In this assignment you will implement and experiment with several methods for integrating particles forward in time, and then design a new animation with particles.

1 Implementation

Write a program that randomly (and uniformly) places N particles in the vertical cylinder:

$$x^2 + z^2 \leq 1$$

$$0 \leq y \leq 2$$

each with radius equal to 10^{-3}, and then integrates these forward for 10 seconds, saving the positions of the particles every $1/24^{\text{th}}$ of a second, in the velocity field:

$$v(x, y, z) = (-25yz, 0, 25xy)$$

The program should allow you to select N, with a default value of 1000. It also should let you select an integration method from one of Forward Euler, TVD-RK3, Backward Euler, and Trapezoidal Rule. Implement Newton's method to solve the equations in the implicit methods. For all of these, use a fixed time step of $1/24^{\text{th}}$ of a second (i.e. one film frame). Note that this may violate the stability limit we discussed in class—part of the point of this assignment is to appreciate how important stability is!

There is C code provided for you as an example to start work on if you want. The program is missing key definitions which are marked throughout. See the README file in the hw1_example directory for more information.

The output of the program should be a sequence of text files in a specified directory, one file per frame. The first line gives the number of particles in that frame, and each subsequent line gives the position (three floating-point numbers), colour (three floating-point numbers), and radius (one floating-point number) of a particle.

There is a program provided in the directory hw1_viewer you can use to visualize your simulations. See the README in that directory for more information. It makes one particular choice for the colours of the particles—you can choose anything you want (even boring all white).

2 Analysis

Given a particle whose initial position is $(r\cos\theta, y, r\sin\theta)$ at time 0, exactly solve where it should be at time t.

Although this is a nonlinear velocity field, why does Newton's method converge in one iteration for the implicit methods?

Since all the particles are distributed uniformly in the cylinder initially at time 0, what can you say about their distribution at time t? (if the system is solved exactly)

Describe (in loose flowery words) what sort of distribution each numerical method you implemented gives you. Explain the errors.

3 Fun

Customize the particle system: change the initial distribution of particles and the velocity field. Your program should take an option that lets you run the original one or the customized one.

If you want, you can add other parameters to the system that change in time such as colour or radius. You can make the new field depend on time, random numbers, interpolation from a grid of values, ... There are no limits, except that it has to boil

down to a simple set of differential equations:

$$\frac{d\mathbf{x}}{dt} = \mathbf{v}(\mathbf{x}, t)$$

that can then be solved with one of the numerical methods mentioned above. Here the vector $\mathbf{x}$ could just be (x, y, z) as above or it could include extra components such as RGB values or more. (the viewer currently only displays the positions and colours of the particles; you may need to extend it to see all of your results—please submit your modified viewer if so)

You can change the time step to something less than $1/24^{\text{th}} s$ for stability if you need, but please make sure that you output the particles at exactly the frame time even if you take intermediate steps between frames.

You may also, if you choose, delete particles when they are too old or go out of bounds, and you may add new particles throughout the simulation. (Look at the Reeves paper for how he did it.)

Play around and come up with the most dazzling display you can get. I'll play back some or all of them in class.

4 What to turn in

Email me your source code in a tar.gz file (don't include object files or the executable!) including the instructions on how to build and run it.

You do not have to use the provided example C code, and you may use a different language for the project, but please give detailed instructions on how to compile it on the computer science UNIX machines and how to change which method is used, which system is integrated (the given one or the one you come up with), and the output directory. If you needed to modify the viewer, please submit the source code and instructions for that too.

Also turn in your answers to the analysis section, either through email or by giving me hardcopy (either in person, or slipped under my office door, CICSR 189).