Projet de Simulateur de Fluide

Stephane LEJEUNE, Jacques PHAM BA NIEN

9 mai 2025

Table des matières

1 Introduction

Pour le projet, nous avons choisis de construire un simulateur de fluide, projet non proposé de base, mais c'est un projet que Stephane et Jacques voulaient déjà faire depuis la L2, nous avons donc proposer le projet et il a été accepté.

Étant en double licence, le style de compétences utiles qu'on peut apprendre avec un projet informatique sont différentes, le projet de simulateur de fluide nous a sembler être un bon compromis entre demandant des compétences informatiques et mathématiques.

Ce projet, nous permet d'explorer comment les techniques qu'on as appris durant différent cours analyse numériques peuvent être utiliser pour quelque chose de plus "réel" (et visuel) que résoudre un système linéaire ou des équations différentiels abstraites.

Du point de vu informatique, ce projet nous oblige à interagir avec une interface graphique, choisir nos structures pour representer les particules (ou ne pas les representer)

Nous sommes donc reconnaissant que notre proposition de projet ait été accepté.

L'objectif du projet est double :

1. Implémenter une simulation de fluide stable, visuellement cohérente, et intéractive, capable de gérer plusieurs centaintes, voire mileirs de particules.
2. Intégrer cette simulation dans un moteur de jeu 2D, en l'occurrence, *Unity*, pour permettre une visualisation en temps réel avec un contrôle

sur les paramètres physiques.

Durant le développement, de nombreux défis ont été rencontrés : gestion des densités divergentes, explosions numériques liées à une mauvaise évaluation de la pression, instabilités aux frontières, ou encore des performances non optimales à partir d'un certain nombre de particules. Plusieurs améliorations progressives ont été apportés, incluant l'ajout d'une grille spatiale, la prédiction de position, un modèle de pression stable (Tait), et une modélisation simplifiée de la viscosité.

Ce rapport détaille les fondements physiques du modèles SPH, la manière dont il a été implémenté,optimisé et visualié dans Unity, ansi que les résultats obtenus, leurs limites et des pistes d'améliorations futures.

2 Théorie de simulateur de fluide

2.1 Equation de Navier-Stokes

Pour comprendre comment un simulateur de fluide marchent, il nous as d'abord fallu comprendre comment un fluide est sensé se comporter.

On as certes un modèle en nous de comment cela fonctionne, on a déjà vu des fluides, mais transmettre cette intuition en instructions est loins d'être facile, cela n'est pas notre travail, c'est celui des physiciens. Nous avons donc regarder à comment les physiciens décrivent les lois que les fluides doivent respecter.

Déjà, un "fluide" en physique ne décrit pas seulement le comportement d'un liquide, cela décrit aussi le comportement des gas, leurs comportement sont étudier dans la "mécanique des fluides".

Ses lois sont appeler “équations de Navier-Stokes” :

1. Équation de continuité :
$$\boldsymbol{\nabla} \cdot \vec{V} = 0$$
2. Équation de moments :
$$\rho(\frac{\partial \vec{V}}{\partial t} + \vec{V} \cdot \boldsymbol{\nabla}\vec{V}) = -\boldsymbol{\nabla} p + \mu \boldsymbol{\nabla}^2 \vec{V} + \vec{F}$$

L’équation de continuité nous dit que l’énergie mécanique du fluide ne change pas avec le temps (sa dérivé est nulle).

L’équation de moments est plus complexe, elle décrit les forces locales qui sont exercé sur notre fluide, décortiquons ce que veut dire chaque termes :

— ρ est la densité (locale, mais cela n’importe peu pour la suite) de notre fluide, donc c’est la masse divisé par le volume autour d’un point.
— $\vec{F}$ est la force externe exercé sur notre fluide, ici c’est juste la gravité. La force exercé par la gravité localement est $\vec{g}\frac{m}{V}$ où $\vec{g}$ est la force gravitationnel universelle, m est la masse et V est le volume. Autrement dit :
$$\vec{F} = \rho\vec{g}$$
.
— $\vec{V}$ est le champ vectoriel de la vitesse du fluide. Champ vectoriel car en tout point, le vecteur direction peut être différent.
— p est la pression du fluide.

— μ est une constante propre au fluide étudié, cela modèle la viscosité, plus elle est grande, plus le fluide est visqueux, ce terme limite l'accélaration.

Cette équation nous dit donc que le changement de vitesse locale est du aux forces extérieurs, et à la différences de vitesse environnantes, en évitant de trop se compresser, et n'accélérant pas trop vite proportionnelement à la constante de viscosité.

Tout du long, nous allons simplifier notre fluide étudier, en général, les fluides dont on s'intéresse (surtout l'eau) ne sont pas visqueux (nous rajouterons la viscosité plus tard pour des raisons de précision numériques, mais ignorons ce paramètre pour l'instant), et aussi (quasiment) incompréssible, cela simplifie nos équations, ce qui rends la simulation plus simple et plus rapide, nos nouvelles équations :

— Conservation :

$$\boldsymbol{\nabla} \cdot \vec{V} = 0$$

— Incompréssible :

$$\boldsymbol{\nabla} p = 0$$

— Moments :

$$\frac{\partial \vec{V}}{\partial t} = \vec{g} - \vec{V} \cdot \boldsymbol{\nabla} \vec{V}$$

2.2 Simulateurs

Certes maintenant on a "comment les fluides doivent agir", on n'a pas "comment les faire agir comme tel".

Pour observer la vie réel, cela n'a pas d'importance, pour un mathématicien non plus, mais un ordinateur de peut pas gérer des équations différentiels de manière exactes sur un volume continu, il y aurait juste beaucoup trop de données à stoquer et de calcul à faire (même, une infinité non dénombrable, d'où cette incapacité).

Ainsi, on est obliger de simplifier notre tache, au lieu d'obtenir un système agissant *exactement* comme un fluide idéal, il faut faire un système qui agit *approximativement* comme un fluide idéal, les physiciens font déjà cela pour simplifier les calculs (nous ne faisons pas en général de la théories des probabilités et des équations depuis le quantique pour des fluides, car il y aurait trop de termes agissant entre eux, obscursissant les calculs, et même pourrait rendre des équations insolvables de manière numériques), en informatique, nous allons passer du continu au discret, perdant ainsi de la précision pour en échange, avoir de la possibilité et de la vitesse de calcul.

Il y a plusieurs manières de discrétiser notre problème :

2.2.1 Simulateur Newtonien

On peut discretiser notre problème en considérant d'avoir une grille de vecteurs au lieu d'un champs, nous appellons chaques éléménts de la grille une "boîte" pour des raisons de facilité de visualisation.

Ainsi, il faut modeler la quantité de fluide passant d'une boîte à ses boites voisinnes.

Nous avons seulement besoin de modéliser le passage de fluide d'une boîte à ses voisinnes directes dans les directions de gauche, droite, haut et bas, pas besoin des diagonales.

Ainsi, une optimisation est de sauvegarder les magnitudes de ces flux au lieu de la magnitude et direction du flux partant du centre de la boîte.

En pseudocode, le calcul du flux ressemble à ça :

Algorithm 1 : Simulation de Newton

Entrée : grille
Sortie : Même grille
for *boîte dans grille* **do**
    Ajout flux de boîte à la boîte en dessous ;
Mettre_pression_à_zero(grille) ;

Pour fixer la pression (faire que le flux en entré et en sortie de toutes boîtes est à 0), une approche (Gauss-Seidel) est de fixer la pression de chaque boîte un par un, cela va defixer nos boîte fixé auparavent, mais en répétant cette opération, on converge vers la bonne solution (car la matrice associé à une diagonale dominante, mais ici la démonstration importe peu).

Appelons $s(i, j)$ la "facilité" à pousser le flux de i vers j (par exemple, peut être mis à 0 pour représenter que j est un mur, permet aussi de mettre en contacte des fluides plus dense, etc.), on peut alors faire :

Pour des raisons de performances, si il n'y a pas beaucoup de flux, il est également possible de mettre plusieurs boîte ensembles, de manière à ne plus avoir de grille, et se focaliser sur les endroits où il y a le plus d'actions (ce sont les endroits les plus important niveau comportement), il faudra alors changer le s, une autre raison pour laquel avoir s est pratique.

Algorithm 2 : Mettre pression à zero Newton

Entrée : grille, nombreÉtapes et s
Sortie : Même grille

```
1  étape = 0 ;
2  while étape < nombreÉtapes do
3      for boîte dans grille do
4          d = 0 ;
5          flux = 0 ;
6          for voisin dans voisins de boîte do
7              d = d + s(boîte, voisin) ;
8              flux = flux + flux_entre(boîte, voisin) * s(boîte, voisin) ;
9          for voisin dans voisins de boîte do
10             flux_entre(boîte, voisin) -= flux * s(boîte, voisin) / d ;
```

2.2.2 Simulateur Lagrangien

Une autre approche est de discretiser notre problème en une manière plus intuitive, en simulant nos *particules* d'eaux, néanmoins, dans un fluide usuel, il y a beaucoup trop de particules, donc à la place, on va simuler des très grosses particules, ce qui est absurde physiquement mais marche relativement bien.

L'approche Lagrangienne permet de ne pas avoir à considerer que tout notre environement est dans un fluide, par exemple ne pas avoir à considerer à la foix l'eau *et* l'air qui l'alentour.

Cette approche est plus proche de comment on intuitionne les fluides, mais plus loins du calcul mathématique que nous avons dérivé, il faut donc travailler avec des calcules plus complexe pour obtenir des valeurs tel que la vitesse, la température ou même la pression, ce problème de pression impacte de manière directe les performances, car on a généralement besoin de cette valeur pour simuler un fluide (à moins qu'on ait un grain très fin, alors les collisions entre particules génèrent d'elles même avec haute probabilités, des intéractions faisant penser à celle de la densité, en effet, un "fluide" n'est qu'un amas de beaucoup de particules intéragissant localement entre elles, notre comportement idéal n'est que le résultat probabiliste moyen que ces intéractions créent à grande échelle, sujet étudier en Physique Statistique).

Les soucis d'implémentation d'une méthode Lagrangienne sera expliquer plus tard (le simulateur SPH (Smoothed Paricle Hydrodynamics)) en détail, nous laissons donc cette sous section sans.

Néanmoins, ce qu'une approche Lagrangienne permet de faire, est de prendre un simulateur physique d'intéraction de corps usuel, et directement avoir nos fluides "gratuitement".

Cela permet aussi de l'étendre plus facilement, en effet, nous avons déjà l'intéraction entre petites particules, avec moins d'effort qu'avec une méthode Eulerienne, nous pouvons rajouter l'interaction de ces particules sur des objets de grande tailles.

2.2.3 Simulateur Mixtes

Tout comme vous avez pus le remarquer, les approches Newtoniennes et Lagrangiennes sont très différentes, et ont des bénéfices et des inconvénients.

Ce qui laisse la possibilité de mixer ces deux approches pour essayer de combler les inconvénients de chaques méthodes par l'autre.

Néanmoins, cette approche demande de gros efforts d'implémentation, beaucoup de test car il y a beaucoup plus de facteurs mouvants dans le système, pour des effets sur la performances qu'on peut obtenir en choisissant un système et l'optimisation bien avec le même effort (mais un système mixte et optimisé pourrait être plus rapide).

2.2.4 Simulateur SPH (Smoothed Paricle Hydrodynamics)

C'est le simulateur que nous avons décider d'implémenté, nous avons d'abord pencher sur une approche Newtonienne, mais TODO TODO

3 Le projet

3.1 Pre-Unity : le choix de rust

TODO

3.2 Post-Unity : l'abandon de rust

TODO

4 Conclusion

TODO