

---

# **Visualisation de données**

## Devoir de programmation OpenGL – FlowVR

---



**Étudiants**  
Damien GOMBAULT  
Cyril LAVEDRINE

1<sup>er</sup> avril 2009

# Sommaire

<b>1</b>	<b>Description de l'archive</b>	<b>3</b>
<b>2</b>	<b>Installation</b>	<b>3</b>
2.1	Dépendances . . . . .	3
2.2	Compilation . . . . .	3
2.3	Remarques . . . . .	3
<b>3</b>	<b>Utilisation</b>	<b>4</b>
3.1	Lancement . . . . .	4
3.2	Contrôle de la caméra . . . . .	4
<b>4</b>	<b>Implémentation</b>	<b>5</b>
4.1	Description . . . . .	5
4.1.1	Module viewer . . . . .	5
	Classe Camera . . . . .	5
	Classe DTM . . . . .	5
	Classe FlowVR . . . . .	5
	Classe FlowVRThread . . . . .	6
	Classe Light . . . . .	6
	Classe OpenGLScene . . . . .	6
	Classe Point3d . . . . .	6
	Classe Water . . . . .	6
4.1.2	Module flood . . . . .	6
	Classe Flood . . . . .	6
	Classe FlowVR . . . . .	6
4.1.3	Connexions entre modules . . . . .	6
4.2	Déroulement de la simulation . . . . .	7

# 1 Description de l'archive

Voici une description du contenu de l'archive :

- `data/` : ce répertoire contient des exemples de terrain (grilles de hauteurs, textures, et sources d'inondation)
- `doc/` : ce répertoire contient ce rapport et ses sources au format  $\text{\LaTeX}$
- `doxygen/` : ce répertoire contient la documentation du projet au format Doxygen
- `include/` : ce répertoire contient les *headers* nécessaires à la compilation du projet
- `src/` : ce répertoire contient les sources C++ du projet
- `tools/` : ce répertoire contient un outil en Python permettant de découper les grilles de hauteurs
- `make-clean.sh` : script shell permettant de nettoyer les sources du projet
- `make-doxygen.sh` : script shell permettant de générer la documentation Doxygen
- `make-simulation.sh` : script shell permettant de compiler le projet
- `simulation.csv` : fichier de configuration du réseau FlowVR

## 2 Installation

### 2.1 Dépendances

Avant de pouvoir compiler le projet, vous devez d'abord installer **Qt** et **FlowVR Suite** (ainsi que leurs dépendances).

Le projet a été testé uniquement sur **Debian squeeze/sid** avec les versions suivantes des logiciels :

- FlowVR Suite 1.5
- Qt 4.5 (avec les modules **QtCore**, **QtGui** et **QtOpenGL**)
- cmake 2.6

### 2.2 Compilation

Le projet s'installe de la même façon que les autres applications FlowVR. Le script shell `make-simulation.sh` situé à la racine permet de compiler le projet.

### 2.3 Remarques

N'oubliez pas de configurer les variables d'environnement nécessaires au bon fonctionnement de FlowVR et du projet. Pour cela, il faut charger les scripts `flowvr-config.sh` (fourni avec FlowVR) et `simulation-config.sh` (créé pendant la compilation du projet) dans votre environnement en modifiant par exemple les fichiers `~/.bashrc` et `~/.bash_profile`.

Si vous souhaitez faire fonctionner le projet en réseau, il est conseillé d'utiliser les mêmes noms d'utilisateur et d'installer le projet dans la même arborescence sur toutes les machines.

Pensez également à générer les clefs SSH, vérifier les connexions, autoriser l'accès distant à X (commande `xhost +`), vérifier les noms des machines (fichier `/etc/hosts`) et à configurer l'application (fichier `simulation.csv`).

Pour de plus amples informations sur le fonctionnement et la configuration de FlowVR, référez vous à son manuel d'utilisation<sup>1</sup>.

---

<sup>1</sup>Manuel d'utilisation de FlowVR

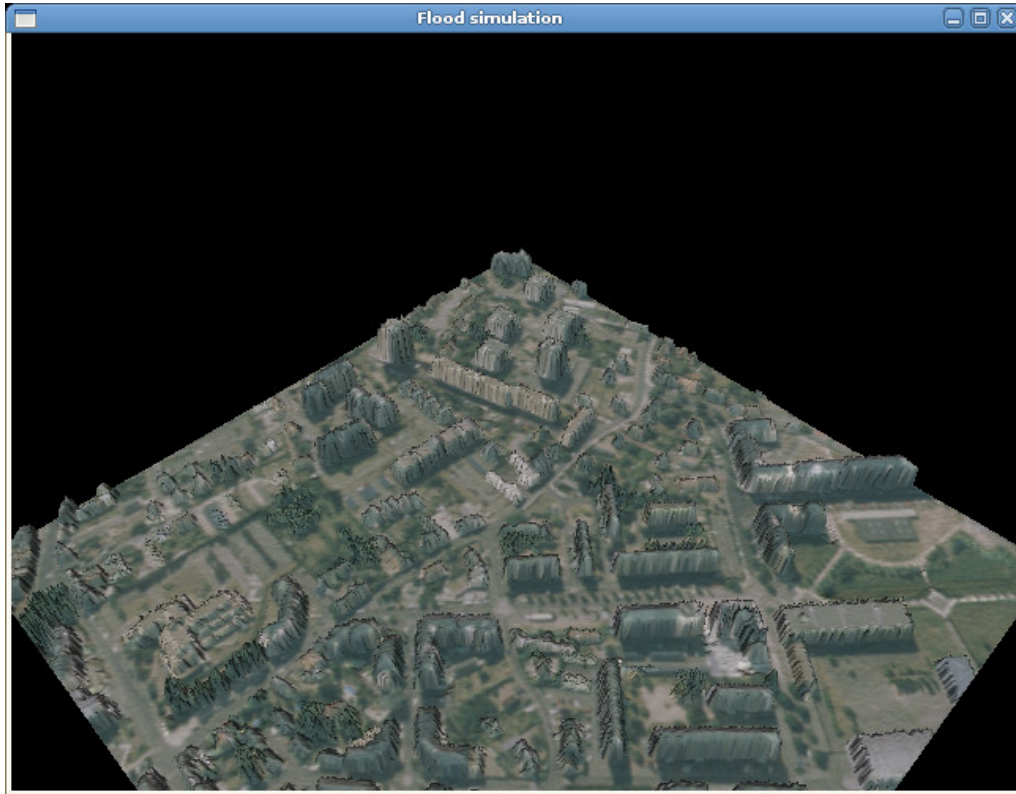
<http://www-id.imag.fr/FLOWVR/manual/flowvr/flowvr-manual.pdf>

## 3 Utilisation

### 3.1 Lancement

Le projet se lance de la même façon que les autres applications FlowVR. Lancez tout d'abord le démon flowvrd puis lancez la simulation avec la commande `flowvr -x -l simulation`.












Une fenêtre de visualisation du terrain et de l'inondation devrait s'ouvrir :



Par défaut, une grille de taille  $512 \times 512$  est chargée. Si votre carte graphique n'est pas assez puissante (principalement au niveau de la quantité de mémoire<sup>2</sup>), vous pouvez choisir une grille plus petite. Pour cela, vous devez modifier les fichiers `include/simulation/components/metamoduleflood.comp.h` et `include/simulation/components/metamoduleviewer.comp.h` et remplacer les fichiers passés en paramètres aux modules dans CmdLine. Relancez ensuite la compilation du projet.

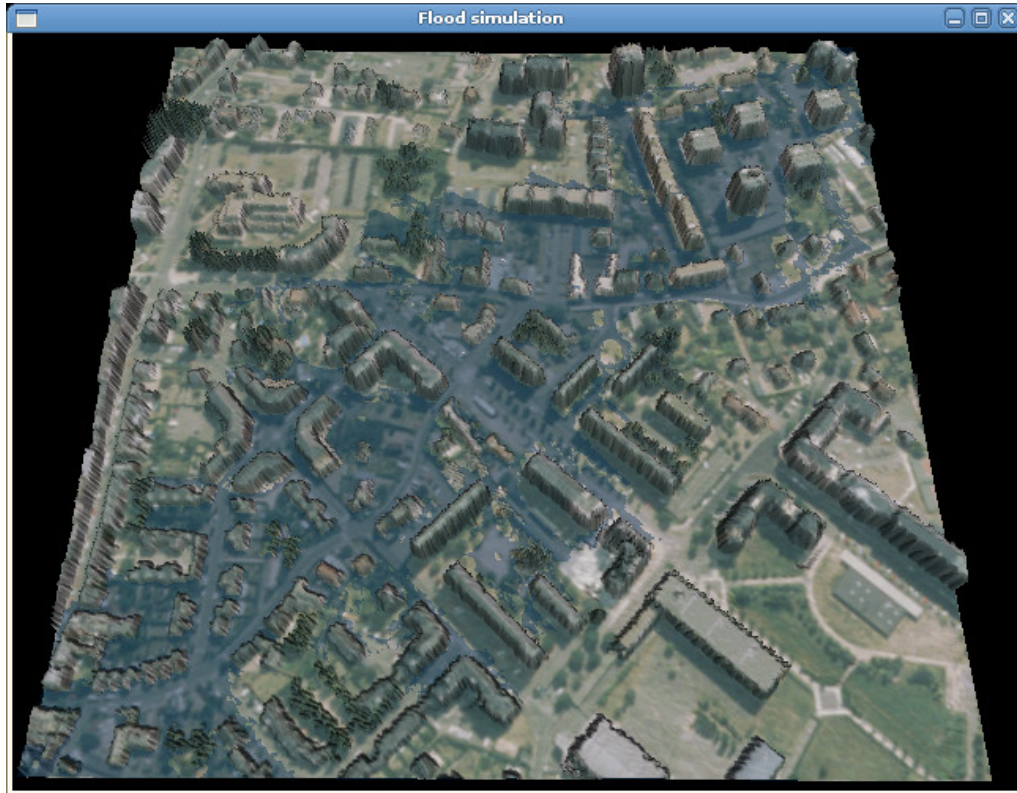
### 3.2 Contrôle de la caméra

Vous pouvez déplacer la caméra en utilisant la souris et le clavier :

- clic gauche enfoncé ou     : tourne la caméra sur elle même
- molette vers le haut ou  : déplace la caméra vers le haut
- molette vers le bas ou  : déplace la caméra vers le bas
-  : déplace la caméra vers l'avant
-  : déplace la caméra vers l'arrière
-  : déplace la caméra vers la gauche
-  : déplace la caméra vers la droite
-  : accélère le déplacement de la caméra

<sup>2</sup>Une carte graphique possédant 256 Mio de mémoire dédiée est requise pour une grille de taille  $512 \times 512$ .

Voici une capture d'écran de la simulation après quelques secondes d'écoulement de l'inondation et déplacement de la caméra :



## 4 Implémentation

### 4.1 Description

Le projet est composé de deux modules FlowVR. Le premier, nommé `viewer`, permet la visualisation du terrain et de l'inondation dans une scène OpenGL. Le second, nommé `flood`, permet de calculer l'écoulement de l'inondation.

#### 4.1.1 Module `viewer`

Ce module de visualisation est divisé en plusieurs classes. Voici leurs descriptions.

**Classe `Camera`** Cette classe implémente le déplacement de la caméra dans la scène OpenGL. Il s'agit d'une caméra de type vol libre.

**Classe `DTM`** Cette classe implémente le terrain. Elle permet de charger le terrain à partir d'un fichier `grd` et initialise ensuite les différentes structures de données nécessaires (sommets, index, normales et coordonnées de textures) pour l'affichage du terrain à l'écran. Pour obtenir de bonnes performances, cette classe utilise des *vertex buffer objects* statiques pour stocker les données directement dans la mémoire vidéo.

**Classe `FlowVR`** Cette classe permet l'initialisation des ports de ce module pour les communications FlowVR. Deux ports sont ainsi créés : `dtmOut` pour l'envoi du terrain au module d'inondation et `waterIn` pour la réception des données de l'inondation.

**Classe FlowVRThread** Les communications de ce module sont déléguées à un *thread*. Cette classe permet ainsi l'envoi du terrain au module d'inondation et la réception des données de l'inondation. L'utilisation d'un *thread* dédié aux communications permet d'éviter de figer l'interface graphique quand le module est en attente d'un message.

**Classe Light** Cette classe implémente la gestion de la lumière de la scène OpenGL.

**Classe OpenGLScene** Il s'agit de la classe principale de ce module. Elle implémente l'interface graphique et gère les événements du clavier et de la souris. Cette classe initialise également les autres éléments de ce module.

**Classe Point3d** Cette classe implémente un point dans l'espace. Elle permet d'effectuer diverses opérations (addition, multiplication, ...).

**Classe Water** Cette classe implémente l'eau. À l'instar de la classe DTM, elle s'occupe d'initialiser les structures nécessaires à l'affichage de l'eau à l'écran. Pour obtenir de bonnes performances, cette classe utilise des *vertex buffer objects* dynamiques. Lorsque cette classe reçoit un signal `updated()` de la classe FlowVRThread, elle met à jour les VBO avec les nouvelles valeurs reçues par le module d'inondation.

#### 4.1.2 Module flood

Ce module dédié au calcul de l'écoulement de l'inondation est divisé en plusieurs classes. Voici leurs descriptions.

**Classe Flood** Cette classe implémente l'écoulement de l'inondation. Elle permet de charger les sources de l'inondation à partir d'un fichier `water`. L'écoulement est ensuite mis à jour à intervalles réguliers, tous les  $\frac{1}{10}$  seconde. Une fois le calcul terminé, le niveau d'eau des cases où il y a une source est augmenté. Puis un message contenant les nouvelles valeurs est envoyé au module de visualisation. La méthode d'écoulement implémentée est similaire à celle présentée dans l'énoncé du devoir.

**Classe FlowVR** Cette classe permet l'initialisation des ports de ce module pour les communications FlowVR. Deux ports sont ainsi créés : `waterOut` pour l'envoi des données de l'inondation au module de visualisation et `dtmIn` pour la réception du terrain.

#### 4.1.3 Connexions entre modules

Les modules sont reliés par deux connexions synchrones.

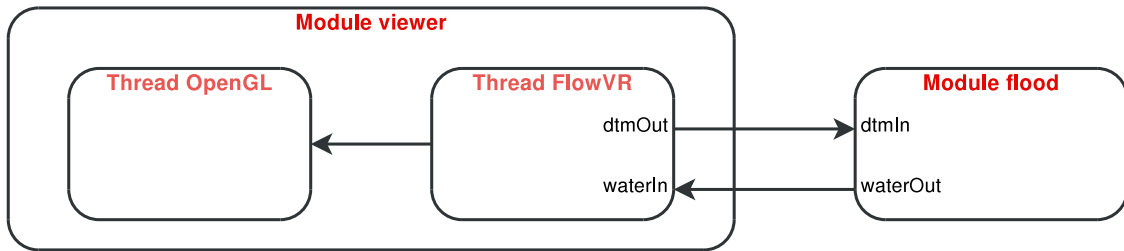
La première connexion relie le port de sortie `dtmOut` du module de visualisation au port d'entrée `dtmIn` du module d'inondation. Cette connexion est utilisée une seule fois, lors de l'initialisation du module d'inondation, afin de recevoir les données du terrain.

La seconde connexion relie le port de sortie `waterOut` du module d'inondation au port d'entrée `waterIn` du module de visualisation. Cette connexion est utilisée à chaque itération de mise à jour de l'écoulement de l'inondation.

## 4.2 Déroulement de la simulation

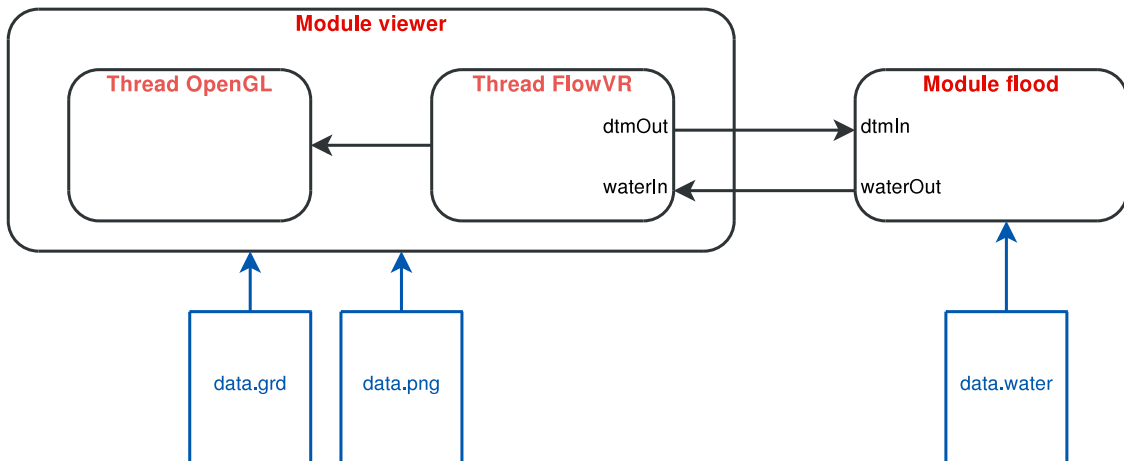
Voici le déroulement du fonctionnement de l'application.

### Étape n°1



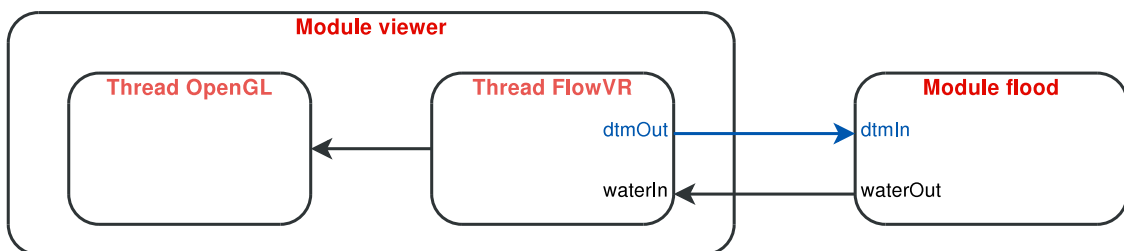
Les deux modules FlowVR se lancent et les ports de communication sont initialisés.

### Étape n°2



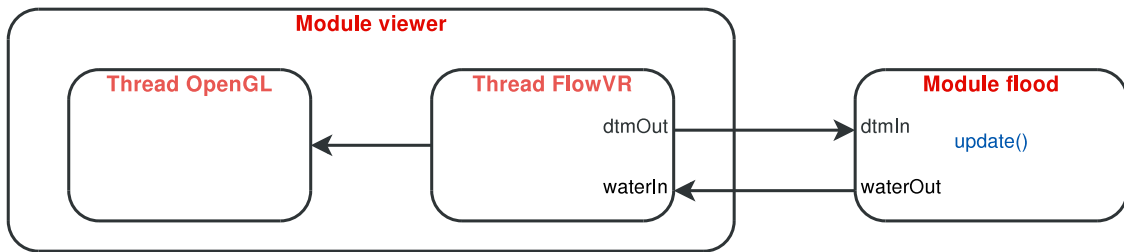
Le module de visualisation charge le terrain à partir du fichier grd contenant les hauteurs. Il charge également la texture à partir du fichier png. Le module d'inondation charge les sources de l'inondation à partir du fichier water.

### Étape n°3



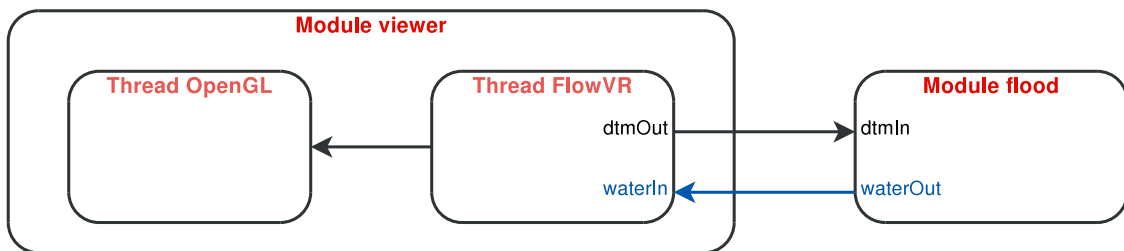
Le module de visualisation envoie les données du terrain au module d'inondation. Le nombre de lignes et de colonnes du terrain sont d'abord envoyées. Ensuite, les hauteurs sont transmises sous la forme d'un tableau de flottants. Les données sortent du port dtmOut du module de visualisation et entrent par le port dtmIn du module d'inondation.

#### Étape n°4



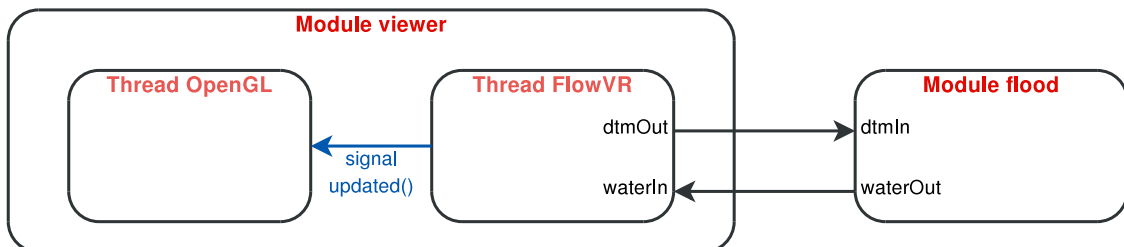
Le module d'inondation possède toutes les informations nécessaires pour commencer à calculer les niveaux de l'eau. Il effectue donc une itération de mise à jour des niveaux.

#### Étape n°5



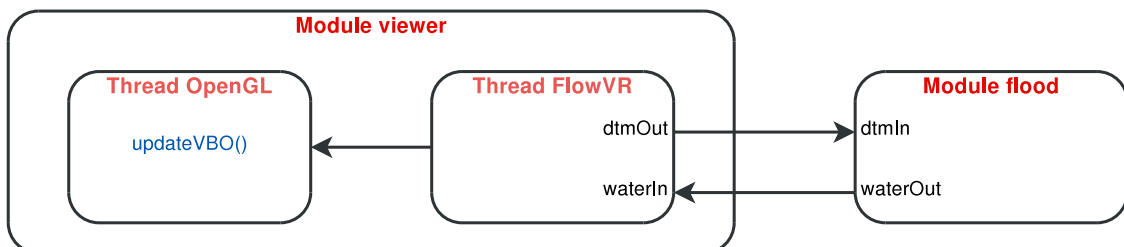
Une fois les nouveaux niveaux calculés, il envoie les nouvelles valeurs au module de visualisation. Les données sortent du port `waterOut` du module d'inondation et entrent par le port `waterIn` du module de visualisation. Les données sont envoyées sous la forme d'un tableau de flottants.

#### Étape n°6



Une fois toutes les données des niveaux reçues, le *thread* chargé des communications FlowVR envoie un signal au *thread* principal, chargé de l'affichage, pour lui indiquer que les données ont été mises à jour.

#### Étape n°7



Le *thread* principal chargé de l'affichage peut ensuite mettre à jour les *vertex buffer objects* pour l'affichage de l'eau.

On recommence ensuite à partir de l'étape n°4. Pendant toutes ces étapes, le module de visualisation affiche de façon totalement indépendante le terrain et l'eau et ne se préoccupe ainsi pas de l'inondation. Ceci permet d'avoir un affichage fluide de la simulation.