Assignment 3 Unconditionally Stable Fluids

Introduction

In assignment 3, you are expected to use C++ with OpenGL to implement an unconditionally stable fluid simulator to realize a simple fluid simulation from scratch. For this assignment, we are providing some reference codes to show you a simple OpenGL framework with an example fluid visualizer.

General Guidance

1. **Plagiarism is always strictly forbidden.**
2. These assignments are designed to help you learn the algorithms and meanwhile practice your programming skills. **We may or may not provide reference resources and everything is up to your design.**
3. Various implementations are accepted as long as the results are satisfactory. You can use your favorite programming language or frameworks if it is not specified in assignment.
4. **It is always requested to explain your codes to our TA after submission.**
5. It is encouraged to utilize open source projects or third party libraries for **non-core** part (tagged with **[non-core]**) of your homework. **BUT you are requested to also explain the key part of how their codes work**. Usage of misunderstood or mysterious codes leads to a score deduction.
6. You need implement yourself for the **core** part (tagged with **[core]**).
7. Feel free to send email to TA or post question on Piazza for help.
8. Good luck and have fun!

Reading Materials

The implementation would become easier if you understand the whole process before you start. There are some materials which might be helpful. Have a quick look before you start.

1. You can play with an awesome demo of a web fluid simulation here.
2. You can play with another toy demo of stable fluids realization here which gives the result that is expected for the assignment.
3. Here is Jos Stam's stable fluids paper. For the details of the algorithm, you can reference to the paper.
4. Here is a good interactive introduction to the stable fluids algorithm.
5. Here is a tutorial of implementing the algorithm on GPU.
6. Here and here provides good references of the python implementation of the algorithm.
7. And always, you can refer to our course slides about the fluid simulation.

Requirements

1. **[core]** Design the fluid data structure to save necessary quantities for simulation
    1. Input: parameters for the fluid simulation
    2. Hint: you should decide what physical properties of the fluid should be saved
    3. Hint: you can try using the Eigen::MatrixXf provided in the Eigen library as your 2D grid
2. **[core]** Implement the interpolation algorithm on your fluid.
    1. Hint: this is necessary to the advection step

    2. Hint: when the asked interpolation position is out of the field, you can either give zeros or find the value on the nearest edge depending on the scene settings
3. **[core]** Implement the advection step.
    1. Algorithm: unconditionally stable semi-Lagrangian advection reference to stable fluids algorithm.
    2. Hint: you should save the fluid properties for the backward tracing.
    3. Hint: velocity field is NOT the only field where the advection is applied.
    4. Hint: you should consider how to handle your boundary correctly when the backward tracing is outside the field.
    5. Hint: you should keep the boundary unchanged to apply the basic boundary conditions.
4. **[core]** Implement the projection step.
    1. Hint: you should first implement the calculation of the gradient and divergence over a field.
    2. Hint: the projection is basically to solve a finite difference Poisson problem.
    3. Hint: you can either implement an iterative linear system solver or take advantage of the solvers provided in Eigen library.
    4. Hint: if the simulation runs into blow-up quickly under a tiny perturbation, you should consider the correctness of the projection step and the boundary handling.
5. **[core]** Implement the diffusion step.
    1. Hint: This step is similar to step 4.
6. **[core]** Implement a jet flow to see the difference of the viscosity.
    1. Hint: This is an extra steps where you set the boundary explicitly.
    2. Hint: You can adjust the handling method of the out-of-field interpolation in the advection step to get a stable jet flow.
7. **[non-core]** Visualization system

Checkpoints

1. **[Req. 1, 7]** Visualize a fluid field with a circle at center without any dynamics *[5 pts]*
2. **[Req. 2]** Test your interpolation algorithm by clicking at the field *[20 pts]*
3. **[Req. 3, 4]** Simulate the behavior of a fluid field without viscosity (e.g. applying force on a circle) *[60 pts]*
4. **[Req. 5, 6]** Simulate the behavior of a jet flow with different viscosities *[15 pts]*
5. **[Optional]** Make some changes to the advection and projection step, in order to alleviate dissipation. You can refer to this paper. *[20 pts]*
6. **[Optional]** Implement the algorithm on GPU. *[20 pts]*
   [Notice] In each assignment, you can get at most 30 pts *[optional]* scores in total.

Starter Code and Submission

In this semester, we take advantage of the github classroom to send and collect assignments. You can access this link to accept the repository with the reference codes and example models we provide. **Please clone the repository down** to write your homework and **submit with git commit and git push.** If you are not familiar with git or github, please take a look at the github document.

If you decide to start from scratch and use little reference resources, you can put your work in the submission folder. Otherwise, you can modify and add codes in the reference folder and commit directly.

Submission Deadline: **23:59 on May 22th 2024**

Late hand-in policy

Each student is allocated a total of **five** late-day points for the whole semester, which work as follows:

- A student can extend a programming assignment deadline by one day using one point.
- If a student does not have remaining late-day points, late hand-ins will lead to a deduction of 10% of the total score per day.
- No assignment will be accepted more than five days after the deadline. This is true whether or not the student has late-day points remaining.
- We will strictly follow the rule above for late-hand-in policy unless you have a **VERY STRONG** reason, which should be explained to the course instructor and TAs.
