Abstract

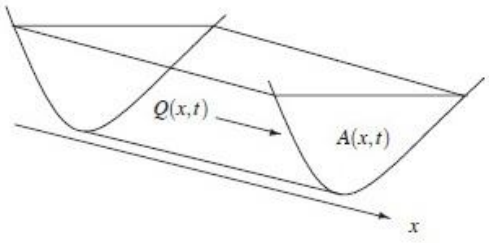
In this report we introduce ODE, then present an extensible Object-Oriented framework – written in C++ – with emphasis on the reusability of modules for ODE solvers. The ability to extend this API to accommodate new algorithms as they are developed is particularly attractive. This facilitates our work to find the best numerical method and speed the development of a dedicated simulator for specific cases.

Mathematical Model

Conservative Form of St-Venant Equations

D'un point de vue de la technique des fluides la propagation d'un front d'onde est un phénomène extrêmement complique impliquant la dynamique du fluide avec une surface libre dans un mouvement turbulent intense sous accélération de la gravite. Quand on essaie de d'écrire mathématiquement cette situation, on doit résoudre les équations de Navier-Stokes 3-D a surface libre. Cependant, sous certaines approximations (St-Venant) on arrive à une représentation mathématique plus simple de la réalité physique et qui trouve de nombreuses applications dans le domaine de l'ingénierie. On obtient donc un ensemble d'équations couplées qui sont un équilibre entre les forces de la pesanteur, de frottement et d'inertie appliquée à la masse d'eau en mouvement. Ces équations décrivent des bilans entre les différents processus: diffusion, convection (transport). Selon que l'un de ces processus domine sur les autres ceci change la nature physique de l'équation. Par exemple, et c'est le cas qui nous intéressé, quand la convection domine on a affaire à un phénomène de propagation.

Les équations unidimensionnelles des écoulements à ciel ouvert sont décrites en termes de la profondeur d'eau et du débit, et l'évolution de ces quantités satisfont aux équations de St-Venant, lesquelles expriment simplement la conservation de la masse et de la quantité de mouvement dans la direction de l'écoulement (appelé système de lois de conservation).



Le système de S-Venant peut être écrit sous différentes formes, nous considérons la forme conservative des équations.

$$\partial_t \boldsymbol{U} + \partial_x \boldsymbol{F}(\boldsymbol{U}) = \boldsymbol{S} \qquad (1)$$

ou

$$U = \begin{pmatrix} A \\ Q \end{pmatrix}$$

$$F = \begin{pmatrix} Q \\ Q^2/A + gI_1 \end{pmatrix}$$

$$S = \begin{pmatrix} 0 \\ gI_2 + gA(S_f - S_b) \end{pmatrix}$$

- **U**: state vector of the state variables, respectively wetted area and discharge of section flow
- **F** : physical flux, convective and pressure
- $\boldsymbol{S_f}$: friction term expressed by the Manning formula
- S_b: bottom slope term expressed as derivative of the bathymetry
- **A**: wetted cross-section area which depends **(x,h(x,t))**
- **g**: acceleration of gravity
- $\boldsymbol{I_1}$: pressure term given by $\int_0^{h(x)} (h(x) - \vartheta)\sigma(x,\vartheta)\, d\vartheta$
- $\boldsymbol{I_2}$: and given by $\int_0^{h(x)} (h(x) - \vartheta) \left[\frac{\partial \sigma(x,\vartheta)}{\partial x}\right] d\vartheta$ section width variation along x

with h water depth, σ width for a fixed depth and ϑ depth integration variable along y axis.

La méthode aux volumes finis peut être considérée comme une méthode aux différences finies appliquée à la forme conservative différentielle des lois de conservation écrites dans un système de coordonnées arbitraires. Ces schémas aux volumes finis approximent la forme intégrale des lois de conservation. À chaque pas en temps, on résout la moyenne intégrale (les variables dépendantes sont approximées par des moyennes intégrales, évolution de la moyenne intégrale) des variables de l'écoulement dans chaque volume.

Il existe plusieurs façons de déduire et de formuler les équations de la mécanique des fluides. En particulier, elles peuvent être déduites à partir du fait que le comportement d'un système physique est complètement déterminé par des lois de conservation. Plus précisément, un certain nombre de propriétés telles que la masse, la quantité de mouvement et l'énergie sont conservées. Ces conditions ne déterminent pas complètement le comportement d'un système physique. Il faut ajouter une loi d'état et, pour que le problème soit bien posé, des conditions initiales et des conditions aux limites.

Le principe de conservation nous donne une expression de la variation d'une quantité à l'intérieur d'un volume, y compris l'effet des forces extérieures (à reformuler). L'écriture de la loi de conservation sous sa forme générale pour une quantité U, quantité scalaire ou vectorielle par unité de volume, agissant sur un volume Ω fixé dans l'espace et borné par la surface S est la suivante:

$$\frac{\partial}{\partial t} \int_{vol} U \, d\Omega + \oint_{surf} \vec{F} \cdot d\vec{S} = \int_{vol} Qv \, d\Omega + \oint_{surf} Qs \, d\vec{S} \qquad (1)$$

pour U scalaire. Si est une quantité vectorielle les termes F et Qs sont des tenseurs et Qv est un vecteur. La méthode des volumes finis utilise directement la forme conservative des équations. L'équation à résoudre est écrite

$$\frac{\partial}{\partial t} \int_{vol} U \, d\Omega + \oint_{surf} \vec{F} \cdot d\vec{S} = \int_{vol} Q \, d\Omega \qquad (2)$$

par le théorème de Gauss, on a:

$$\frac{\partial}{\partial t}\int_{vol} U\, d\Omega + \oint_{surf} \vec{\nabla}\cdot\vec{F}\, d\Omega = \int_{vol} Q\, d\Omega \quad (3)$$

sous forme différentielle

$$\frac{\partial U}{\partial t} + \vec{\nabla}\cdot\vec{F} = Q \quad (4)$$

où est un champ scalaire.

Voici comment on formule un problème aux volumes finis. Le domaine de calcul est partitionné dans un nombre fini de volume de contrôle [$x_{i+1/2}, x_{i-1/2}$] autour de la position nodale x_i . La position de l'interface droite $x_{i+1/2}$ est définie comme ($x_i + x_{i+1}$)/2. L'équation différentielle sous forme conservative est intégrée sur chaque volume de contrôle pour produire l'équivalent discret de la loi de conservation

$$U_i^{n+1} = U_i^n + \frac{dt}{dx}\left[\left(F_{i-1/2} - F_{i+1/2}\right) + Q_i^n\right]$$

avec Q étant le terme source. La différence principale entre plusieurs schémas aux volumes finis est dans la façon d'évaluer (ou approximer) les flux convectifs à l'interface

$$F_{i+1/2} = F\left(U\left(x_{i+1/2}, t\right)\right).$$

Technique par laquelle la formulation intégrale des lois de conservation sont discrétisées directement dans l'espace physique (je ne vois pas très bien ce que ca veut dire?) Reformuler tout cà dans mes propres mots.

Système de simulation: SFX package

Il est bien connu que les environnements de type «framework», application semi-complète sous forme d'abstractions de haut niveau, ont comme principal objectif de réduire de façon dramatique le temps et les efforts requis pour le développement de programmes pour des domaines d'applications particulier. L'avantage principal, le détail des «class» internes qui permet de compléter le «framework» est invisible au reste du programme et donc, les détails de l'implémentation peuvent être modifiés sans affecter le reste de l'application. Le «framework» que nous avons développé, toujours en cours de développement, est un environnement de programmation flexible pour le prototypage rapide de simulateur physique qui est composé d'un ensemble de class C++ sous forme de librairies. Celui-ci offre tous les types de base afin de résoudre numériquement les équations de St-Venant sur le problème du bris de barrage (unidimensionnel) par méthode aux différences finies explicites. On met l'emphase sur la facilité à programmer des algorithmes physiques rapidement en utilisant des composantes déjà développées. Dans les projets de recherche industriel, le physicien est souvent appelé à tester ou à expérimenter différents scenarios, ce type d'environnement répond à ce besoin.

Requis de notre système

- **Ça doit être possible de changer la méthode numérique utilisée**

Nous pouvons avoir plusieurs raisons pour vouloir changer la méthode numérique. Peut-être que celle-ci conduit à des résultats qui sont incorrects. Ceci inclus une solution instable, également des solutions qui sont mathématiquement corrects mais physiquement incorrects, ou des solutions qui ne sont pas assez précises.

- **Ça doit être facile de construire un programme qui résout le problème bris de barrage**

Nous demandons que le système soit extensible. Ce n'est pas suffisant de pouvoir solutionner un ensemble de problèmes prédéfinis avec un ensemble de solutions prédéfinies. Un utilisateur doit être capable de bâtir de nouveaux problèmes, ou des nouvelles méthodes, et d'être capable de les ajouter au système. Ceci est très pratique si, par exemple, l'utilisateur développe un schéma numérique et veut le comparer avec d'autres méthodes.

- **Le système doit être flexible. Les composantes doivent être changeables**

Toutes les composantes de l'algorithme physique (résoudre numériquement les équations ODE) devraient être facilement interchangeables. Particulièrement, facile de changer entre différentes implémentations d'un intégrateur numérique, ou de changer entre l'algorithme du flux numérique. Lors de la validation des schémas numériques, on veut être en mesure d'essayer plusieurs alternatives et ainsi comparer.

- **Ça doit être facile d'étendre le système avec de nouvelles composantes**

Par exemple, on peut vouloir traiter des termes avec des schémas aux différences, d'autres avec des techniques sophistiquées telles des schémas de capture de choc, ou l'on utilise un solveur de Riemann. Au niveau temporel, plusieurs algorithmes d'intégration numérique sont disponibles : Euler, Runge-Kutta ne sont que quelques-uns des algorithmes utilises.

- **Ça doit être possible de permettre plusieurs modèles de s'intégrer dans le système**

Différents modèles peuvent être utilises durant la même simulation et leur résultat compare ou combine. Également possible, par exemple, d'étudier le même aspect de deux scenarios ou deux aspects du même scenario.

Concepts clés de notre application

Énoncé du problème - Résoudre un problème physique par simulation numérique.

Concept Discrétisation Globale (passage du continu au discret). Le concept de discrétisation peut être caractérisé de la manière suivante:

- Le domaine du problème est représenté par un ensemble de sous-domaines, appelé cell. Dans le cas 1-D le domaine [a,b] de l'équation est discrétisée par une série de points x_i, i=0,...,N-1 où la solution "w" de l'équation est discrétisée;
- Sur chacune des cell, le processus physique est approximé par des fonctions de type (polynomiale ou autre), et un système d'équations algébriques reliant les quantités physiques aux différents points, appelé nœuds, de l'élément est alors construit;
- Les équations discrétisées sont solutionnées sur chacune des cellules (cell);

Concept Discrétisation Spatiale

La discrétisation spatiale consiste à construire une maille ou grille sur laquelle le domaine continu est remplacé par un ensemble fini de points où les valeurs numériques des variables seront évaluées.

Concept Discrétisation Temporelle

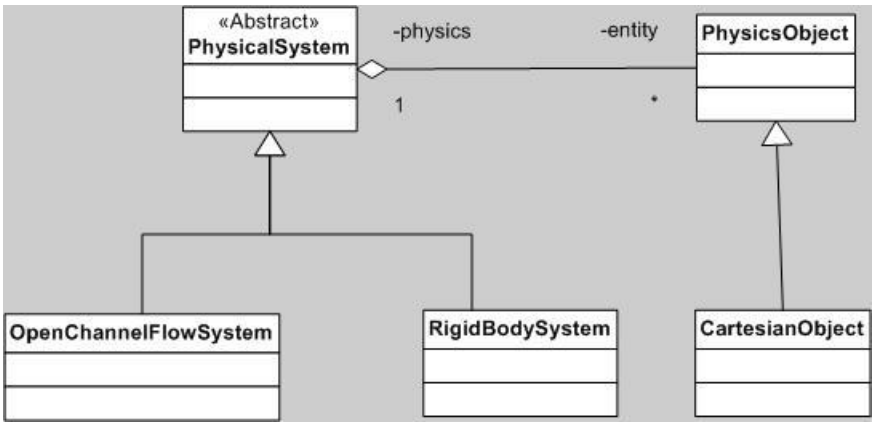
Schéma explicite/implicite. Partie temporelle est discrétisée selon ces deux types de schémas: dans le premier les variables inconnues au pas de temps suivant sont exprimées en fonction des variables du pas temps précèdent. Dans le second, on doit résoudre un système matriciel linéaire/non-linéaire à chaque pas de temps. Variables inconnues au pas de temps courant dépendent des variables au pas de courant.

Concept Représentation discrète

Une fois que la maille est définie les équations peuvent être discrétisées, conduisant à une transformation des équations en un système algébrique qui exprime la relation entre les inconnues et la maille. La base de toutes les méthodes numériques consiste en cette transformation des équations physiques en un système algébrique, linéaire ou non-linéaire.

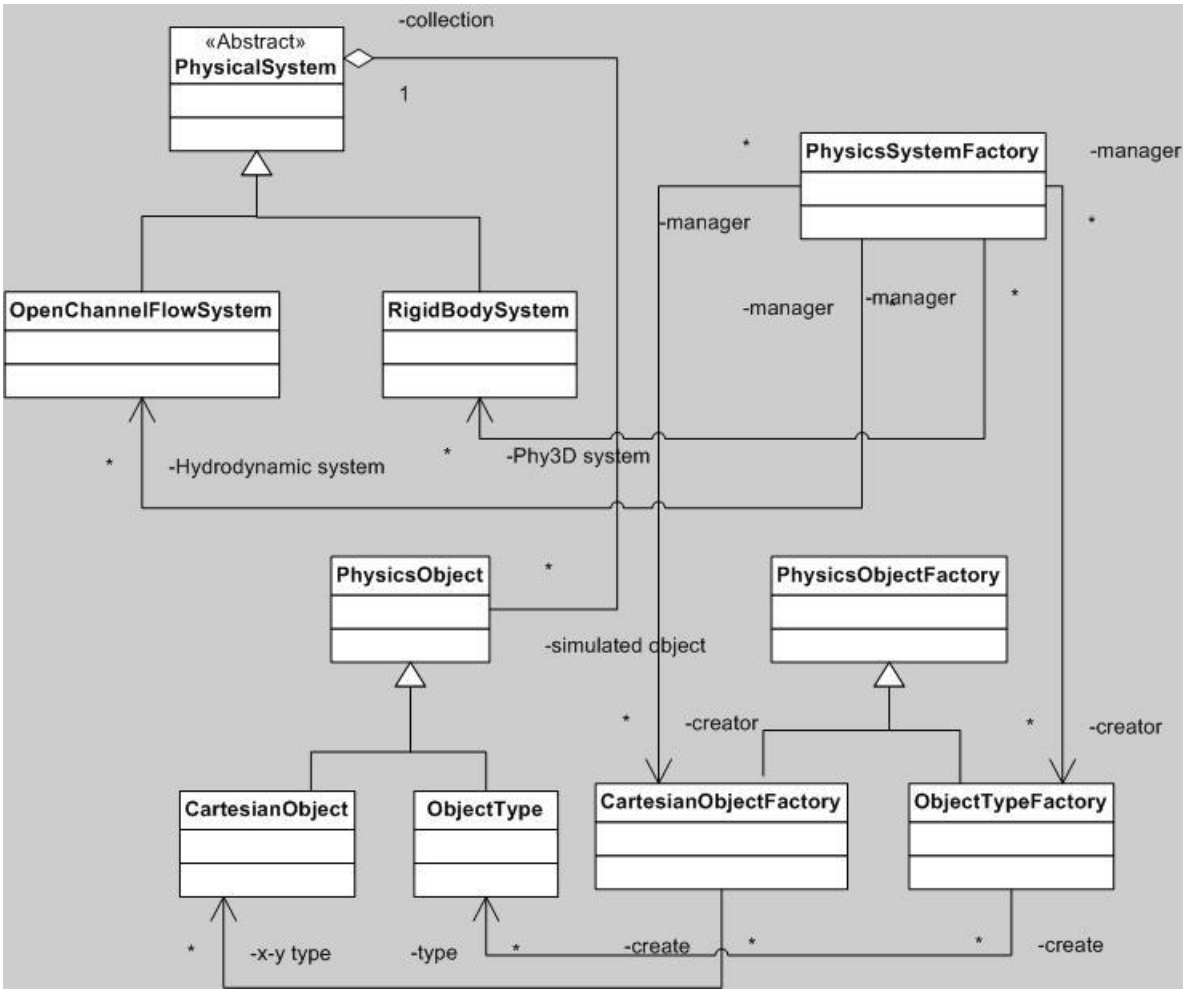
Système Physique

Premièrement nous commençons avec le système physique que nous voulons étudier. Chaque système physique est décrit par des objets physiques qui font parties du système que l'on étudie. Dans la section



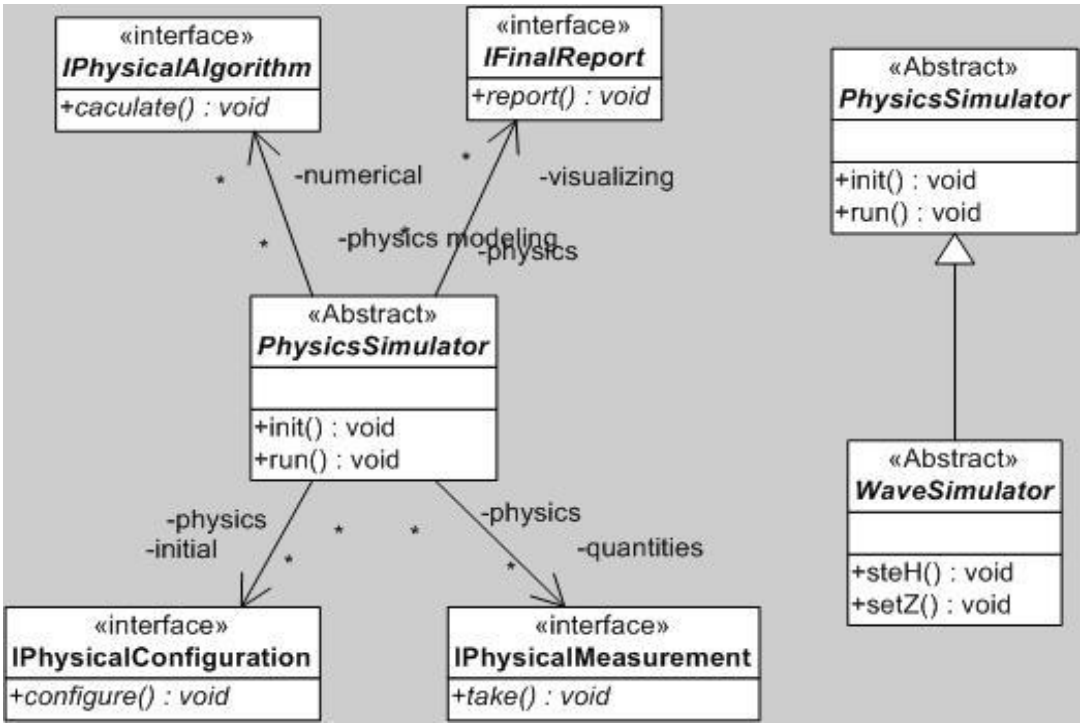
Dans le dessin ci-haut **PhysicalSystem** et **PhysicalObject** sont des class de haut niveau d'abstraction. L'association entre **PhysicalSystem** et **PhysicalObject** est une association de type "has a" ou appelé agrégation comme l'indique le petit losange. En d'autres termes ceci signifie que **PhysicalSystem** est composé de ou est fait de **PhysicalObjects**. Comment est-ce que le « **framework**» utilise ces deux class comme class de base afin d'étendre le système? Il faut noter que dans le diagramme UML (ci-haut) que ces deux class sont des class abstraites ce qui signifie qu'elles ne peuvent être instanciées. Elles ont été construites de cette façon afin d'être en mesure d'utiliser les patrons de conception de type Factory et Abstract Factory. Pour l'instant il faut seulement prendre pour acquis qu'une Factory est une façon standard de créer des objets qui sont similaires mais qui ne sont pas nécessairement reliés entre eux. Une **abstract factory** crée des objets similaires d'une façon standard ou les objets sont reliés entre eux.

Puisqu'il peut y avoir plusieurs types de **PhysicalSystems** dépendamment de la physique que l'on veut simuler, une **factory** est très bien adaptée pour nous assister dans la création de ces objets. De la même manière, pour les **PhysicalObjects** une **abstract factory** est utilisée pour créer ceux-ci. Un diagramme de class UML plus détaillé ci-dessous :



Celles-ci ne sont que quelques-unes des class qui font partie du SFX package. Au niveau le plus élevé de la hiérarchie de ce diagramme est la **PhysicalSystemFactory**. Cette class est une class concrète et est responsable pour la création du **PhysicalSystem** spécifié par l'utilisateur. Ceci se fait par la création d'une instance du **PhysicalSystem** spécifié par l'usager et ajoute à celui-ci le nombre spécifié de **PhysicalObjects** en utilisant la **PhysicalObjectFactory** appropriée.

La prochaine partie est de comprendre la relation entre les autres class du système du simulation package et leur association avec le **PhysicsSimulator** dans le SFX package. Le diagramme UML ci-dessous montre ces relations:



Les class au bout des flèches sont des Interfaces. Une interface spécifie un contrat qui doit être remplie par la class qui l'implémente. Par exemple, l'interface IFinalReport spécifie qu'une méthode du nom de report doit être implémentée et qui prend comme arguments une instance de Simulation et une de PhysicalSystem.

Les quatre interfaces que l'on montre ici sont celles pour lesquelles chaque développeur qui a l'intention d'utiliser le «framework» devra fournir une implémentation.

Le nom de chacune des interfaces parle par lui-même. L'implémentation de la class **IPhysicalConfiguration** est pour la mise en place de la solution initiale. Dans l'exemple d'application les sections d'écoulements sont initialisées avec la hauteur d'eau et la position du barrage. Le **PhysicsSimulator** utilise l'implémentation de la class **IPhysicalAlgorithm** pour intégrer numériquement les équations du mouvement St-Venant. Dans l'exemple d'application un solveur de Runge-Kutta au deuxième ordre est utilisé pour résoudre l'équation semi-discrète. L'implémentation de la class **IPhysicalMeasurement** nous renseigne sur quelles quantités physiques seront mesurés périodiquement. L'implémentation de la class **IFinalReport** nous donne un sommaire sur les statistiques et les résultats finaux qui sont fournis à l'utilisateur sous forme d'un simple format texte.

La dernière class du SFX package est la class **IDataStore**. Le but de ces class (ses dérivées) est de rendre la class **PhysicsSimulator** transparente au reste de l'application sur la façon de sauver les données.

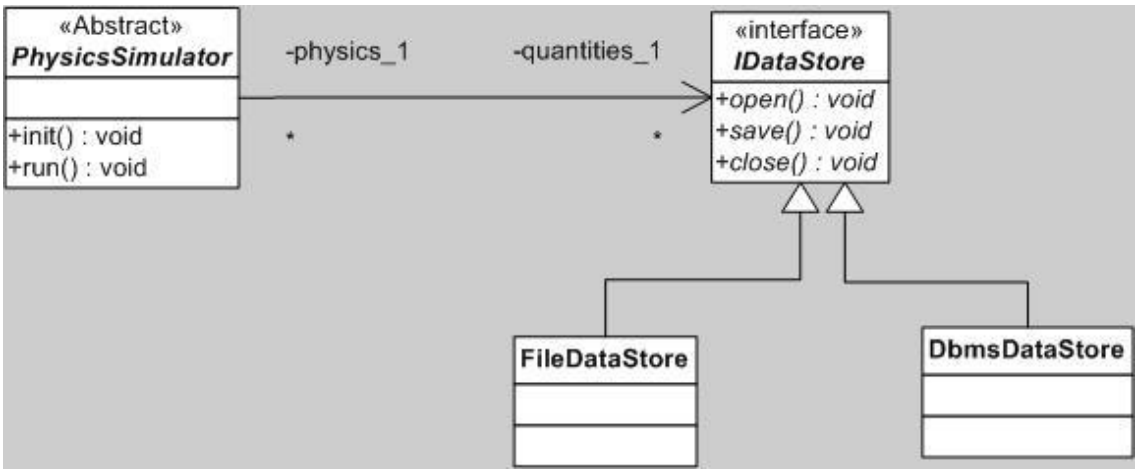


Diagramme Use Case (scénario d'un cas d'utilisation)

Plusieurs choses peuvent nous intéresser durant la simulation. Par exemple être en mesure de sauvegarder et de rapporter la position, la vitesse et l'accélération de l'onde. On devrait aussi être capable de calculer les énergies potentielles et totales et de les rapporter. Finalement, la simulation devrait rapporter la hauteur maximum atteinte par l'onde, le temps de la simulation.

Nous décrivons un scenario possible d'utilisation de la librairie.

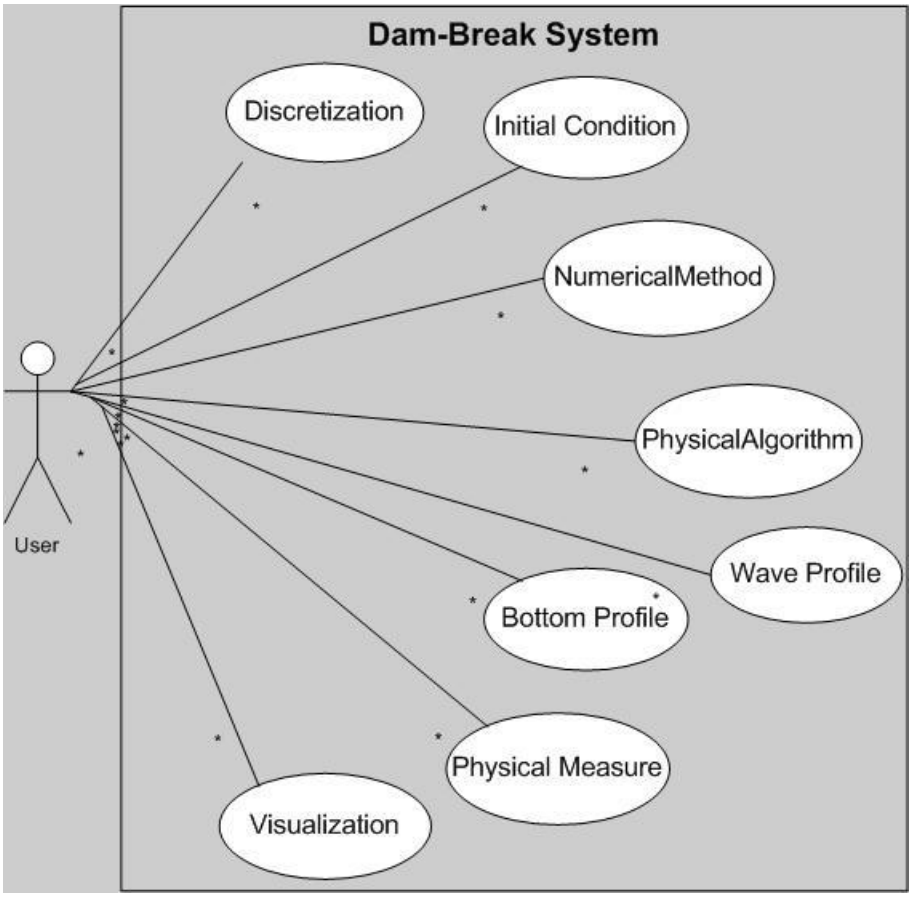


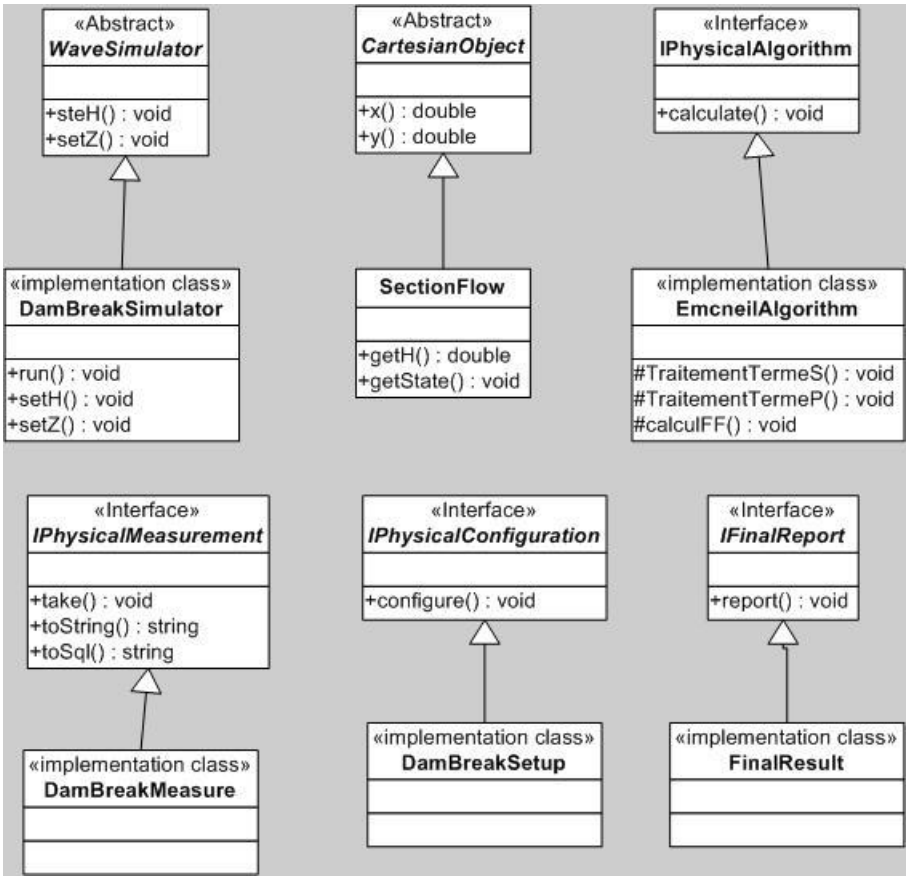
Diagramme de class

Diagramme statique qui modélise les interactions des concepts de notre système. Class représente un concept de l'application (propriété ou attribut avec leur méthode ou message). Le diagramme est un des plus important, modélise les relations entre les concepts (association, agrégation, inheritance, …)

Livre de recettes

Traduit les use case (requis) en modèle de class. Ci-dessous les principales étapes pour développer un simulateur en utilisant le «framework» :

- Créer la class Simulator physique par inheritance de la class de base (PhysicsSimulator);
- Fournir une implémentation de la configuration physique (IPhysicalConfiguration), par exemple les conditions initiales;
- Fournir une implémentation de l'algorithme physique (IPhysicalAlgorithm), traitement numérique des termes de l'équation selon la physique que l'on veut modéliser.
- Fournir une implémentation pour les mesures physiques (IPhysicalMeasurement), les différentes mesures physiques telles que : énergie, vitesse;
- Fournir une implémentation pour la production d'un rapport (IFinalReport), un rapport final afin de rapporter les statistiques, visualiser la solution, etc.…
- Fournir une implémentation pour la sauvegarde des données (ex. mesures physiques), data store;
- Choisir une méthode numérique, ceci consiste à définir un modèle (numérique), par exemple une méthode semi-discrète, schéma Runge-Kutta pour la discrétisation temporelle et des schémas aux différences finies;
- L'algorithme physique, traitement des termes de l'équation physique, par exemple évaluation des flux numériques a l'interface, calcul des termes de friction, de pression;



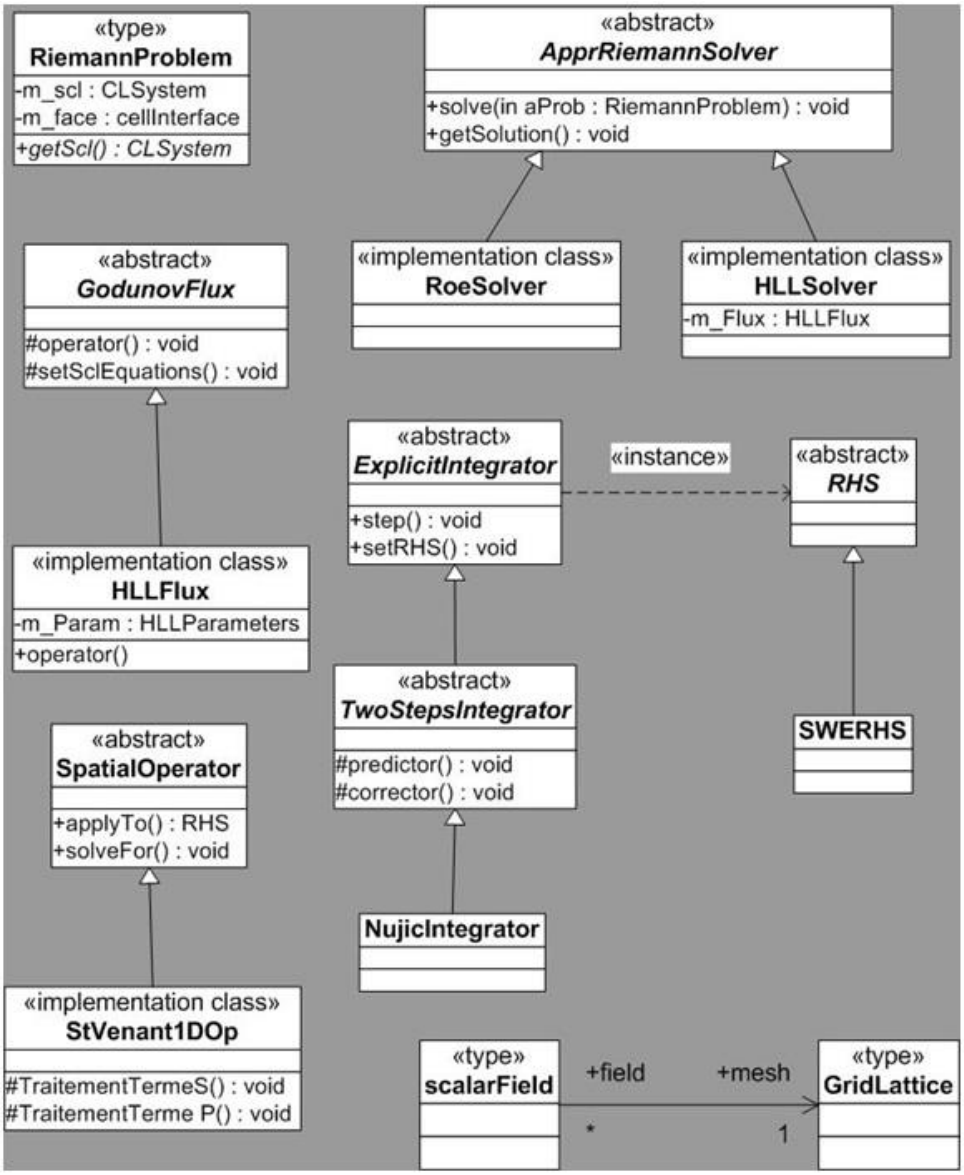
SFX Numeric Types

gridLattice, scalarField Class Concept de discrétisation spatiale et de représentation numérique des variables d'état. Concept qui est au cœur de notre modèle (passage du continu au discret) que l'on appelle (approximation) méthode numérique. Dans notre cas ceci correspond à une approximation aux différences finies.

ExplicitIntegrator class Concept de discrétisation temporelle (avance la solution par «pas» de temps). La librairie supporte des schémas explicites de type Euler ainsi que de la famille Runge-Kutta («pas» de temps multiples). D'autres schémas peuvent être programmes tout simplement créant une class qui dérive de la class abstraite ExplitIntegrator. Cette dernière fournit des services pour tous les schémas explicites.

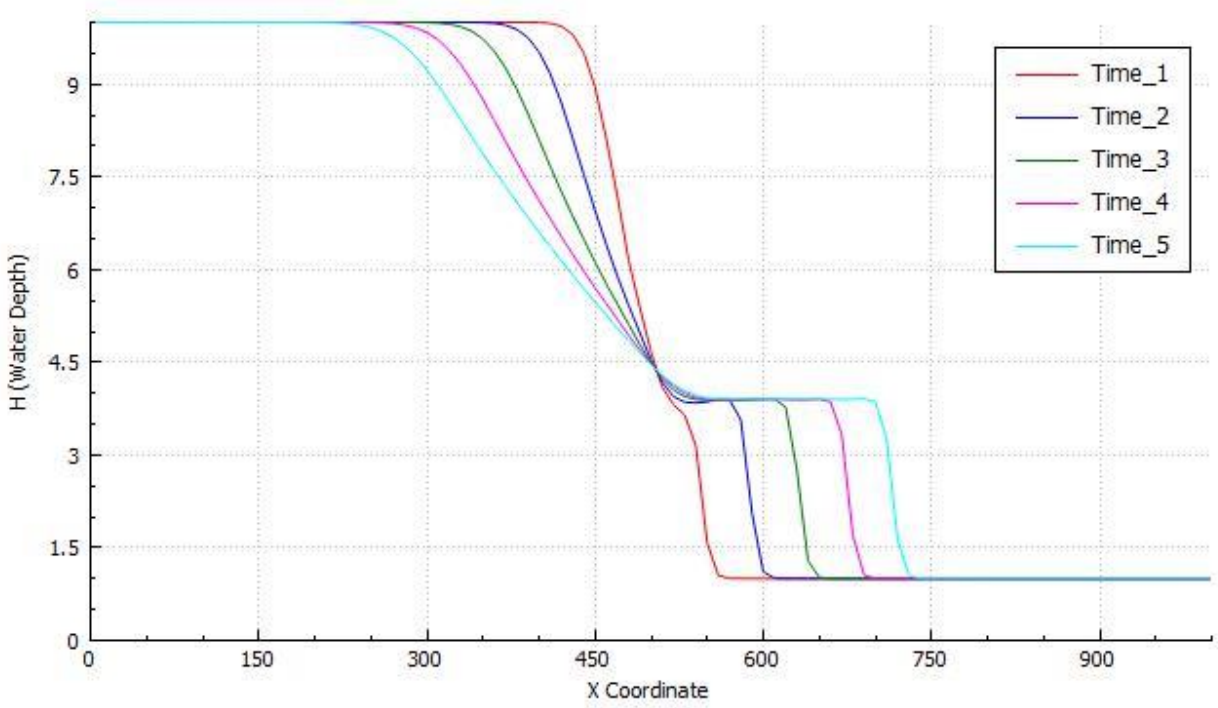
SpatialOperator class Concept qui représente les termes spatiaux de l'équation. Ces termes peuvent contenir des dérivées. Class abstraite avec une implémentation numérique par défaut pour les différents termes de l'équation de St-Venant unidimensionnelle.

Riemann Solver class Concept d'un solver (composante «stand-alone» qui résout un problème particulier). Dans ce cas-ci, on solutionne le problème de Riemann pour obtenir le flux à l'interface (cell). Différents «solver» existent, parmi les plus populaire est celui de Roe. Une class abstraite fournit les services qui doivent implémenter par les sous-class.



Modélisation de la Propagation d'une Onde de choc

Un schéma semi-discret au deuxième ordre en temps et en espace (modèle aux volumes finis) basé sur une discrétisation aux différences finies est utilisé pour résoudre les équations de Shallow-water. Le modèle avance par « pas » de temps selon un solveur de Runge-Kutta qui utilise une méthode prédicteur-correcteur. Les flux numériques sont calculés par un solveur de Riemann de type HLL (Harten-Lax-van Leer). Un schéma simple TVD est employé pour la méthode de capture de choc et une fonction de flux «Minmod limiter» est utilisée. Dans la figure ci-dessous nous voyons les profils de la propagation de l'onde de choc. La solution numérique est comparée avec les résultats analytiques. Les erreurs de la solution analytique sont estimées et présentées.



HLL Riemann Solver