Particle Based Simulation

Physics 301, Fall 2022

Ayden Cook

12/9/22

Table of contents

Chapter 1

Introduction

The primary goal of the project is to create a general-purpose framework for particle based simulations. Particles have been chosen as the primary means of simulation because of their simplicity and generality. Also, most phenomena in reality can be described by underlying particle dynamics, adding further reason to investigate their use in simulation.

1.1 Background

Because the framework aims to support classical and non-relativistic behavior, the foundations of classical physics must be respected:

- Newton's Laws of Motion
- Conservation Laws (m, E, $\vec{p}$, $\vec{l}$)
- Laws of Thermodynamics

One of the primary issues with using particles as the underlying unit of simulation is the fact that most things that occur at the scale of humans have an order of magnitude of around 10^{24} particles involved. Performing any interesting simulation would thus be nearly impossible if each and every particle were to be simulated.

In order to account for this, methods for recovering a continuum from a smaller set of discrete particles will be needed. The most popular method for doing so is Smoothed Particle Hydrodynamics (SPH), which will be the subject of Section 2.1.2.

1.2 Usefulness

Important systems are often complex and involve many areas of physics merging and interacting. Thus the importance of this project stems from the fact that having a unified framework for physical simulation opens the door to studying situations that involve coupled and interacting phenomena.

Furthermore, having a single general purpose framework that captures a lot of overlapping implementation details would reduce much of the boilerplate and redundancy that is necessary when creating a simulation. Instead of needing a large amount of manual code just to test a new idea or study an important system, one can simply tap into what this framework provides and add any additional behavior as necessary.

Chapter 2

Methods

To remain general purpose, modular, and extensible, the framework has been created in such a way that allows for any type of particle behavior to easily tap into the system's main update loop. Two different APIs have been created to do so.

The first API is called an `Interaction`, with the entrypoint occuring at the beginning of the main loop. The second is called a `Constraint`, with an entrypoint after the particle integration step (see Listing 6.1).

The ordering of the update loop has been chosen carefully to allow as wide a range of behavior possible. In physics simulations, it is most accurate to first calculate forces and then perform integration based on those forces, therefore interactions are handled first. After the equations of motion have been integrated for each particle, the system proceeds to projecting constraints (see Section 2.2).

2.1 Interactions

Because the framework is crafted in a modular fashion, anyone can implement their own interactions. However, many useful/common types have already been implemented.

2.1.1 Particle Dynamics

Force-based simulation is ubiquitous, and is thus an obvious inclusion in a framework such as this one.

To reduce redundancy, a handful of common force "types" have been pre-implemented, including things such as force potentials, pairwise forces between particles, and forces that arrise due to an interaction with a field.

Many things can be simulated using these forces alone, such as kinetic gases, spring systems, Lennard-Jones fluids, gravity, etc.

2.1.2 Smoothed Particle Hydrodynamics (SPH)

Because of the large number of particles participating, macroscopic phenomena are often modelled as continuous. It is thus important to provide a way to recover a continuum from discrete particles. SPH is a method that aims to do so.

Because a particle has finite mass and zero extent, trying to calculate something like density (or another continuum quantity) requires the use of a Dirac-Delta function. Thus the foundational starting point of the SPH method is the following identity:

$$A(\vec{r}) = \iiint_{\mathbb{R}^3} A(\vec{r}\,')\delta(\vec{r} - \vec{r}\,')dV' \tag{2.1}$$

By recognizing that $dm = \rho dV$, and by substituing an approximation to the Dirac-Delta distribution, called the kernel W, we arrive at the following continuous formula:

$$A(\vec{r}) = \iiint\limits_{\mathbb{R}^3} \frac{A(\vec{r}')}{\rho(\vec{r}')} W(\vec{r}-\vec{r}', h) dm'$$

A primary constraint on W is that it must approach the Dirac-Delta distribution as $h \to 0$, giving h the name "the smoothing length." The final step is the discretization. Our spatial domain is sampled by a collection of particles, and so the integral over that domain will become the following discrete sum over the particles:

$$A(\vec{r}) = \sum_i^N m_i \frac{A_i}{\rho_i} W(\vec{r}-\vec{r}_i, h) \tag{2.2}$$

Equation 2.2 is the central equation to the SPH method, as it allows any set of continuum equations to be discretized and for the interpolatation of any continuous field using a set of discrete sampling particles.

A particularly common value that needs to be calculated is the density, so by setting $A(\vec{r}) = \rho(\vec{r})$, we arrive at the following equation:

$$\rho(\vec{r}) = \sum_i^N m_i W(\vec{r}-\vec{r}_i, h) \tag{2.3}$$

In a continuum, the particles experience an acceleration due to the local pressure field[1].

$$\frac{d\vec{r}_i}{dt} = -\frac{1}{\rho_i} \nabla_{\vec{r}_i} P$$

Discretizing this force alone will allow us to capture a lot of the behavior of a fluid, so it will serve as the primary example shown here.

A nice thing about the SPH method is that it allows us to easily calculate gradients by just passing the gradient operator through the summation, leading to:

[1]They also experience other forces such as viscosity, body forces, elasticity, etc, but we can treat all of them as independent.

$$\frac{d\vec{r}_i}{dt} = -\frac{1}{\rho_i} \sum_j^N m_j \frac{P_j}{\rho_j} \nabla W(\vec{r}_i - \vec{r}_j, h)$$

However, upon further inspection one may notice that the force on particle i due to particle j is not antisymmetric as Newton's Third Law requires for momentum conservation $\vec{F}_{ij} \neq -\vec{F}_{ji}$. To fix this, the following identity can be used:

$$\nabla(\frac{P}{\rho}) = \frac{1}{\rho}\nabla P - \frac{P}{\rho^2}\nabla\rho$$

Using this instead, we arrive at the pairwise force that is central to SPH:

$$\vec{F}_{ij}^{pressure} = -m_i m_j (\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2}) \nabla W(\vec{r}_i - \vec{r}_j, h) \tag{2.4}$$

Given that $\nabla W(\vec{r}_i - \vec{r}_j, h) = -\nabla W(\vec{r}_j - \vec{r}_i, h)$, this pairwise force will conserve momentum. Finally, to close the equations of motion, an equation of state is needed so that pressure can be calculated. A common one in astrophysics is the polytrope $P = K\rho^{\frac{n+1}{n}}$.

2.2 Constraints

Oftentimes when trying to describe the behavior of a system it is easier to describe what it *can't do* rather than how it does it. It is for this reason that constraints play a central role in the framework alongside interactions. For example, instead of trying to analyze every force involved in a triple pendulum's motion, an easier description would be to specify the fixed distance between each mass.

2.2.1 Extended Position Based Dynamics (XPBD)

The primary method for constraint solving that has already been implemented is XPBD, which is a mathematical framework and algorithm for compliant constrained dynamics. The implementation here is adapted from Macklin et al. (2019).

The framework's main update loop has been ordered in such a way as to allow for the natural inclusion of XPBD (and other constraints).

2.2.1.1 Equations

XPBD works by specifying a function of particle positions that must always be satisfied, called a constrain function. The implementation here allows for two types:

$$C(\vec{x}_1, ..., \vec{x}_i) = 0 \tag{2.5}$$

$$C(\vec{x}_1, ..., \vec{x}_i) \geq 0 \tag{2.6}$$

The goal is to then find a position change for each particle $\Delta\vec{x}_i$ such that the constraint becomes satisfied. XPBD does this by using an approximate implicit integration step where the constraint function is linearized. A detailed derivation can be found in Macklin et al. (2019).

$$\Delta\vec{x}_i = \lambda w_i \nabla_{\vec{x}_i} C(\vec{x}) \tag{2.7}$$

$$\lambda = \frac{-C(\vec{x}) - \frac{\alpha\beta}{\Delta t}\sum_i \nabla_{\vec{x}_i} C(\vec{x}) \cdot (\vec{x}_i - \vec{x}_i^{prev})}{(1 + \frac{\alpha\beta}{\Delta t})\sum_i w_i |\nabla_{\vec{x}_i} C(\vec{x})|^2 + \frac{\alpha}{\Delta t^2}} \tag{2.8}$$

XPBD uses the inverse mass of each particle w_i so that setting $w_i = 0$ creates a particle that cannot be moved by the constraint. Also, XPBD allows the constraints to have both compliance and dampening by adjusting α and β respectively.

Chapter 3

Results

Chapter 4

Discussion

Chapter 5

Conclusion

Chapter 6

Appendix

6.1 Code

Listing 6.1 Update Algorithm

```
sub_dt = dt / substep_count

for each substep_count:
    for each interaction:
        interaction.handle(sub_dt)

    for each particle:
        particle.integrate(sub_dt)
        particle.forces.clear()

    for each constraint:
        constraint.project(sub_dt, false)

    for each particle:
        particle.update_vel(sub_dt)

time += dt
```

6.2 Relevent Links

- Project GitHub Repository

References

Macklin, Miles, Kier Storey, Michelle Lu, Pierre Terdiman, Nuttapong Chentanez, Stefan Jeschke, and Matthias Müller. 2019. "Small Steps in Physics Simulation." In *Proceedings of the 18th Annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 1–7.