Симуляция динамики жидкости и её взаимодействия с твёрдой поверхностью средствами графических ускорителей.

Юлия Хохлова
Физический факультет
Новосибирский государственный университет, Новосибирск, Россия
girl.from.ff.nsu@gmail.com

Денис Гладкий
ООО «Онлайника», Новосибирск, Россия
generalgda@gmail.com

Станислав Кузиковский, Игорь Белаго
Институт автоматики и электрометрии СО РАН, Новосибирск, Россия
{stas, bel}@sl.iae.nsk.su

Аннотация

В данной статье предлагается метод моделирования динамики жидкости и её взаимодействия с окружающим виртуальным миром в интерактивных графических приложениях. Такой подход применим для анимации разлива воды на дороге или попадания объекта под струю водопада. Объём жидкости представляется набором взаимодействующих друг с другом частиц, поверхность, формируемая окружающими жидкость объектами, задаётся картой высот и нормалей, а поверхность жидкости генерируется и отображается как полигональная сетка. Высокая скорость расчёта кадра, позволяющая производить анимацию в реальном времени, достигается за счёт использования возможностей эффективных параллельных вычислений общего назначения на современных графических процессорах с помощью программного интерфейса DirectX 11.

***Ключевые слова**: Моделирование Жидкости GPU, Карта Высот, Динамика Жидкости в Реальном Времени*.

1. ВВЕДЕНИЕ

В современных приложениях виртуальной реальности, требования к которым по качеству генерируемых изображений постоянно растут, моделирование и визуализация жидкостей является актуальной задачей. Использование программируемого конвейера видеокарты в качестве математического сопроцессора – один из немногих способов получения интерактивных сцен с присутствующей на них жидкостью.

Полная реализация симуляции на графическом процессоре позволяет избежать многократного относительно медленного копирования данных между системной памятью и памятью видеокарты.

Эффективность моделирования взаимодействия жидкости с твёрдой поверхностью окружающего виртуального мира может быть повышена применением техники, не зависящей от способа представления объектов сцены.

2. ДОСТИЖЕНИЯ В ПРЕДМЕТНОЙ ОБЛАСТИ

В настоящее время в игре Alice: Madness Return [5] реализовано моделирование подобных явлений с использованием до 10000 частиц, однако отображение жидкости проводится обработкой проекции модели на экран. Множество реализаций моделирования жидкости набором частиц представлено в работах по исследованию возможностей параллельных вычислений на видеокартах с использованием CUDA [4]. Однако в таких задачах не изучается ни вопрос взаимодействия жидкости с окружающим миром, ни применение различных методов визуализации результатов моделирования на экране.

В статье [1] предлагается метод, при котором высота столба жидкости не влияет на объём вычислений, поскольку детальное внимание сосредоточено на верхних слоях жидкости, но при симуляции процессов разлива жидкости по поверхности преимущества такого подхода не проявят себя. В работе [2] представлена техника, комбинирующая моделирование большого объёма жидкости частицами и полем высот, при этом твёрдая поверхность окружающего мира задаётся полигональной моделью. В [7] описывается алгоритм, позволяющий представлять поверхность жидкости в виде набора треугольников, который на каждом шаге модифицируется, а не генерируется заново, но реализации метода была произведена только на центральном процессоре и дала частоту кадров анимации 6 кадр/с.

3. ВЫБОР ТЕХНОЛОГИЙ

В настоящее время существует множество реализаций моделирования жидкости набором частиц на графическом процессоре. В задачах компьютерной графики главная цель состоит в реалистичном визуальном результате, что подразумевает не только моделирование динамики жидкости, но и отображение её поверхности.

В отличие от широко применяемой в научных кругах технологии CUDA или OpenCL, разработка игр ведётся преимущественно на языке C++ с использованием программного интерфейса DirectX. Поэтому было решено использовать возможности Shader Model 5.0 DirectX 11, в частности, Compute Shader 5.0 для вычислений общего назначения, а также Geometry Shader 4.0 для генерации геометрии на видеокарте по произвольным данным, описывающим модель виртуального мира.

4. ДИНАМИКА ЖИДКОСТИ

4.1 Гидродинамика сглаженных частиц

Для моделирования жидкости предлагается использовать представление её объёма в виде системы частиц, что позволяет использовать достаточно простую форму уравнения Навье-Стокса

$$\rho \frac{\partial \boldsymbol{v}}{\partial t} = -\nabla p + \mu \Delta \boldsymbol{v} + \rho \boldsymbol{g}. \quad (1)$$

Влияние каждой частицы на свойства оценивается в соответствии с расстоянием до интересующей точки. Дискретные частицы имеют характерный радиус h, на котором их свойства сглаживаются радиально симметричной функцией ядра W($|r|$, h). Значение физической величины A в точке $\boldsymbol{r}$ описывается выражением

$$A(\boldsymbol{r}) = \int A(\boldsymbol{r}')W(|\boldsymbol{r} - \boldsymbol{r}'|, h)d\boldsymbol{r}\,, \quad (2)$$

причём функция W должна удовлетворять следующим условиям:

1. $\int_{R^3} W(|\boldsymbol{r}|, h)dr = 1$ – условие нормировки.
2. $\int_{R^3} A(\boldsymbol{r}')\lim_{h\to 0} W(|\boldsymbol{r} - \boldsymbol{r}'|, h)d\boldsymbol{r} = A(\boldsymbol{r})$ – условие вырождения, т.е. при $h \to 0$ величина определена в одной точке пространства.

В методе гидродинамики сглаженных частиц для каждого элемента жидкости отслеживаются координаты, скорость и плотность. Значение величины в конкретной точке получается суммированием по всем окружающим частицам

$$A(\boldsymbol{r}) = \sum_j A_j \frac{m_j}{\rho_j} W\big(|\boldsymbol{r} - \boldsymbol{r}_j|, h\big), \quad (3)$$

а значение, например, плотности в интересующей точке составляет

$$\rho(\boldsymbol{r}) = \sum_j m_j\, W\big(|\boldsymbol{r} - \boldsymbol{r}_j|, h\big). \quad (4)$$

Пренебрегая влиянием удалённых частиц, выбирается функцию ядра такая, что $W(|\boldsymbol{r}|, h) = 0$ при $|\boldsymbol{r}| > h$,. При этом отпадает необходимость проводить суммирование по всем частицам, ограничившись лишь близлежащим, что может значительно сэкономить время вычислений.

Преимущество гидродинамики сглаженных частиц состоит в простоте оценивания фигурирующих в уравнении (1) частных производных, поскольку их вычисление сводится к дифференцированию ядра:

$$\frac{\partial}{\partial x} A(\boldsymbol{r}) = \sum_j A_j \frac{m_j}{\rho_j} \frac{\partial W\big(|\boldsymbol{r} - \boldsymbol{r}_j|, h\big)}{\partial x}, \quad (5)$$

$$\nabla A(\boldsymbol{r}) = \sum_j A_j \frac{m_j}{\rho_j} \nabla W\big(|\boldsymbol{r} - \boldsymbol{r}_j|, h\big), \quad (6)$$

$$\Delta A(\boldsymbol{r}) = \sum_j A_j \frac{m_j}{\rho_j} \Delta W\big(|\boldsymbol{r} - \boldsymbol{r}_j|, h\big). \quad (7)$$

4.2 Моделирование сил

Перейдём к формулам расчёта сил, действующих на частицу в текущий момент времени, и приобретаемого при этом частицей ускорения.

$$\boldsymbol{a}_i = \frac{\boldsymbol{f}_i}{\rho_i} \quad (8)$$

$$\boldsymbol{f}_i = \rho_i \boldsymbol{g} + \boldsymbol{f}_i^{pressure} + \boldsymbol{f}_i^{viscosity} \quad (9)$$

Согласно формуле (3) при использовании метода гидродинамики сглаженных частиц сила давления находится по формуле (10)

$$\boldsymbol{f}_i^{pressure} = -\boldsymbol{\nabla} p(\boldsymbol{r}_i) = -\sum_j m_j \frac{p_j}{\rho_j} \nabla W\big(|\boldsymbol{r}_i - \boldsymbol{r}_j|, h\big) \quad (10)$$

Однако в такой ситуации сила давления будет несимметрична. При взаимодействии двух частиц, частица i учитывает влияние только частицы j и наоборот, а полученные значения сил могут быть различны по величине, что противоречит третьему закону Ньютона. В литературе описаны различные решения этой проблемы, согласно [3] приемлема с точки зрения скорости вычислений и стабильности формула моделирования силы давления (11)

$$\boldsymbol{f}_i^{pres} = -\nabla p(\boldsymbol{r}_i) = -\sum_j m_j \frac{p_j + p_i}{\rho_j} \nabla W\big(|\boldsymbol{r}_i - \boldsymbol{r}_j|, h\big). \quad (11)$$

Давление в точке предлагается моделировать простой формулой (12), где ρ_0 – плотность жидкости в свободном состоянии.

$$p = k(\rho - \rho_0) \quad (12)$$

Сила вязкости определяется соотношением (13)

$$\boldsymbol{f}_i^{visc} = \mu \Delta p(\boldsymbol{r}_i) = \mu \sum_j m_j \frac{\boldsymbol{v}_j}{\rho_j} \Delta W\big(|\boldsymbol{r}_i - \boldsymbol{r}_{j|}, h\big). \quad (13)$$

Опять же, для симметричности силы вводится модифицированная формула 14:

$$\boldsymbol{f}_i^{visc} = \mu \Delta p(\boldsymbol{r}_i) = \mu \sum_j m_j \frac{\boldsymbol{v}_j - \boldsymbol{v}_i}{\rho_j} \Delta W\big(|\boldsymbol{r}_i - \boldsymbol{r}_j|, h\big). \quad (14)$$

Подробные обоснования выбора моделей действующих сил приводятся в [3, 8].

4.3 Функции ядра

За время применения гидродинамики сглаженных частиц было предложено множество функций, удовлетворяющих требованиям, описанным в пункте 4.1. При моделировании жидкости в задачах компьютерной графики наибольшее распространение получили следующие функции, которые используются при реализации алгоритма.

1. Для вычисления плотности частиц используется

$$W_{poly6}(r, h) = \frac{315}{64\pi h^9} \begin{cases} (h^2 - r^2)^3, \text{если } 0 \le r \le h \\ 0, \text{иначе} \end{cases} \quad (15)$$

2. Для расчёта силы давления жидкости, действующей на частицу жидкости, применяется функция

$$W_{spiky}(r, h) = \frac{15}{\pi h^6} \begin{cases} (h - r)^3, \text{если } 0 \le r \le h \\ 0, \text{иначе} \end{cases} \quad (16)$$

3. Для учёта вязкости в жидкости служит функция

$$W_{visc}(r, h) = \frac{15}{2\pi h^3} \begin{cases} -\frac{r^3}{2h^3} + \frac{r^2}{h^2} + \frac{h}{2r} - 1, 0 \le r \le h \\ 0, r > h \end{cases} \quad (17)$$

Подробное обоснование выбора ядер можно найти [10].

5. ВЗАИМОДЕЙСТВИЕ С ПОВЕРХНОСТЬЮ

5.1 Представление поверхности

Окружающий виртуальный мир, в котором симулируется жидкость, может быть представлен самым различными моделями данных. При этом учёт каждой из них при взаимодействии с элементами жидкости создаёт необходимость специальной подготовки дополнительных данных художниками, их хранения в памяти и специальная обработка, что неэффективно. По этой причине предлагается получать данные о строении поверхности путём отрисовки модели окружающего мира в карту высот. На современных видеокартах эта операция производится достаточно быстро [11]. Более того, карта высот зачастую генерируется для использования в других алгоритмах, таких как освещение или обнаружение столкновений. Также для расчёта взаимодействия частиц с полем высот требуется соответствующая поверхности карта нормалей.

5.2 Поиск точки контакта

Для обнаружения столкновения частицы с поверхностью и обработки отражения был адаптирован алгоритм отображения рельефных текстур [9].

Траектория движения частицы между положениями в двух последовательных моментах времени является прямой. На

каждом шаге моделирования для каждой частицы происходит проверка на попадание под поверхность. Если частица оказывается ниже поверхности, то необходимо произвести поиск пересечения отрезка траектории с картой высот. Для этого используется метод дихотомии.

Чтобы избежать ошибки этого метода при многократном пересечении траектории с поверхностью (см. рисунок № в центре), сначала выполняется линейный поиск с постоянным шагом δ, размер которого зависит от угла между направлением движения частицы и горизонтальной плоскостью. При обнаружении двух точек, одна из которых над поверхностью, другая - под ней, запускается алгоритм двоичного поиска.

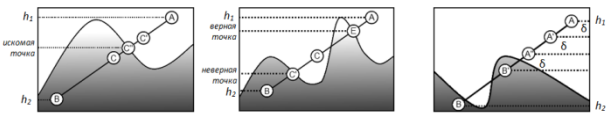

Рисунок 1: Поиск пересечения отрезка с полем высот методом дихотомии (слева). Возможные ошибки прямого применения метода дихотомии (в центре). Выполнение линейного поиска для исключения ошибки метода дихотомии (справа).

Пусть A и B – начальное и конечное положение частицы, причём A находится над поверхностью, B – под поверхностью. Берём точку C – среднюю точку между A и B и проверяем её положение относительно поверхности. Если точка C оказывается над поверхностью, повторяем действия для отрезка CB, иначе, для отрезка AC. Схема работы алгоритма представлена на рисунке № слева.

5.3 Отражение частицы

Вблизи точки столкновения элемент поверхности задаётся плоскостью, ориентированной нормалью. Отрезки траектории частицы до и после отражения лежат в одной плоскости. Если частица пересекла поверхность в точке C, то после упругого удара она должна оказаться в точке D, симметричной точке B относительно прямой γ, по которой пересекаются плоскости падения-отражения и поверхности.

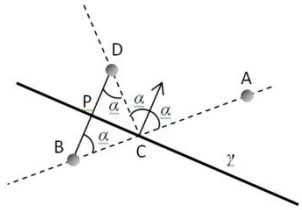

Рисунок 2: Схема отражения частицы от плоскости

Пусть O – начало координат, тогда $\overrightarrow{OA}$ и $\overrightarrow{OB}$ – трёхмерные векторы координат точек A и B соответственно. Описанным выше алгоритмом находятся координаты точки C. Далее необходимо вычислить положение $\overrightarrow{OD}$ частицы после отражения. Итак,

$$\overrightarrow{OD} = \overrightarrow{OB} + \overrightarrow{D} = \overrightarrow{OB} + 2\overrightarrow{\quad}$$

Между тем,

$$\overrightarrow{\quad} = C\ co\ s\alpha \cdot \vec{} = (\overrightarrow{C}, \vec{}) \cdot \vec{},$$

тогда

$$\overrightarrow{OD} = \overrightarrow{OB} + 2(\overrightarrow{C}, \vec{}) \cdot \vec{}$$

Аналогичными рассуждениями находится изменение направления движения частицы при сохранении модуля скорости.

Заметим, что коэффициент при множителе $(\overrightarrow{C}, \vec{}) \cdot \vec{}$ заключает в себе степень упругости удара и может изменяться в интервале от 1 до 2. Абсолютно упругий удар соответствует коэффициенту 2. Промежуточные значения указанного интервала симулируют различную степень потери энергии частицы при столкновении. Однако в случае моделирования жидкости более естественным выглядит неупругое столкновение с условием непротекания, когда нормальная компонента скорости частицы после контакта со стенкой становится нулевой. Такая модель соответствует коэффициенту 1.

6. ВИЗУАЛИЗАЦИЯ ЖИДКОСТИ

При выборе метода визуализации жидкости было поставлено требование получения полигональной модели поверхности жидкости, которая в дальнейшем может быть применена для наложения текстур или построения теней. Эффективным решением поставленной задачи является метод марширующих кубиков (Marching cubes) [6], основной параметр которого заключается в выборе размера ячейки при построении решётки. На практике было установлено, что удовлетворительный визуальный результат отображения поверхности при требовании интерактивности достигается при размере ячейки, по порядку величины равном размеру частиц.

7. РЕЗУЛЬТАТЫ

На рисунке 3 приведены результаты моделирования с использованием 65 тысяч частиц и размером ячейки марширующих кубиков равным 0,7 размера частицы. Симуляция проводилась в режиме реального времени с частотой кадров 47 кадр/с.

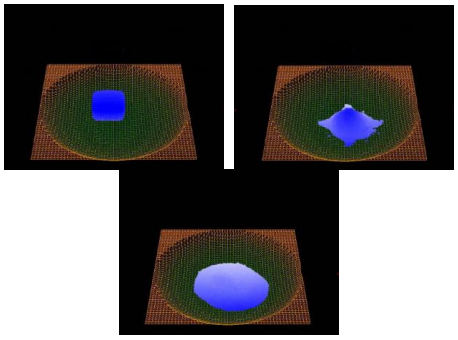

Рисунок 3: Кадры анимации.

При реализации предложенного алгоритма моделирования динамики жидкости и её взаимодействия с твёрдой поверхностью на графическом процессоре NVIDIA GeForce GTX 550Ti достигнута следующая производительность (данные приведены в таблице 1).

Количество частиц	Частота кадров, кадр/с	Время кадра, мс
8192	315	3,2
16384	270	3,7
32768	200	5
65536	115	8,7
131072	60	16,7
262144	25	40

Таблица 1: Результаты измерений.

8. ЗАКЛЮЧЕНИЕ

В статье был предложен и реализован метод симуляции динамики жидкости, представленной набором частиц, и её взаимодействия с твёрдой поверхностью, задаваемой картой высот, что делает алгоритм не зависимым от модели представления объектов окружающего виртуального мира, с которыми взаимодействует жидкость. Реализация на графическом ускорителе позволила получить скорость расчёта кадра анимации, достаточную для применения в интерактивных графических приложениях. Техника реализации демонстрирует возможности Shader Model 5.0 DirectX 11, а также использует возможность генерации геометрических примитивов на видеокарте по произвольной модели данных, описывающей объект виртуального мира.

На данном этапе было поставлено требование интерактивности анимации, которое и было достигнуто. Оптимизация полученного решения будет проведена в дальнейшей работе над проблемой.

9. ССЫЛКИ

[1] Nuttapong Chentanez, Matthias Müller. *Real-Time Eulerian Water Simulation Using a Restricted Tall Cell Grid* / *ACM Transactions on Graphics (SIGGRAPH 2011)*.

[2] Nuttapong Chentanez, Matthias Müller. *Real-Time Simulation of Large Bodies of Water with Small Scale Details* / *SIGGRAPH/EUROGRAPHICS Proceedings, 2010*.

[3] Mathieu Desbrun, Marie-Paule Cani. *Smoothed Particles: A New Paradigm for Animating Highly Deformable Bodies* / *Proceedings of EG Workshop on Animation and Simulation*, *2009*.

[4] Nolan Goodnight. *CUDA/OpenGL Fluid Simulation* / *Technical article, NVIDIA Corporation, 2007*.

[5] Simon Green, Richard Tonge, Miguel Sainz, Dane Johnston, David Schoemehl. *Fluid Simulation in Alice: Madness Returns* / *2011*.

[6] William E. Lorensen, Harvey E. Cline. *Marching cubes: A high resolution 3d surface construction algorithm* / *SIGGRAPH Proceedings, 1987*.

[7] Matthias Müller. *Fast and Robust Tracking of Fluid Surfaces. ACM SIGGRAPH / EUROGRAPHICS Proceedings, 2009.*

[8] Matthias Müller, David Charypar, Markus Gross. *Particle-Based Fluid Simulation for Interactive Applications / Eurographics/SIGGRAPH Proceedings, 2003*.

[9] Fabio Policarpo, Manuel M. Oliveira, Joao L. D. Comba. *Real-Time Relief Mapping on Arbitrary Polygonal Surfaces / SIGGRAPH Proceedings, 2005*.

[10] Natalya Tatarchuk, Jeremy Shopf, Chris DeCoro. *Scalar to Polygonal: Extracting Isosurfaces Using Geometry Shaders / ShaderX7: Advanced Rendering Techniques, Course Technology, 2009*.

[11] NVIDIA Corporation. *GPU Programming Guide GeForce 8 and 9 Series, 2008.*