

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

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.

😣 🖨 💷 OpenFLUID-DevStudio			
I Dia Contraction I I I I I I I I I I I I I I I I I I I	hregistrer sous Configurer C	ontruire	
Simulateurs			
Nom Image: Second Seco			
nobuild.repos			
Observateurs			
Extensions Builder			

1.2 Création d'un nouveau simulateur

Nous allons tout d'abord générer le code source "vide" d'un simulateur. Dans OpenFLUID-DevStudio, aller dans Fichier/ Nouveau ware/ Simulateur.

Dans la fenêtre qui s'ouvre, dans *ID du Ware* indiquer le nom du simulateur training.signal.prod. Dans *Nom du fichier source (.cpp)* indiquer SignalSim.cpp comme nom de fichier, dans *Nom de la classe C++* indiquer SignalSimulator comme nom de classe C++ qui contiendra le simulateur. Laisser les autre options par défaut et cliquer sur OK.

	training.signa	al.prod				
ources						
Nom du f	ichier source (.cpp) SignalSim.cpp				
Nom de la classe C++		e C++ SignalSimulator	SignalSimulator			
Générei	r les fichier de	l'interface de paramétrage				
Nom du f	ichier source (C++ d'interface de paramétrage (.cpp)	ParamsUiWidget.cpp			
	Nom de la c	lasse C++ d'interface de paramétrage	ParamsUiWidget			
Mode de	construction	On - sim2doc must be run manually				
Activer	ctiver l'installation 🧭					
🖉 Créer u	n fichier ware	shub.json				

Vous devriez obtenir un dossier training.signal.prod contenant 4 fichiers :

• 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 (à priori, à ne pas modifier),
- wareshub.json : fichier qui contient les métadonnées du simulateur (non utilisé durant la formation).

😣 🗐 💿 OpenFLUID-DevStudio	
∎ E iii iii Nouveau Ouvrir Enregistrer I	inregistrer sous Configurer Contruire
Simulateurs	training.signal.prod 🗱
Nom I landr.process.saclay J plant.canopy.lai-growth.files-an plant.canopy.lai-growth.files-an pop.plant.surf.application.anses pop.plant.degradation.first.ord pop.plant.foliar-washoff.willis-d pop.plant.foliar-washoff.willis-d pop.plant.foliar-washoff.willis-d pop.plant.surf.application.anses Training.signator SignalSim.cpp water.atm.surf.ringation.files-a water.atm.surf.ringation.files-a water.atm.surf.ringation.files-a water.atm.surf.ringation.files-a water.atm.surf.ringation.files-a water.atm.surf.ringation.files-a water.surf.transfer-su.hayami water.surf.transfer-su.hayami water.surf.transfer-su.hayami water.surf.transfer-su.hayami water.surf.transfer-su.hayami water.ur.surf.transfer-su.hayami water.ur.surf.transfer-su.hayami bar water.ur.surf.surf.watsfar	<pre> #" #" #" #" #" #" #" #" #" #" #" #" #" #" #" #" ########</pre>
Extensions Builder	

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... Remplir le champ DECLARE_AUTHOR en indiquant votre nom et un email. Vous pouvez également remplir les champs DECLARE_NAME et DECLARE_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 à l'aide de l'instruction DECLARE_SCHEDULING_DEFAULT. Pour écrire cette instruction, passer de préférence par le menu contextuel que propose OpenFLUID-DevStudio afin d'éviter les erreurs: pour cela, faire un clic-droit à l'endroit souhaité, puis aller dans OpenFLUID/Signature/Scheduling et sélectionner Declare default deltaT scheduling.

liaceurs	*training.signal.prod 🕱			
Nom I land.atm-surf.topology.firespread I land.surf.representation.geomhydas I land.surf.watershed-delineation.saga I land.rpatch.network I land.rprocess.saclay I land.rcanogy.laiogrowth.files-anses I land.reprocess.laiogrowth.files-anses I land.reprocess.laiograwth.files-anses I land.	<pre>35 DRCLARE VERSION(**); 36 DRCLARE STATUS (openfluid::ware::EX 37 DRCLARE STATUS (openfluid::ware::EX 39 DRCLARE PROCESS(**); 30 DRCLARE METHOD(**); 31 DRCLARE ADTHON(**,**); 33 dr (openfluid::*); 33 dr (openfluid::*); 34 dr (openfluid::*); 33 dr (openfluid::*); 34 dr (openfluid::*); 35 dr (openfluid::*); 36 dr (openfluid::*); 37 dr (openfluid::</pre>	PERIMENTAL);		
printiculty printipularies pop.latt.offlaness pop.latt.offlaness pop.latt.offlaness pop.latt.offlaness pop.latt.offlaness pop.latt.offlaness RemotesystemsTempfiles training signalprod Chalkein config	34 Sm_arr Annuer Ctrl+ 35 Retablir Ctrl+Maj-Z 36 Coller 40 Coller 41 /* Supprimer 42 Supprimer 43 Coller 44 Class Coller	id::ware::PluggableSim	ulator	
CMakeLists.txt	45 (C&C++ 46 priv OpenFLUID	Signature 1	prort signature block	
wareshub.ison	47 Open 2010	Compute code	Seneral	
training.su.prod	49 public:	compace code	tabur .	
water.atm-surf.irrigation.files-anses	50		signific contaut	
water.atm-surf.rain-etp.files-anses	51 52 SignalSimulator(): BluggableSimu	lator()	cientific context	
water.atm-surf.rain-su.files	53 {	10001()	Parameters +	
water.surf-uz.runorr-inrittration.msey	54		/ariables >	
water.surf.transfer-rs.havami	56 }		Accribuces >	
avater.surf.transfer-su.hayami		E	vents +	
👕 water.uz-sz.soil-transfer.watsfar	59 //		spacial structure	
	60 //		xtra riles	
	61		scheduling •	Declare default deltaT scheduling
	CMake.in.config 🗱 *SignalSim.cpp 🗱			Declare rixed scheduling
				Declare scheduling in range

Placer cette instruction avant la fin de la signature signalée par l'appel à END_SIMULATOR_SIGNATURE. La signature de votre simulateur doit ressembler à cela :

```
BEGIN_SIMULATOR_SIGNATURE("training.signal.prod");
DECLARE_NAME("");
DECLARE_DESCRIPTION("Trainingusimulator");
DECLARE_VERSION("");
DECLARE_STATUS(openfluid::ware::EXPERIMENTAL);
DECLARE_DOMAIN("");
DECLARE_PROCESS("");
DECLARE_METHOD("");
DECLARE_AUTHOR("DoeuJ.","doe@foobar.org");
//Scheduling
DECLARE_SCHEDULING_DEFAULT;
END_SIMULATOR_SIGNATURE
```

2 Configurer, construire et installer le simulateur

Pour configurer le simulateur, cliquer sur l'icône Configurer. Le résultat de la configuration apparaît dans la partie *Messages* et un nouveau dossier build-release-2.1 est apparu dans le dossier du simulateur.

OpenFLUID-DevStudio	ам <i>х</i>
Nouveau Ouvrir Enregistrer Enreg	istrer sous Configurer Contruire
Simulateurs	training.signal.prod 🗱
Nom Is Ind at m-surf.topology.firespread Is Ind.surf.representation.geombydas Is Ind.surf.representation.geombydas Is Ind.surf.representation.geombydas Is Ind.patch.network Is Ind.patch.network Is Ind.process.saclay Is Ind.proc	20 DECLARE_FORCES(**); 27 DECLARE_FORCES(**); 28 DECLARE_FORCES(**); 29 DECLARE_FORCES(**); 20 DECLARE_FORCES(**); 20 DECLARE_FORCES(**); 21 DECLARE_FORCES(**); 23 DECLARE_FORCES(**); 24 DECLARE_FORCES(**); 25 DECLARE_FORCES(**); 26 DECLARE_FORCES(**); 27 DECLARE_FORCES(**); 28 DECLARE_FORCES(**); 29 DECLARE_FORCES(**); 20 DECLARE_FORCES(**); 21 DECLARE_FORCES(**); 22 DECLARE_FORCES(**); 23 DECLARE_FORCES(**); 24 DECLARE_FORCES(**); 25 DECLARE_FORCES(**); 26 DECLARE_FORCES(**); 27 DECLARE_FORCES(**); 28 DECLARE_FORCES(**); 29 DECLARE_FORCES(**); 20 DECLARE_FORCES(**); 21 DECLARE_FORCES(**); 22 DECLARE_FORCES(**); 23 DECLARE_FORCES(**); 24 DECLARE_FORCES(**); 25 DECLARE_FORCES(**); 26 DECLARE_FORCES(**); 27 DECLARE_FORCES(**); </th
 Wateratm-sufriain-ecj.rites-anses Watersuf-sufrain-suffiainsuffies Watersuf-sufrainser-sexchange-gu.ba Watersuftransfer-sexchange-gu.ba Watersuftransfer-shayami Watersuftransfer-subayami 	Message: - Performing Test COMPILER, SUPPORTS_CXX11 - - Performing Test COMPILER, SUPPORTS_CXX11 - success - Performing Test COMPILER, SUPPORTS_CXX11 - s
Observateurs	Files SignalSincpp Files SignalSincpp Target: Tealing, signal, prod Files SignalSincpp Target: Tealing, signal, prod Files SignalSincpp Target: Tealing, signal, prod Files SignalSincpp Files S

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

A l'icône Construire, choisir *Construire et installer* et 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

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.

