cancel	Finish

La fenêtre suivante est la 1ère étape de la création du simulateur. Cette première étape consiste à paramétrer ce qui va être créé.

Cliquer sur Create new project et indiquer le nom /training.signal.prod. Choisir SignalSim.cpp comme nom de fichier (*Source file name*), SignalSimulator comme nom de classe C++ qui contiendra le simulateur (*Class name*). Laisser coché *Create Cmake build system files* ainsi que *Run Cmake on project creation* Cliquer sur *Next* une fois cette étape complétée.

8	
OpenFLUID simulator	
Sources files and build syst	em
● <u>C</u> reate new project:	[training.signal.prod
○ <u>U</u> se existing container:	/OpenFLUID-project/trainings/2013-06/openfluid-practical_fr
Sources	
Source file name (.cpp):	ignalSim.cpp
<u>C</u> ++ class name: S	ignalSimulator
Create CMake build sys Build system Build subdirectory: _build Install directory: /home	tem files //rabotin/.openfluid/simulators extation.ution.cfm24oc
Kun Chiake on project	u cavan
?	<back next=""> Cancel Finish</back>

La deuxième étape consiste à renseigner les informations sur le simulateur, et notamment son identifiant (*Simulator ID*) par lequel il va être connu. Taper training.signal.prod comme identifiant de simulateur. Les autres champs sont facultatifs, vous pouvez y renseigner un nom plus détaillé du simulateur, votre nom et email. Cliquer sur *Next* une fois cette étape complétée.

OpenFLUID simula Simulator ID and m	tor eta-information
Simulator ID: Simulator name:	training signal prod
Simulator Domain: Description:	
Author's name: Author's email(s):	
1	<back next=""> Cancel Finish</back>

La troisième et dernière étape consiste à renseigner les informations sur les variables/paramètres/données d'entrées/évènements qui sont produits/utilisés/mis à jour par le simulateur.

Phase and a bit and		
Description		
	Ren	nove
	Modfy	Modify Ren

Aller dans l'onglet Scheduling et cocher l'option Scheduling uses the default DeltaT value.

8
OpenFLUID simulator
Data and dynamics
Parameters       Input data       Variables       Events       Extra files       Scheduling         Undefined scheduling       Scheduling uses the default DeltaT value       Scheduling uses a fixed DeltaT value       Scheduling uses a DeltaT value varying from       to
() < Back Next > Cancel Finish

Enfin cliquer sur *Finish*.

Si tout s'est bien passé, vous devriez obtenir un dossier training.signal.prod contenant 3 fichiers dans le *Project Explorer* d'Eclipse :

- SignalSim.cpp : fichier source du simulateur,
- CMakeLists.txt : fichier du système de construction du simulateur (à priori, à ne pas modifier) ,
- CMake.in.config : fichier de configuration de la construction du simulateur.

ainsi qu'un sous-répertoire \_build qui contiendra les résultats de compilation/construction du simulateur.



## 2 Construire et installer le simulateur

La construction/installation du simulateur peut se faire de deux manières :

- Soit en utilisant la ligne de commande en exécutant la commande make install depuis le sous-répertoire \_build
- Soit depuis Eclipse, dans le menu Project > Build Project ou cliquer sur l'icône Marteau de la barre de menu Outils. La méthode depuis Eclipse est à privilégier.



Cette construction/installation est à exécuter à chaque fois que le code source du simulateur est modifié.

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

openfluid -f.

Le simulateur devrait alors apparaître dans la liste.

🙉 🗐 🛛 rabotin@lisah-ddaylewis: ~/000Lisah/1_projets/5_openfluid/9999_workspace/training.signal.prod/_build
Fichier Édition Affichage Rechercher Terminal Aide
Generating done Build files have been written to: /home/rabotin/000Lisah/1_projets/5_openfluid/9999_workspace/training.signal.prod/ build
rabotin@lisah-ddaylewis:~/000Lisah/1_projets/5_openfluid/9999_workspace/training.signal.prod/_build\$ openfluid -f
OpenFLUID v2.0.0~alphaS
software environment for Modelling Fluxes in Landscapes
LISAH, Montpellier, France
Available simulators: - training.signal.prod - examples.primitives.unitsA.up - examples.primitives.unitsA.prod
- examples.road.traffic - examples.trafficlight.state - examples.printives.unitsB.prod
rabotin@lisah-ddaylewis:~/000lisah/1_projets/5_openfluid/9999 workspace/training signal prod/_build\$ []