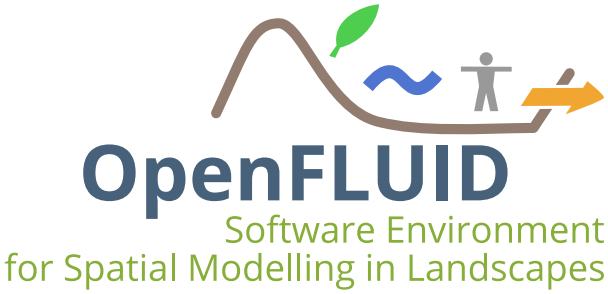

TP5 : Utilisation d'attributs spatiaux

Objectifs:	Utiliser des attributs spatiaux en tenant compte des conditions requises/utilisées
Pré-requis:	TP3,TP4

Fonctionnalités:	`DECLARE_USED_ATTRIBUTE()`
	`OPENFLUID_IsAttributeExist()`
	`OPENFLUID_GetAttribute()`

Nous allons (encore) améliorer le simulateur `training.su.prod` créé lors des TP3 et 4, en ajoutant le coefficient de rétention S comme attribut distribué dans le domaine spatial, et pris en compte dans le simulateur. Ceci permet de donner une valeur à ce coefficient de manière indépendante pour chaque SU. Une fois cette modification apportée, le simulateur pourra utiliser, par ordre de priorité, l'attribut distribué, le paramètre de simulateur ou la valeur par défaut.

1 Nouveau projet OpenFLUID

Afin de repartir du TP précédent, créer un projet OpenFLUID nommé TP5 et y importer le jeu de données d'entrée du TP4.

2 Code source

Les modifications du code source vont porter sur la signature et sur la méthode `runStep()`.

2.1 Signature

Nous allons déclarer la prise en compte d'un attribut distribué facultatif pour le coefficient de rétention (nommé `S`). Cette déclaration se fait au travers de l'instruction `DECLARE_USED_ATTRIBUTE`.

Une fois complétée, la signature devrait être similaire à:

```
BEGIN_SIMULATOR_SIGNATURE("training.su.prod")

  DECLARE_NAME("");
  DECLARE_DESCRIPTION("");

  DECLARE_VERSION("13.05");
  DECLARE_STATUS(openfluid::ware::EXPERIMENTAL);

  DECLARE_DOMAIN("");
  DECLARE_PROCESS("");
  DECLARE_METHOD("");
  DECLARE_AUTHOR("","");

  DECLARE_USED_PARAMETER("S","","-");
  DECLARE_REQUIRED_VARIABLE("water.atm-surf.H.rain","SU","rainfall␣height␣on␣the␣SU","m");

  DECLARE_PRODUCED_VARIABLE("water.surf.H.runoff","SU","water␣runoff␣height␣on␣surface␣of␣SU","m");
  DECLARE_PRODUCED_VARIABLE("water.surf.H.infiltration","SU",
                       "water␣infiltration␣height␣through␣the␣surface␣of␣SU","m");

  // declaration de l'attribut spatial facultatif
  DECLARE_USED_ATTRIBUTE("S","SU","","-");


  // Scheduling
  DECLARE_SCHEDULING_DEFAULT;

END_SIMULATOR_SIGNATURE
```

2.2 runStep()

La modification à apporter dans le `runStep()` concerne la prise en compte de l'attribut spatial `S` uniquement s'il existe. La présence de cet attribut spatial peut être testé au travers de l'instruction `OPENFLUID_IsAttributeExist`. La valeur de l'attribut spatial peut être récupéré au travers de l'instruction `OPENFLUID_GetAttribute`.

Une fois complétée, la méthode runStep() devrait être similaire à:

```
openfluid::base::SchedulingRequest runStep()
{
  openfluid::core::SpatialUnit* pSU;
  openfluid::core::DoubleValue RainValue;
  openfluid::core::DoubleValue RunoffValue;
  openfluid::core::DoubleValue InfiltrationValue;
  double S;


  OPENFLUID_UNITS_ORDERED_LOOP("SU",pSU)
  {

    S = m_S; // Coeff. de retention recoit la valeur par defaut

    // recuperation de la valeur du signal de pluie
    OPENFLUID_GetVariable(pSU,"water.atm-surf.H.rain",RainValue);

    // recuperation du S par SU s'il existe
    if (OPENFLUID_IsAttributeExist(pSU,"S"))
    {
      OPENFLUID_GetAttribute(pSU,"S",S);
    }
```

```
    // calcul du ruissellement selon la methode SCS
    RunoffValue = 0.0;
    if (RainValue > (0.2*S))
    {
      RunoffValue = std::pow(RainValue-(0.2*S),2)/(RainValue+(0.8*S));
    }
    // calcul de l'infiltration par deduction
    InfiltrationValue = RainValue-RunoffValue;

    // production de l'infiltration et du ruissellement
    OPENFLUID_AppendVariable(pSU,"water.surf.H.infiltration",InfiltrationValue);
    OPENFLUID_AppendVariable(pSU,"water.surf.H.runoff",RunoffValue);
  }
  return DefaultDeltaT();
}
```

3 Simulation

Dans le jeu de données d'entrée, l'attribut distribué S doit être présent pour chaque SU pour être prise en compte. Nous allons donc l'ajouter dans le jeu de données fourni.

3.1 ... avec l'interface OpenFLUID-Builder

Dans l'onglet *Domaine spatial*, sélectionner la classe d'unité `SU`. Aller dans l'onglet *Attributs* et cliquer sur le bouton d'ajout d'un attribut (+). Rajouter un attribut nommé `S` à la classe, et donner une valeur comprise entre 0.000002 et 0.0008. Cliquer sur OK.
Si nécessaire, modifier ensuite la valeur par défaut par des valeurs différentes pour chaque unité, en double cliquant directement dans la cellule de la valeur à modifier.

Lancer la simulation et visualiser les résultats dans le fichier pdf créé.

3.2 ... en ligne de commande

Une fois complété, le fichier `domain.fluidx` devrait être structuré comme suit pour la partie sur les attributs des SU:

```
<?xml version="1.0" standalone="yes"?>
<openfluid>
 <domain>
  <attributes unitsclass="SU" colorder="S;area;flowdist;slope">
1       0.0008  5427.69 41.6    0.02172
2       0.00007 16529.9 49.2    0.0001
3       0.0008  3085.65 26.7    0.01789
4       0.0008  3556.82 51.4    0.0192
5       0.0008  5167.89 52.3    0.02631
6       0.00002 7144.49 85.4    0.11577
7       0.00007 7291.46 38.6    0.16371
8       0.00007 8656.04 41      0.04661
9       0.0008  8867.9  80.6    0.11799
10      0.0008  2559.26 44.1    0.20639
11      0.0008  10598.6 43.4    0.05499
12      0.0008  2558.58 31.1    0.0651
13      0.0008  3335.56 51.7    0.01797
```

```
14       0.0008   2235.47 21.7     0.02281
15       0.0008   5231.5  40.5     0.06828
  </attributes>
 </domain>
</openfluid>
```

La commande à exécuter est donc :

```
openfluid run <Bureau>/formation/projects/TP5 -c
```

(à taper sur une seule ligne)

Si tout s'est bien passé, les résultats de la simulation sont accessibles dans `<Bureau>/formation/projects/TP5/OUT`.