



## Tutoriel : Utilisation et paramétrage des observateurs pour visualiser des résultats de simulation

---

**Objectifs:** Enregistrer des variables de sortie calculées par OpenFLUID dans différents formats de données pour la visualisation

---

**Pré-requis:** TP0 : Découverte et prise en main de la plateforme OpenFLUID

---

Les observateurs OpenFLUID permettent de suivre les avancées d'une simulation. Ils permettent donc de visualiser les résultats d'une simulation (dynamique des variables d'une simulation) sous différents formats. A l'heure actuelle, 5 observateurs sont livrés avec OpenFLUID :

- L'observateur *export.vars.files.csv* permet l'export de variables au format CSV (compatible avec les principaux tableurs),
- l'observateur *export.vars.plot.gnuplot* permet l'export de variables dans un graphique GNUplot, avec possibilité de sortie fichier (PDF),
- l'observateur *export.vars.files.kml-plot* permet l'export de variables au format kmz qui permet de visualiser la dynamique d'une variable sur des graphiques associés à des entités spatiales visualisées dans Google Earth,
- l'observateur *export.vars.files.kml-anim* permet l'export de variables au format kmz qui permet de visualiser la dynamique d'une variable à l'aide du curseur chronologique et animé dans Google Earth,
- l'observateur *export.spatial-graph.files.dot* permet d'exporter le graphe de connexions des unités spatiales d'un domaine au format Graphviz.

### 1 Paramétrage des observateurs

Les observateurs se paramètrent tous selon le même principe. Une fois choisi l'observateur désiré (identifié par son ID), l'utilisateur indique pour chaque paramètre (`param name`), la valeur du paramètre (`value`). Le paramétrage des observateurs se fait entre les balises `monitoring` du fichier `monitoring.fluidx` ou via l'onglet `Monitoring` d'OpenFLUID-Builder.

**Exemple avec deux observateurs paramétrés :**

```

<?xml version="1.0" standalone="yes"?>
<openfluid>
  <monitoring>

    <observer ID="example.observer_1">
      <param name="nom_du_parametre_1" value="la_valeur_du_parametre_1"/>
      <param name="nom_du_parametre_2" value="la_valeur_du_parametre_2"/>
    </observer>
    <observer ID="example.observer_2">
      <param name="nom_d'un_autre_parametre_1" value="la_valeur_du_parametre_1"/>
      <param name="nom_d'un_autre_parametre_2" value="la_valeur_du_parametre_2"/>
    </observer>

  </monitoring>
</openfluid>

```

## 2 Observateur export.vars.files.csv

Cet observateur exporte les résultats de la simulation au format CSV. Le format CSV représente les données sous format tabulaire et est compatible avec la plupart des logiciels de tableur.

Au moins un format doit être défini à l'aide des paramètres suivants :

- format.<formatname>.date : Format de la date avec le standard de format de date du langage C,
- format.<formatname>.commentchar : caractère pour les lignes de commentaires,
- format.<formatname>.header : type d'en-tête de fichier,
- format.<formatname>.precision : précision des nombres décimaux.

### Exemple du paramétrage du format f1 :

```

<observer ID="export.vars.files.csv">
  <param name="format.f1.date" value="%Y%m%d%H%M%S"/>
  <param name="format.f1.commentchar" value="#" />
  <param name="format.f1.header" value="colnames-as-comment"/>
  <param name="format.f1.precision" value="8" />
  .....
</observer>

```

A chaque format est associé un jeu de données d'export défini avec les paramètres suivants :

- set.<setname>.unitclass : classe d'unité du set (jeu de sorties),
- set.<setname>.unitsIDs : identifiant(s) de(s) (l') unité(s) du set (séparés par un point virgule ";"). Utiliser \* pour inclure toutes les unités de la classe,
- set.<setname>.vars : la(es) variable(s) incluse(s) dans le set, séparée(s) par des point-virgule(s). Utiliser \* pour inclure toutes les variables,
- set.<setname>.format : le <formatname> utilisé, défini par les paramètres de format.

### Exemple du paramétrage d'un jeu de données associé au format f1 :

```

<observer ID="export.vars.files.csv">
  .....
  <param name="set.SU.format" value="f1"/>
  <param name="set.SU.unitclass" value="SU"/>
  <param name="set.SU.unitsIDs" value="1;6;14"/>
  <param name="set.SU.vars" value="water.surf.Q.downstream-su"/>
</observer>

```

Ainsi cet observateur complet est constitué de :

```
<observer ID="export.vars.files.csv">
  <param name="format.fl.date" value="%Y_%m_%d_%H_%M_%S"/>
  <param name="format.fl.commentchar" value="#" />
  <param name="format.fl.header" value="colnames-as-comment"/>
  <param name="format.fl.precision" value="8" />
  <param name="set.SU.format" value="f1"/>
  <param name="set.SU.unitclass" value="SU"/>
  <param name="set.SU.unitsIDs" value="1;6;14"/>
  <param name="set.SU.vars" value="water.surf.Q.downstream-su"/>
</observer>
```

Pour exporter des variables d'une autre classe d'unités à cet observateur, il suffit de définir un nouveau jeu de données associé à ce format. Exemple avec un ajout de l'export de toutes les variables de toutes les unités RS :

```
<observer ID="export.vars.files.csv">
  <param name="format.fl.date" value="%Y_%m_%d_%H_%M_%S"/>
  <param name="format.fl.commentchar" value="#" />
  <param name="format.fl.header" value="colnames-as-comment"/>
  <param name="format.fl.precision" value="8" />
  <param name="set.SU.format" value="f1"/>
  <param name="set.SU.unitclass" value="SU"/>
  <param name="set.SU.unitsIDs" value="1;6;14"/>
  <param name="set.SU.vars" value="water.surf.Q.downstream-su"/>
  <param name="set.RS.format" value="f1"/>
  <param name="set.RS.unitclass" value="RS"/>
  <param name="set.RS.unitsIDs" value="*/>
  <param name="set.RS.vars" value="*/>
</observer>
```

### 3 Observateur export.vars.plot.gnuplot

Cet observateur permet d'exporter les résultats de simulation sous forme de graphes au format gnuplot, qui peuvent être transformés en document (notamment en PDF). Il associe des séries de valeurs à un ou plusieurs graphes, trois catégories de paramètres sont à déclarer.

Les paramètres pour déclarer les séries à utiliser :

- serie.<seriename>.var : le nom de la variable à tracer,
- serie.<seriename>.unitclass : le nom de la classe d'unité,
- serie.<seriename>.unitid : identifiant de l'unité,
- serie.<seriename>.sourcefile : le fichier de données existantes à tracer (ex : measured\_data.dat), si cette série n'est pas basée sur des variables de la simulation,
- serie.<seriename>.style : le style GNUplot à utiliser pour dessiner la série (ex : linespoint),
- serie.<seriename>.label : l'étiquette à utiliser pour cette série à la place d'une étiquette automatique.

```
<observer ID="export.vars.plot.gnuplot">
  <param name="serie.s1.var" value="tests.double.dot" />
  <param name="serie.s1.unitclass" value="TestUnits" />
  <param name="serie.s1.unitID" value="1" />
  <param name="serie.s1.label" value="a_double" />
  <param name="serie.s1.style" value="line" />
  .....
</observer>
```

Les paramètres pour déclarer les graphes :

- graph.<graphname>.series : une liste de <seriename> séparée par des point-virgules (ex : s1;s2),
- graph.<graphname>.title : le titre du graphe à la place du titre automatique,
- graph.<graphname>.key : la présence et le format de la légende (off, default, outside),
- graph.<graphname>.ylabel : l'étiquette de l'axe y du graphe.

```
<observer ID="export.vars.plot.gnuplot">
  .....
  <param name="graph.g1.series" value="s1" />
  <param name="graph.g1.key" value="default" />
  <param name="graph.g1.ylabel" value="values_(xx.yy-1)" />
  <param name="graph.g1.title" value="un_graphe" />
</observer>
```

Les paramètres globaux à déclarer sont :

- terminal : le mode de sortie du terminal en remplacement du mode par défaut wxt (ex : pdfcairo size 11.7,8.3),
- output : le nom du fichier de sortie quand export du graphe dans un fichier (e.g. my-graph.pdf),
- tryopengnuplot : valeur de 1 pour lancer GNUplot à la fin de la simulation,
- persistent : valeur de 1 pour garder GNUplot ouvert après la fin de la simulation.

```
<observer ID="export.vars.plot.gnuplot">
  .....
  <param name="terminal" value="pdfcairo_size_11.7,8.3" />
  <param name="output" value="test.pdf" />
  <param name="persistent" value="0" />
  <param name="tryopengnuplot" value="1" />
</observer>
```

Ainsi cet observateur complet est constitué de :

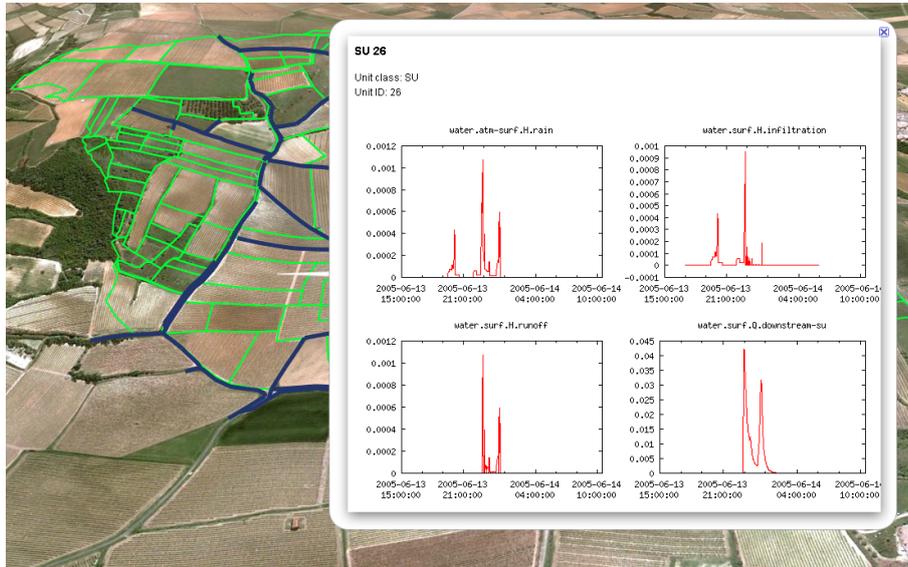
```
<observer ID="export.vars.plot.gnuplot">
  <param name="serie.s1.var" value="tests.double.dt" />
  <param name="serie.s1.unitclass" value="TestUnits" />
  <param name="serie.s1.unitID" value="1" />
  <param name="serie.s1.label" value="a_double" />
  <param name="serie.s1.style" value="line" />
  <param name="graph.g1.series" value="s1" />
  <param name="graph.g1.key" value="default" />
  <param name="graph.g1.ylabel" value="values_(xx.yy-1)" />
  <param name="graph.g1.title" value="un_graphe" />
  <param name="terminal" value="pdfcairo_size_11.7,8.3" />
  <param name="output" value="test.pdf" />
  <param name="persistent" value="0" />
  <param name="tryopengnuplot" value="1" />
</observer>
```

## 4 Observateur vars.files.kml-plot

Google Earth permet de visualiser des données spatiales à partir de fichiers au format KMZ. L'observateur *vars.files.kml-plot* crée un fichier kmz qui permet de visualiser la dynamique des variables sur des graphiques. Il récupère les variables produites, va créer les graphiques des dynamiques des variables qui seront associées aux données spatiales sous format KMZ.

**Attention:** Les données spatialisées doivent impérativement être au format shapefile et projetées avec le système de projection WGS84.

Lors de l'exécution de la simulation, le fichier kmz sera créé dans le répertoire des sorties de simulation. Ce fichier est lisible par Google Earth.



L'observateur possède trois paramètres globaux :

- kmzfilename : nom du fichier créé,
- title : titre du fichier kml,
- tryopengearth : 1 pour lancer Google Earth une fois la simulation terminée, 0 sinon.

```
<observer ID="export.vars.files.kml-plot" >
  <param name="kmzfilename" value="myfile.kmz" />
  <param name="title" value="test_of_the_kml_plot_export" />
  <param name="tryopengearth" value="0" />
  .....
</observer>
```

Il faut ensuite définir les paramètres pour chaque couche que l'on désire afficher :

- unitclass : classe de l'entité à afficher,
- sourcetype : type de source de la donnée (ex : file),
- sourcefile : nom du fichier de données spatiales,
- varlist : liste des variables à afficher,
- defaultcolor : couleur des entités sans graphe,
- plottedcolor : couleur des entités avec des graphes,
- linewidth : largeur des lignes.

```
<observer ID="export.vars.files.kml-plot" >
  .....
  <param name="layers.1.unitclass" value="SU" />
  <param name="layers.1.sourcetype" value="file" />
  <param name="layers.1.sourcefile" value="data/extractroujan_su_wgs84.shp" />
  <param name="layers.1.varlist" value="*" />
  <param name="layers.1.defaultcolor" value="ff316A23" />
```

```

    <param name="layers.1.plottedcolor" value="ff33ff00" />
    <param name="layers.1.linewidth" value="2" />

</observer>

```

Ainsi cet observateur complet est constitué de :

```

<observer ID="export.vars.files.kml-plot" >
  <param name="kmzfilename" value="myfile.kmz" />
  <param name="title" value="test_of_the_kml_plot_export" />
  <param name="tryopengearth" value="0" />

  <param name="layers.1.unitclass" value="SU" />
  <param name="layers.1.sourcetype" value="file" />
  <param name="layers.1.sourcefile" value="data/extractroujan_su_wgs84.shp" />
  <param name="layers.1.varslist" value="*" />
  <param name="layers.1.defaultcolor" value="ff316A23" />
  <param name="layers.1.plottedcolor" value="ff33ff00" />
  <param name="layers.1.linewidth" value="2" />

</observer>

```

Si on veut ajouter des graphiques pour une autre classe d'entités, il suffit de rajouter une nouvelle couche à afficher :

```

<observer ID="export.vars.files.kml-plot" >
  <param name="kmzfilename" value="myfile.kmz" />
  <param name="title" value="test_of_the_kml_plot_export" />
  <param name="tryopengearth" value="0" />

  <param name="layers.1.unitclass" value="SU" />
  <param name="layers.1.sourcetype" value="file" />
  <param name="layers.1.sourcefile" value="data/extractroujan_su_wgs84.shp" />
  <param name="layers.1.varslist" value="*" />
  <param name="layers.1.defaultcolor" value="ff316A23" />
  <param name="layers.1.plottedcolor" value="ff33ff00" />
  <param name="layers.1.linewidth" value="2" />

  <param name="layers.2.unitclass" value="RS" />
  <param name="layers.2.sourcetype" value="file" />
  <param name="layers.2.sourcefile" value="data/extractroujan_rs_wgs84.shp" />
  <param name="layers.2.varslist" value="*" />
  <param name="layers.2.defaultcolor" value="ff6a3423" />
  <param name="layers.2.plottedcolor" value="ffff6632" />
  <param name="layers.2.linewidth" value="5" />

</observer>

```

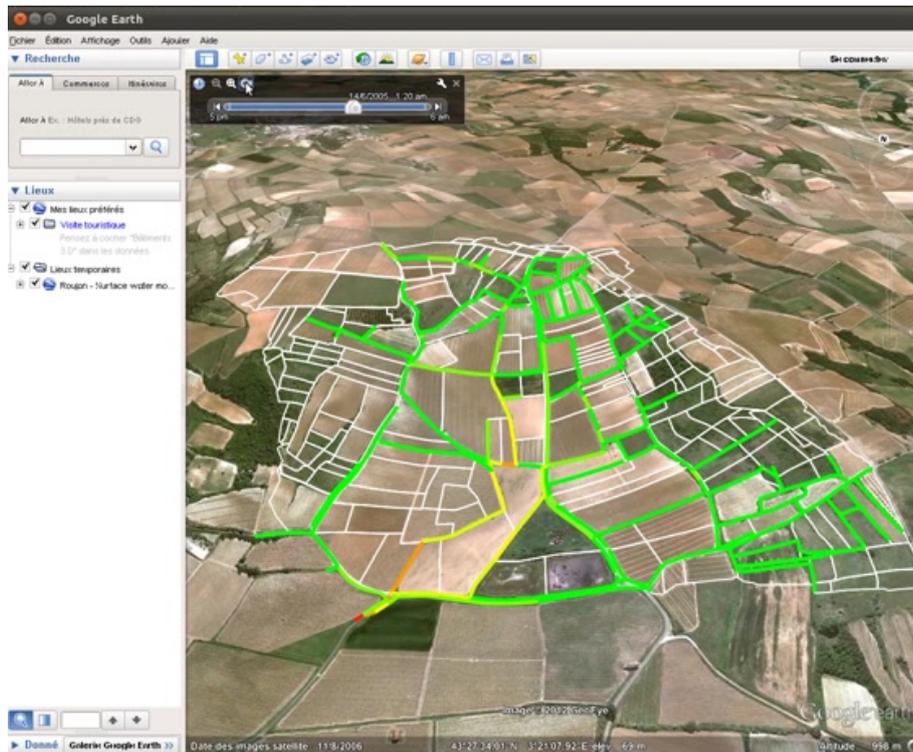
## 5 Observateur vars.files.kml-anim

L'observateur *vars.files.kml-anim* crée un fichier kmz qui permet de visualiser la dynamique d'une variable sous la forme d'une animation, à l'aide du curseur chronologique de Google Earth. Il récupère les variables produites, va créer les graphiques des dynamiques des variables qui seront associées aux données spatiales sous format KMZ.

**Attention:** Les données spatialisées doivent impérativement être au format shapefile et projetées avec le système de projection WGS84.

**Note:** On ne peut animer qu'une seule variable à la fois sur une seule couche mais d'autres couches statiques peuvent également être affichées

Lors de l'exécution de la simulation, le fichier `kmz` sera créé dans le répertoire des sorties de simulation. Ce fichier est lisible par Google Earth. Sous Google Earth, utilisez le curseur chronographique (dont la vitesse est paramétrable à l'aide de sa boîte à outil).



L'observateur possède trois paramètres globaux :

- `kmzfilename` : nom du fichier créé,
- `title` : titre du fichier kml,
- `tryopengearth` : 0 pour ne pas lancer Google Earth une fois la simulation terminée, 1 pour lancer Google Earth.

```
<observer ID="export.vars.files.kml-plot" >
  <param name="kmzfilename" value="myfile.kmz" />
  <param name="title" value="test_of_the_kml_anim_export" />
  <param name="tryopengearth" value="0" />
  . . . . .
</observer>
```

Il faut ensuite définir les paramètres de la couche à animer (`layers.anim`) puis ceux de la(es) couche(s) statique(s) (`layers.static`).

- `layers.anim.unitclass` : classe de l'entité à afficher,
- `layers.anim.varname` : nom de la variable à afficher,
- `layers.anim.sourcetype` : type de source de la donnée (ex : file),
- `layers.anim.sourcefile` : nom du fichier de données spatiales,

- layers.anim.linewidth : largeur des lignes,
- layers.anim.colorscale : échelle de couleur (RGB),
- layers.static.<layername>.unitclass : classe de l'entité à afficher,
- layers.static.<layername>.sourcetype : type de source de la donnée (ex : file),
- layers.static.<layername>.linewidth : largeur des lignes,
- layers.static.<layername>.color : couleur de la couche.

Exemple de la visualisation de la dynamique de la variable `water.surf.H.level-rs` sur les unités RS et affichage d'une couche statique de SU.

```
<observer ID="export.vars.files.kml-anim" >
  .....

  <param name="layers.anim.unitclass" value="RS" />
  <param name="layers.anim.varname" value="water.surf.H.level-rs" />
  <param name="layers.anim.sourcetype" value="file" />
  <param name="layers.anim.sourcefile" value="data/extractroujan_rs_wgs84.shp" />
  <param name="layers.anim.linewidth" value="4" />
  <param name="layers.anim.colorscale" value="ff00ff00;14;ff00ff76;18;ff00ffdc;22;
ff00faff;26;ff0099ff;28;ff001cff"/>

  <param name="layers.static.1.unitclass" value="SU" />
  <param name="layers.static.1.sourcetype" value="file" />
  <param name="layers.static.1.sourcefile" value="data/extractroujan_su_wgs84.shp" />
  <param name="layers.static.1.linewidth" value="3" />
  <param name="layers.static.1.color" value="ffffffff" />
</observer>
```

Ainsi cet observateur complet est constitué de :

```
<observer ID="export.vars.files.kml-anim" >
  <param name="kmzfilename" value="myfile.kmz" />
  <param name="title" value="test_of_the_kml_anim_export" />
  <param name="tryopengearth" value="0" />
  <param name="layers.anim.unitclass" value="RS" />
  <param name="layers.anim.varname" value="water.surf.H.level-rs" />
  <param name="layers.anim.sourcetype" value="file" />
  <param name="layers.anim.sourcefile" value="data/extractroujan_rs_wgs84.shp" />
  <param name="layers.anim.linewidth" value="4" />
  <param name="layers.anim.colorscale" value="ff00ff00;14;ff00ff76;18;ff00ffdc;22;
ff00faff;26;ff0099ff;28;ff001cff"/>

  <param name="layers.static.1.unitclass" value="SU" />
  <param name="layers.static.1.sourcetype" value="file" />
  <param name="layers.static.1.sourcefile" value="data/extractroujan_su_wgs84.shp" />
  <param name="layers.static.1.linewidth" value="3" />
  <param name="layers.static.1.color" value="ffffffff" />
</observer>
```

## 6 Observateur export.spatial-graph.files.dot

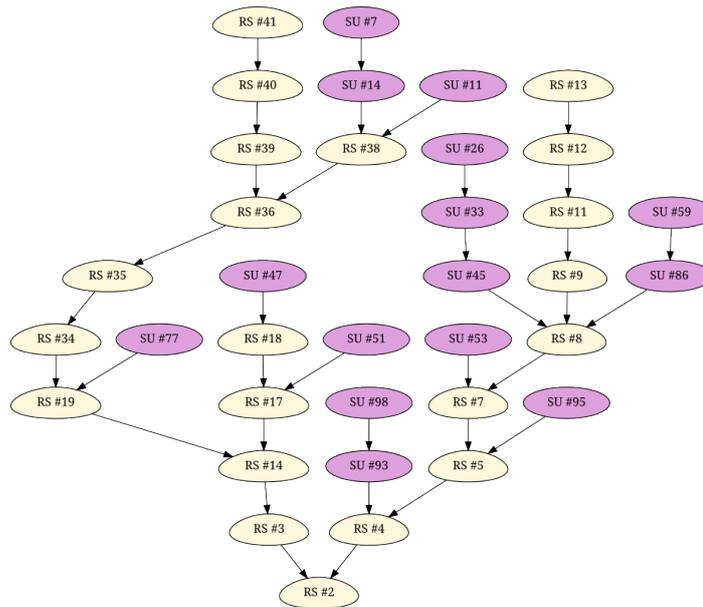
Cet observateur permet d'exporter le graphe des unités spatiales d'un domaine au format Graphviz (fichier dot) à différents instants de la simulation. Il permet ainsi de suivre l'évolution des connexions des entités du domaine spatial dans le cas où un ou des simulateurs feraient évoluer les connexions au cours de la simulation.

Il est ainsi possible d'exporter le graphe :

- au moment de l'initialisation de la simulation : création du fichier `spatial-graph_init.dot`,

- à chaque pas de temps de la simulation : création d'un fichier pour chaque pas de temps `spatial-graph_tXX.dot` où XX est la valeur de chaque pas de temps,
- à la fin de la simulation : création du fichier `spatial-graph_final.dot`.

Lors de l'exécution de la simulation, le(s) fichier(s) dot sera(ont) créé(s) dans le répertoire des sorties de simulation. Ce(s) fichier(s) dot peut(vent) ensuite être transformé(s) en fichier(s) image(s) ou vectoriel(s) à l'aide de l'outil *GraphViz*.



Spatial domain at finalization

L'observateur possède quatre paramètres globaux :

- title : titre du graphe,
- when.init : si mis à 1, le fichier représentant l'état du graphe du domaine spatial à l'initialisation de la simulation sera produit,
- when everytime : si mis à 1, un fichier représentant l'état du graphe à chaque pas de temps de la simulation sera produit,
- when.final : si mis à 1, le fichier représentant l'état du graphe du domaine spatial lors de la finalisation de la simulation sera produit.

Exemple d'un graphe qui sera exporté à l'initialisation de la simulation, puis pour chaque pas de temps et à la finalisation :

```

<observer ID="export.spatial-graph.files.dot" >
  <param name="title" value="Graph_of_the_spatial_domain" />
  <param name="when.init" value="1" />
  <param name="when.everytime" value="1" />
  <param name="when.final" value="1" />
  .....
</observer>

```

Il faut ensuite définir les paramètres de style des entités à visualiser à l'aide du paramètre `style.<unitclass>.<attribute>` où le champ `unitclass` est le nom de la classe d'entités OpenFLUID à représenter et où le champ `attribute` peut prendre les valeurs suivantes :

- `shape` : permet de paramétrer la forme des noeuds du graphe (egg, circle, box...)
- `style` : permet de paramétrer l'apparence des noeuds du graphe (filled, rounded..)
- `fillcolor` : permet de paramétrer la couleur des noeuds du graphe (cornsilk, white...)
- ...

**Note:** Pour plus de précisions sur le paramétrage des styles du format dot, consulter la documentation de la librairie GraphViz <http://www.graphviz.org/Documentation.php> .

Exemple d'un graphe où les unités OpenFLUID de type RS seront représentées sous forme ovoïde et de couleur jaune sable et les unités de type SU avec une forme par défaut et de couleur prune :

```
<observer ID="export.spatial-graph.files.dot" >
    .....

    <param name="style.RS.shape" value="egg" />
    <param name="style.RS.style" value="filled" />
    <param name="style.RS.fillcolor" value="cornsilk" />
    <param name="style.SU.style" value="filled" />
    <param name="style.SU.fillcolor" value="plum" />
</observer>
```

Ainsi cet observateur complet est constitué de :

```
<observer ID="export.spatial-graph.files.dot" >
    <param name="title" value="Graph_of_the_spatial_domain" />
    <param name="when.init" value="1" />
    <param name="when.everytime" value="1" />
    <param name="when.final" value="1" />

    <param name="style.RS.shape" value="egg" />
    <param name="style.RS.style" value="filled" />
    <param name="style.RS.fillcolor" value="cornsilk" />
    <param name="style.SU.style" value="filled" />
    <param name="style.SU.fillcolor" value="plum" />
</observer>
```

Pour transformer un fichier dot en un format plus lisible (pdf, svg, ps...), la commande dot en ligne de commande peut être utilisée :

Exemple pour transformer un fichier dot en format svg :

```
dot -Tsvg spatial-graph_final.dot -o graphe_final.svg
```

Exemple pour transformer un fichier dot en format pdf :

```
dot -Tpdf spatial-graph_final.dot -o graphe_final.pdf
```