Денис, Вы страницы не забыли поставить?

**План.**

Слово «план», думаю, можно удалить.

Введение   
1. Аналитический раздел

1.1. Выбор метода моделирования

1.2. Углубленный анализ метода SPH

1.3. Выбор метода визуализации  
2. Конструкторский раздел

2.1. Структуры данных

2.2. Алгоритм SPH

2.3. 2D жидкость

2.4. Алгоритмы визуализации  
3. Технологический раздел

3.1. Структура программы

3.2. Описание классов

3.3. Интерфейс  
4. Исследовательский раздел

4.1. Исследование поведения жидкости в зависимости от выбранных функций ядра.

4.2. Исследование скорости вычисления физики жидкости

4.3. Исследование скорости генерации изображения в зависимости от кол-ва частиц каждым алгоритмом.

4.4. Исследование реалистичности визуализации.

Заключение  
Список литературы

**Введение.**

Обычно выравнивание делаю по ширине (возможно у меня плохо сконвертировалось).

Обычно «имена» разделов выравнивают по левому краю.

**Моделирование жидкости** - область [компьютерной графики](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%BA%D0%B0), использующая средства [вычислительной гидродинамики](http://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%B3%D0%B8%D0%B4%D1%80%D0%BE%D0%B4%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D0%BA%D0%B0) для реалистичного моделирования, анимации и визуализации жидкостей, газов, взрывов и других связанных с этим явлений. Имея на входе некую жидкость и геометрию сцены, симулятор жидкости моделирует её поведение и движение во времени, принимая в расчёт множество физических сил, объектов и взаимодействий. Моделирование жидкости широко используется в компьютерной графике и ранжируется по вычислительной сложности от высокоточных вычислений для кинофильмов и спецэффектов до простых аппроксимаций, работающих в режиме [реального времени](http://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%B2%D1%80%D0%B5%D0%BC%D1%8F) и использующихся преимущественно в [компьютерных играх](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0).

[17]

**Цель работы:**

Разработать программу, которая будет моделировать поведение некоторой жидкости в каком-либо объеме и визуализировать её.

**Задачи:**

1 рассмотреть методы моделирования жидкости и выбрать метод, с помощью которого будет работать моя программа. После числа обычно ставят точку или скобку. Если предложение заканчивается точкой, то начинается оно с большой буквы.

2 на основе выбранного метода разработать структуры данных и алгоритмы для моделирования жидкости.

3 программно реализовать алгоритмы моделирования жидкости.

4 исследовать скорость моделирования и реалистичность моделирования жидкости в зависимости от заданных параметров

(а программно реализовывать не будем, сразу исследовать?). Это стоит удалить.

5 рассмотреть методы визуализации жидкости.

6 реализовать наиболее подходящий метод визуализации.

7 исследовать быстродействие реализованного метода и влияние параметров модели на получаемый результат.

1. **Аналитический раздел.**

Существует два разных метода моделирования жидкости.

**Первый - моделирование только поверхности жидкости**. Этот метод достаточно быстрый для того, чтобы работать в реальном времени, но он подходит только для очень узкого круга задач.

**Второй - полноценное моделирование жидкости**. Этот метод позволяет моделировать любые сцены с участием жидкости.

**1.1. Выбор метода моделирования**

В свою очередь есть несколько конкурирующих методов полноценного моделирования жидкости, каждый из которых имеет свои преимущества и недостатки. Наиболее распространёнными являются сеточные методы Эйлера, [гидродинамика сглаженных частиц](http://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%B4%D1%80%D0%BE%D0%B4%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D0%BA%D0%B0_%D1%81%D0%B3%D0%BB%D0%B0%D0%B6%D0%B5%D0%BD%D0%BD%D1%8B%D1%85_%D1%87%D0%B0%D1%81%D1%82%D0%B8%D1%86) ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *smoothed particle hydrodynamics — SPH*), методы, основанные на завихрениях, и [метод решёточных уравнений Больцмана](http://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D1%80%D0%B5%D1%88%D1%91%D1%82%D0%BE%D1%87%D0%BD%D1%8B%D1%85_%D1%83%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B9_%D0%91%D0%BE%D0%BB%D1%8C%D1%86%D0%BC%D0%B0%D0%BD%D0%B0). Эти методы возникли в среде вычислительной гидродинамики и были позаимствованы для практических задач в индустрии компьютерной графики и спецэффектов. Основное требование к данным методам со стороны компьютерной графики — визуальная правдоподобность. То-есть если человек-наблюдатель через просмотр не может заметить неестественность анимации, то моделирование считается удовлетворительным. В физике, технике и математике, с другой стороны, основные требования предъявляются к физической корректности и точности моделирования, а не к её визуальному результату.

Сеточные методы Эйлера и методы, основанные на завихрениях сложны и в своей работе я их не рассматриваю.

**Метод решёточных уравнений Больцмана.**

В отличие от многих других методов, метод LBM не решает [уравнения Навье — Стокса](http://ru.wikipedia.org/wiki/%D0%A3%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F_%D0%9D%D0%B0%D0%B2%D1%8C%D0%B5_%E2%80%94_%D0%A1%D1%82%D0%BE%D0%BA%D1%81%D0%B0), а моделирует поток [ньютоновской жидкости](http://ru.wikipedia.org/wiki/%D0%9D%D1%8C%D1%8E%D1%82%D0%BE%D0%BD%D0%BE%D0%B2%D1%81%D0%BA%D0%B0%D1%8F_%D0%B6%D0%B8%D0%B4%D0%BA%D0%BE%D1%81%D1%82%D1%8C) дискретным [кинетическим уравнением Больцмана](http://ru.wikipedia.org/wiki/%D0%9A%D0%B8%D0%BD%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D1%83%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5_%D0%91%D0%BE%D0%BB%D1%8C%D1%86%D0%BC%D0%B0%D0%BD%D0%B0). Столкновения зачастую учитываются с помощью [модели Батнагара — Гросса — Крука](http://ru.wikipedia.org/w/index.php?title=%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%91%D0%B0%D1%82%D0%BD%D0%B0%D0%B3%D0%B0%D1%80%D0%B0_%E2%80%94_%D0%93%D1%80%D0%BE%D1%81%D1%81%D0%B0_%E2%80%94_%D0%9A%D1%80%D1%83%D0%BA%D0%B0&action=edit&redlink=1). Методы решёточных уравнений Больцмана удобны благодаря их концептуальной и вычислительной простоте, их использование ограничено малыми скоростями и тем, что LBM обладает неустойчивым поведением на границе подвижных тел.

**Алгоритм**

Метод LBM рассматривает жидкость как совокупность относительно небольшого числа частиц, причем на каждом шаге рассматривается их распространение и столкновения (релаксация).

В каждой ячейке решётки поток жидкости рассматривается как совокупность элементарных потоков (например, идущих в соседние и следующие за соседними ячейки).

Этот метод чаще всего используется для моделирования завихрений и изменений плотности, например, воздуха, в турбулентной камере. Для моделирования жидкостей чаще применяется метод описанный

**Метод «Гидродинамика сглаженных частиц».**

Гидродинамика сглаженных частиц ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Smoothed Particle Hydrodynamics*, **SPH**) — [вычислительный метод](http://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4) для [симуляции жидкостей](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BC%D1%83%D0%BB%D1%8F%D1%86%D0%B8%D1%8F_%D0%B6%D0%B8%D0%B4%D0%BA%D0%BE%D1%81%D1%82%D0%B8) и [газов](http://ru.wikipedia.org/wiki/%D0%93%D0%B0%D0%B7). Используется во многих областях исследований, включая [астрофизику](http://ru.wikipedia.org/wiki/%D0%90%D1%81%D1%82%D1%80%D0%BE%D1%84%D0%B8%D0%B7%D0%B8%D0%BA%D0%B0), [баллистику](http://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%BB%D0%BB%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0), [вулканологию](http://ru.wikipedia.org/wiki/%D0%92%D1%83%D0%BB%D0%BA%D0%B0%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F) и [океанографию](http://ru.wikipedia.org/wiki/%D0%9E%D0%BA%D0%B5%D0%B0%D0%BD%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F). Метод гидродинамики сглаженных частиц является не-сеточным ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *mesh-free*) [лагранжевым методом](http://ru.wikipedia.org/wiki/%D0%9B%D0%B0%D0%B3%D1%80%D0%B0%D0%BD%D0%B6%D0%B5%D0%B2%D0%B0_%D0%BC%D0%B5%D1%85%D0%B0%D0%BD%D0%B8%D0%BA%D0%B0) (то есть координаты движутся вместе с жидкостью), и разрешающая способность метода может быть легко отрегулирована относительно [переменных](http://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F), таких как [плотность](http://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%BE%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C).

**Метод**

Метод SPH работает путем деления жидкости на дискретные элементы, называемые частицами. Эти частицы имеют пространственное расстояние (известное как «длина сглаживания», обычно представляемая в уравнениях как ), на котором их свойства «сглаживаются» [*функцией ядра*](http://ru.wikipedia.org/wiki/%D0%AF%D0%B4%D1%80%D0%BE). Это значит, что любая физическая величина любой частицы может быть получена путем суммирования соответствующих величин всех частиц которые находятся в пределах двух сглаженных длин. Например, [температура](http://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BC%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D1%83%D1%80%D0%B0) в точке  зависит от температуры всех частиц на расстоянии 2 от .



Влияние каждой частицы на свойства оценивается в соответствии с её плотностью и расстоянием до интересующей частицы. Математически, это описывается функцией ядра (обозначается ). В качестве функции ядра часто используют [функцию Гаусса](http://ru.wikipedia.org/w/index.php?title=%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_%D0%93%D0%B0%D1%83%D1%81%D1%81%D0%B0&action=edit&redlink=1) (функция [нормального распределения](http://ru.wikipedia.org/wiki/%D0%9D%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%80%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5)) или [кубический сплайн](http://ru.wikipedia.org/wiki/%D0%9A%D1%83%D0%B1%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%81%D0%BF%D0%BB%D0%B0%D0%B9%D0%BD). Последняя функция равна нулю для частиц находящихся дальше чем две сглаженные длины (в отличие от функции Гаусса, где имеется небольшое влияние на любом конечном расстоянии). Это позволяет экономить вычислительные ресурсы, исключая относительно малое влияние отдаленных частиц.



Значение любой физической величины  в точке , задаётся формулой:



Формулы договаривались нумеровать как «номер раздела.номер формулы в разделе». Те формулы, на которые вы не ссылаетесь в тексте, можно не нумеровать (забыл сказать).

(3)



где  — [масса](http://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81%D0%B0) частицы j,  — значение величины *A* для частицы *j*,  — [плотность](http://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%BE%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C) связанная с частицей *j*, и *W* — функция ядра упомянутая выше. Например, плотность частицы () может быть выражена как:



(4)



где суммирование  включает все частицы в симуляции.



Аналогично, пространственная производная количества может быть получена используя [интегрирование по частям](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%BF%D0%BE_%D1%87%D0%B0%D1%81%D1%82%D1%8F%D0%BC) для смещения [оператора набла](http://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80_%D0%BD%D0%B0%D0%B1%D0%BB%D0%B0) () от физической величины к функции ядра:



(5)



Хотя размер длины сглаживания может быть фиксирован и в пространстве, и во времени, это не позволяет использовать всю мощь SPH. Назначая каждой частице её собственную длину сглаживания и разрешая ей меняться со временем, разрешающая способность симуляции может автоматически подстраивать себя к локальным условиям. Например, в очень плотной области, где много частиц расположены близко одна к другой, длина сглаживания может быть сделана относительно короткой, приводя к высокому пространственному разрешению. И наоборот, в областях с малой плотностью, где частицы размещены далеко одна от другой и разрешающая способность низкая, длина сглаживания может быть увеличена, оптимизируя вычисления для данной области. Объединённая с [уравнением состояния](http://ru.wikipedia.org/wiki/%D0%A3%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5_%D1%81%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D1%8F) и [интегратором](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D1%82%D0%BE%D1%80_(%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE)), гидродинамика сглаженных частиц может эффективно симулировать гидродинамические потоки. Однако традиционная искусственная формулировка вязкости, используемая в гидродинамике сглаженных частиц, имеет тенденцию «смазывать» ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) smear out) [ударные волны](http://ru.wikipedia.org/wiki/%D0%A3%D0%B4%D0%B0%D1%80%D0%BD%D0%B0%D1%8F_%D0%B2%D0%BE%D0%BB%D0%BD%D0%B0) и контактные разрывы в гораздо большей степени, чем современные основанные на сетках методы.

Адаптивность гидродинамики сглаженных частиц, основанной на Лагранжевом подходе, аналогична адаптивности современных сеточных кодов с применением адаптивных сеток [уточнения адаптивной сети](http://ru.wikipedia.org/w/index.php?title=%D0%A3%D1%82%D0%BE%D1%87%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B0%D0%B4%D0%B0%D0%BF%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B9_%D1%81%D0%B5%D1%82%D0%B8&action=edit&redlink=1) ([англ.](http://en.wikipedia.org/wiki/Adaptive_mesh_refinement)) ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Adaptive mesh refinement), хотя в последнем случае можно измельчить сетку согласно любому критерию, выбранному пользователем. Поскольку гидродинамика сглаженных частиц лагранжева по своей природе, то она ограничена в параметрах измельчения, используя только плотность.

**Использование SPH** **в моделировании жидкости**

Гидродинамика сглаженных частиц всё более часто используется для моделирования движения жидкостей. Это происходит из-за некоторых **преимуществ метода SPH** по сравнению с традиционными основанными на сетке методиками:

**Во-первых**, SPH гарантирует сохранение массы без дополнительных вычислений, так как частицы сами по себе представляют массу.

**Во-вторых**, SPH вычисляет давление от воздействия соседних частиц, также имеющих массу, а не решает систему линейных уравнений.

**Наконец**, в отличие от основанных на сетке методик, которые должны прослеживать границы жидкости, SPH создаёт свободную поверхность для непосредственно двухфазных взаимодействующих жидкостей, так как частицы представляют более плотную жидкость (обычно воду), а свободное пространство представляет более лёгкую жидкость (обычно воздух). По этим причинам благодаря SPH возможно моделировать движение жидкости в режиме реального времени.

**Единственный недостаток SPH** по сравнению с основанными на сетке методиками состоит в том, что необходимо большое количество частиц для создания симуляции с эквивалентной разрешающей способностью. В типичной реализации основанных на сетке методик и SPH, много вокселей или частиц будут находиться под поверхностью воды, в глубине водяного объёма, и никогда не будут визуализированы. Однако точность может быть значительно увеличена со сложными основанными на сетке методиками, особенно с теми, которые используются совместно с методами частиц (такими, как наборы уровней частиц).

Для некритических приложений, таких как компьютерные игры и кинофильмы, производительность и визуальный реализм намного более значимы, чем вычислительная точность. Muller и другие использовали SPH для симуляции воды, которая течёт в стакан. При этом использовалось несколько тысяч частиц, а частота кадров составляла около 5 кадров/сек. Kipfer и Westermann (Technical University at Munich, Germany) использовали SPH для симуляции реки. Такахиро Харада ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Takahiro Harada*) и другие использовали современные графические процессоры GeForce 8800 GTX для симуляции 49 153 частиц со скоростью 17 кадров/сек.

**Вывод**

На основе анализа было принято решение реализовать метод Smoothed Particle Hydrodynamics или Гидродинамики сглаженных частиц.

**1.2. Углубленный анализ метода SPH**

Для того, чтобы ясно и чётко понять как работает метод SPH было проанализировано довольно много статей, публикаций и исследований.

Наиболее качественными оказались статьи:

«Particle-Based Fluid Simulation for Interactive Applications» Matthias Müller, David Charypar and Markus Gross.

«Smoothed particle hydrodynamics» *Monaghan J*.*J*.

«Simulating free surface flows with SPH» *Monaghan J*.*J*.

«Метод гидродинамики сглаженных частиц» И.В. Абрамов, М.А. Алексеев, Д.О. Левченко, С.А. Моргунов.

«Реализация метода SPH на CUDA для моделирования несжимаемых жидкостей» Суравикин А. Ю.

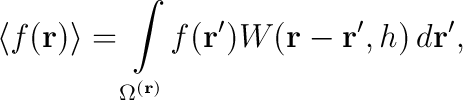
«Simulation and Rendering of a Viscous Fluid using Smoothed Particle Hydrodynamics» Marcus Vesterlund.

**Основа метода**

В основе метода сглаженных частиц лежит аппроксимация поля набором частиц. Чтобы обеспечить такую аппроксимацию, используется интегральная форма записи которая «сглаживает» области между частицами с помощью специальной функции-ядра. Определим некоторую область сглаживания , где kh – длина сглаживания. Параметр k – зависит от функции-ядра, а h выбирается в зависимости от радиуса частицы. Тогда аппроксимация функции f(r) будет выглядеть так:

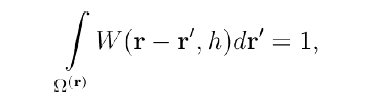


(6)



, где W – функция-ядро, которая обладает свойствами:

(7)



(8)



(9)



где *δ*(r) — дельта-функция Дирака. Функция-ядро также устанавливает параметр *k* в формуле длины сглаживания *kh*. Пример функции-ядра с *k* = 2:

(10)



Коэффициент зависит от размерности эксперимента D и длины сглаживания:



(11)



Каждая частица описывается набором параметров: положение, скорость, масса.

Помимо этих параметров есть второстепенные параметры, такие как плотность и давление.

При моделировании жидкостей методом SPH требуется на каждом временном шаге моделирования вычислять силу воздействия F*i* на каждую частицу *i*:

F*i* = F*pres* + F*visc* + F*tens* + mg (12)

, где **F***pres* — сила давления, **F***visc* — сила вязкости, **F***tens* — сила поверхностного натяжения, m**g** — сила гравитации. При моделировании экспериментов больших масштабов (> 10 см) силой поверхностного натяжения можно пренебречь.

После вычисления **F***i* производится обновление векторов скоростей (за счет ускорения, приданного силой воздействия) и позиций частиц.

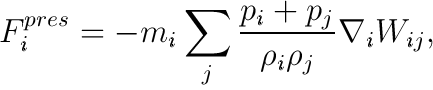
F=ma; dv=a; => dv=F/m; (13a)

dx=v; (13b)

Для вычисления сил взаимодействия между частицами требуется обеспечивать симметрию воздействия, то есть для всех частиц *i* и *j* должно выполняться равенство **F***ij* = –**F***ji*.

**Сила давления** вычисляется~~, например,~~ по формуле

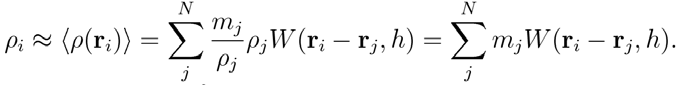
(14)



где *j* — соседи частицы *i*, *pi*, *pj,* — давление частиц, которое вычисляется в зависимости от типа жидкости.

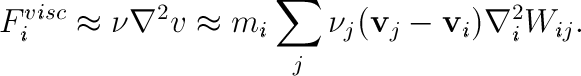
**Плотность частицы** *ρi* вычисляется с помощью аппроксимации ядром функции *f* (**r**)=*ρ*(**r**) [5] (переместил сюда, сделайте так же и для остальных формул):

(15)



**Вычисление силы вязкости** допускает несколько подходов в вычислении. Простейшим с точки зрения вычислительных затрат является вычисление аппроксимации оператора Лапласа:

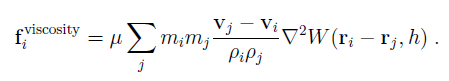
(16)



[5]

**Более правильной формулой для вычисления вязкости является**

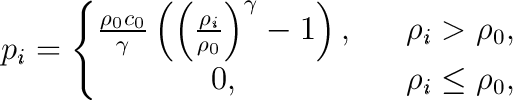
(17)



[6]

**Вычисление давления.** Для моделировании (я) несжимаемой жидкости существует несколько подходов, которые отличаются, именно, способом вычисления сил давления. Один из таких подходов — «слабо-сжимаемый» метод сглаженных частиц (WCSPH, Weakly Compressible SPH). Он основан на вычислении давления для частицы *i* по формуле:

(18)



где *γ* = 7, *c*0 — коэффициент жесткости, равный максимально возможной скорости частицы в эксперименте, *ρ*0 — исходная плотность жидкости, *ρi* — плотность *i*-й частицы. Заметим, что если в некоторой точке происходит разрежение среды, то есть *ρi* < *ρ*0, то вычисленное давление станет отрицательным. В таком случае для этой точке (и) устанавливаем давление, равное нулю.

[16]

**Ещё один способ вычисления давления** основывается на законах идеального газа и записывается в простейшей форме:

(19)



, где p0 – нулевое давление, а k выбирается из соображений скорости звука.

[6]

**1.3. Выбор метода визуализации**

Как уже было сказано: и SPH, и основанные на сетке методики всё ещё нуждаются в визуализируемой свободной поверхностной геометрии и используют полигонизационные методики, такие как [metaballs](http://ru.wikipedia.org/wiki/Metaballs), [marching cubes](http://ru.wikipedia.org/wiki/Marching_cubes), point splatting или «ковровую» визуализацию («carpet» visualization).

**Metaball**

([рус.](http://ru.wikipedia.org/wiki/%D0%A0%D1%83%D1%81%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Метасфера*, также встречается «метаболл») — n-мерный объект в [компьютерной графике](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%BA%D0%B0), представляющий собою замкнутую сглаженную поверхность. Техника [рендеринга](http://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BD%D0%B4%D0%B5%D1%80%D0%B8%D0%BD%D0%B3) метасфер была изобретена [Джимом Блинном](http://ru.wikipedia.org/w/index.php?title=%D0%91%D0%BB%D0%B8%D0%BD%D0%BD,_%D0%94%D0%B6%D0%B8%D0%BC&action=edit&redlink=1) ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Jimm Blinn*) в начале 1980-х годов.

Использование полигонов в компьютерной графике часто дает несглаженные модели, причем степень сглаженности сильно зависит от масштаба. Для получения гладких поверхностей используются различные методы, такие как [B-сплайны](http://ru.wikipedia.org/wiki/B-%D1%81%D0%BF%D0%BB%D0%B0%D0%B9%D0%BD) и [поверхности Безье](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B2%D0%B5%D1%80%D1%85%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D0%91%D0%B5%D0%B7%D1%8C%D0%B5). При использовании метасфер подразумевается, что в пространстве задано множество управляющих точек, обладающих потенциалом, и заданы функции зависимости потенциала от расстояния. Вычисляя потенциал поля, можно построить сглаженные изоповерхности довольно сложной формы.

На рисунке 1 (такого нет) - взаимодействие двух положительных метасфер, смоделированное в [Bryce](http://ru.wikipedia.org/wiki/Bryce).  
Обратите внимание, как две маленьких метасферы сливаются в одну большую.

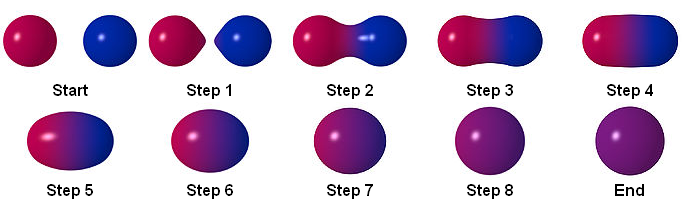


Рисунок 1.1

**Способ задания**

Каждая управляющая точка определяет собственную n-мерную потенциальную [функцию](http://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) Ui(x, y, z) (обычно n=3). Затем выбирается некое значение U0 (потенциал), которое определяет форму метасферы (фактически, определяется [эквипотенциальная поверхность](http://ru.wikipedia.org/wiki/%D0%AD%D0%BA%D0%B2%D0%B8%D0%BF%D0%BE%D1%82%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BE%D0%B2%D0%B5%D1%80%D1%85%D0%BD%D0%BE%D1%81%D1%82%D1%8C)). Таким образом, неравенство

(20)



определяет, находится ли точка (x, y, z) внутри поверхности, заданной  управляющими точками, или нет.



Часто в качестве функции, задающей метасферу, используют

(21)



, где (x0, y0, z0)   — центр метасферы. Однако использование деления делает эту функцию неэффективной по скорости, поэтому обычно ее заменяют аппроксимирующими полиномиальными функциями.

При поиске **более эффективной функции** потенциала желательно, чтобы она удовлетворяла следующим **требованиям**:

**Компактность**[**носителя**](http://ru.wikipedia.org/wiki/%D0%9D%D0%BE%D1%81%D0%B8%D1%82%D0%B5%D0%BB%D1%8C_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8). Такая функция обращается в ноль за пределами некой ограничивающей сферы. При вычислении поля, создаваемого управляющей точкой, нет необходимости вычислять его, когда расстояние превышает радиус ограничивающей сферы. Иерархическая система отсечений может, таким образом, сильно сократить количество управляющих точек, необходимых для расчета поля в данной точке.

**Гладкость**. Поскольку метасфера является результатом суперпозиции полей управляющих точек, ее гладкость зависит от гладкости функции потенциала.

Простейшая функция потенциала, удовлетворяющая этим критериям:

(22)



, где  – расстояние между управляющей точкой и заданной точкой пространства. Она также довольно эффективна, поскольку не использует деления и извлечения корня.



Более сложные модели используют для лучшего сглаживания гауссов потенциал, ограниченный конечным радиусом набора многочленов.

Есть множество способов рендеринга метасфер. Для трехмерных метасфер чаще всего применяют [**рэйкастинг**](http://ru.wikipedia.org/wiki/Ray_casting) и алгоритм [marching cubes](http://ru.wikipedia.org/wiki/Marching_cubes) [15].

**Marching cubes**

Marching cubes ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) движущиеся кубики) — [алгоритм](http://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) в [компьютерной графике](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%BA%D0%B0), впервые предложенный в [1987 году](http://ru.wikipedia.org/wiki/1987_%D0%B3%D0%BE%D0%B4) на конференции [SIGGRAPH](http://ru.wikipedia.org/wiki/SIGGRAPH) Вильямом Лоренсеном ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) William E. Lorensen) и Харви Клайном ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Harvey E. Cline), для обработки полигональной сетки [изоповерхности](http://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D0%B0%D0%BB%D1%8F%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D0%BE%D0%BB%D0%B5#.D0.9F.D0.BE.D0.B2.D0.B5.D1.80.D1.85.D0.BD.D0.BE.D1.81.D1.82.D1.8C_.D1.83.D1.80.D0.BE.D0.B2.D0.BD.D1.8F) трехмерного [скалярного поля](http://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D0%B0%D0%BB%D1%8F%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D0%BE%D0%BB%D0%B5) (чаще называемой сеткой [вокселей](http://ru.wikipedia.org/wiki/%D0%92%D0%BE%D0%BA%D1%81%D0%B5%D0%BB)). Аналогичный алгоритм на плоскости называется [marching squares](http://ru.wikipedia.org/wiki/Marching_squares).

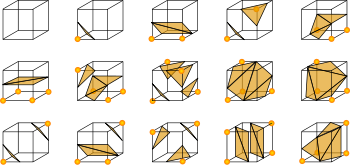
**Принцип работы**

Алгоритм пробегает скалярное поле, на каждой итерации просматривает 8 соседних позиций (вершины [куба](http://ru.wikipedia.org/wiki/%D0%9A%D1%83%D0%B1), параллельного осям координат) и определяет полигоны, необходимые для представления части изоповерхности, проходящей через данный куб. Далее на экран выводятся полигоны, образующие заданную изоповерхность.

Так как алгоритм выбирает полигоны, исходя только из положения вершин куба относительно изоповерхности, всего получается 256 () возможных конфигураций полигонов, которые можно вычислить заранее и сохранить в массиве. Поэтому каждый куб можно представить восьмибитным числом, сопоставив каждой вершине 1, если значение поля в точке больше, чем на изоповерхности, и 0 в противном случае. Полученное число используется в качестве индекса элемента массива, хранящего конфигурации полигонов. Наконец каждая вершина сгенерированного полигона помещается в подходящую позицию на том ребре куба, на котором она лежала изначально. Позиция вычисляется путем линейной интерполяции значений скалярного поля в концах ребра.



15 различных конфигураций куба



Заранее вычисленный массив из 256 конфигураций полигонов можно получить поворотами и отражениями 15 различных конфигураций куба. Однако использование всего 15 базовых конфигураций не гарантирует получение замкнутой поверхности. Базовые конфигурации, лишенные этого недостатка, можно найти в специальной литературе.

[Градиент](http://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D0%B4%D0%B8%D0%B5%D0%BD%D1%82) скалярного поля в каждой точке сетки также является нормальным вектором предполагаемой изоповерхности, проходящей через эту точку. Поэтому возможно [интерполировать](http://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D0%B0%D1%8F_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D0%BE%D0%BB%D1%8F%D1%86%D0%B8%D1%8F) эти нормали вдоль ребер каждого куба, чтобы найти нормали сгенерированных вершин, для корректного отображения модели при использовании освещения.

Данный алгоритм широко используется в медицине, например, в [компьютерной](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D1%82%D0%BE%D0%BC%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F) и [магнитно-резонансной томографии](http://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D0%B3%D0%BD%D0%B8%D1%82%D0%BD%D0%BE-%D1%80%D0%B5%D0%B7%D0%BE%D0%BD%D0%B0%D0%BD%D1%81%D0%BD%D0%B0%D1%8F_%D1%82%D0%BE%D0%BC%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F), а также для 3D моделирования [метасфер](http://ru.wikipedia.org/wiki/Metaballs) или других метаповерхностей.

[15]

**Ray casting**

Ray casting, рейкастинг, Метод «бросания лучей» ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) ray casting — [рус.](http://ru.wikipedia.org/wiki/%D0%A0%D1%83%D1%81%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) бросание лучей) — один из [методов](http://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4) [рендеринга](http://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BD%D0%B4%D0%B5%D1%80%D0%B8%D0%BD%D0%B3) в [компьютерной графике](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%BA%D0%B0), при котором сцена строится на основе замеров пересечения лучей с визуализируемой поверхностью. Этот термин впервые использовался в компьютерной графике в 1982 году в публикации Скотта Рота ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Scott Roth), который применил его для описания метода рендеринга [CSG](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%B0%D1%8F_%D1%81%D0%BF%D0%BB%D0%BE%D1%88%D0%BD%D0%B0%D1%8F_%D0%B3%D0%B5%D0%BE%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%8F)-моделей.

Хотя термины «ray casting» (метод бросания лучей) и «ray tracing» ([трассировка лучей](http://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D1%81%D1%81%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0_%D0%BB%D1%83%D1%87%D0%B5%D0%B9)) часто использовались как синонимы в ранней литературе, посвященной компьютерной графике, в современной компьютерной графике эти термины различны и описывают разные методики, которые, тем не менее, имеют много общего.

Алгоритм

Рейкастинг не является синонимом к рейтрейсингу (трассировке лучей), но он может быть представлен как сокращённая и существенно более быстрая версия алгоритма трассировки лучей. Оба алгоритма являются «image order» и используются в компьютерной графике для рендеринга трёхмерных сцен на двухмерный экран с помощью проекционных лучей, которые проецируются от глаз наблюдателя к [источнику света](http://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA_%D1%81%D0%B2%D0%B5%D1%82%D0%B0).

Метод бросания лучей не вычисляет новые тангенсы лучей света, которые возникнут после того, когда луч, который проецируется от глаза к источнику света, пересечётся с поверхностью. Эта особенность делает невозможным точный рендеринг отражений, преломлений и естественной проекции теней с помощью рейкастинга. Однако все эти особенности могут быть добавлены с помощью «фальшивых» (обманных, аппроксимационных) методик, например, через использование текстурных карт или другие методы. Высокая скорость вычисления сделала рейкастинг удобным методом рендеринга в ранних компьютерных играх с трёхмерной графикой реального времени.

[18]

В выделенном интервал одинарный.

**Вывод**

После анализа существующих методов было решено реализовать метод визуализации рейкастинг.

Представлять жидкость в пространстве было решено несколькими способами:

1 Для получения общего представления об объёме жидкости заменять частицы сферами.

2 Заменять частицы сферами, но использовать для подсчёта нормалей поверхности дополнительные данные о плотности.

3 Заменять частицы метасферами с функцией U = W.

1. **Конструкторский раздел.**

**2.1. Структуры данных**

struct **Vector3DF** {

float X;

float Y;

float Z;

}

**Структура параметров частицы.**

struct **structParticle** {

Vector3DF Pos; //Положение частицы

Vector3DF V; //Вектор скорости частицы

Vector3DF F; //Вектор силы, действующей на частицу

float Density; //Плотность поля в точке Pos

float P; //Давление в точке Pos

}

**Структура для хранения информации о частицах-соседях.** В ней хранится дополнительная информация, ускоряющая вычисления.

struct **structNeighbours** {

int i; //индекс первой частицы

int j; //индекс второй частицы (i < j)

float dist; //Расстояние между частицами

float W; //Значение функции ядра для этих двух частиц

}

Частицы жидкости хранятся в одномерном массиве структур structParticle[].

Для хранения информации о соседях также используется массив structNeighbours[].

Замечу, что в массиве structNeighbours[] не хранятся пары (A B) и (B A), т.к. все формулы расчетов взаимодействий между частицами подобраны так, что Fab = Fba. Таким образом объем вычислений уменьшается вдвое.

**2.2. Алгоритм SPH**

На основе анализа метода SPH была составлена схема алгоритма, рассчитывающего новое положение частиц в зависимости от старого.

Алгоритм работает с массивом частиц structParticle[]. Помимо массива частиц необходимо задание констант, которые используются в формулах SPH. Также используется класс clsKernel для расчета значений функции ядра. В результате работы алгоритма в массиве частиц обновляются все параметры, такие как положение, текущая скорость, плотность в точке, действующие на частицу силы.

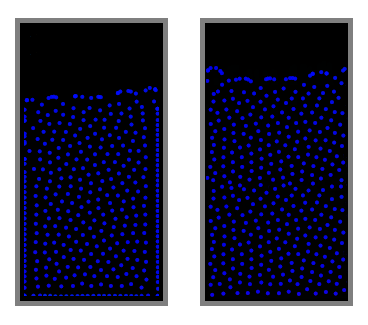
Блок-схема алгоритма находится в приложении номер таком-то.

**2.3. 2D жидкость**

Для начала было решено протестировать метод моделирования жидкости SPH на 2D модели жидкости.

В процессе отладки 2-мерного варианта были сделаны некоторые выводы и внесены некоторые коррективы:

**Поведение частиц жидкости рядом со стенкой. Моделирование стен.**



Изначально моделирование стенок сцены описывалось в алгоритме условием:

Если (частица.х < 0),

то частица.x = 0;

и т.д.

На первом изображении ясно видно, что у стен сосуда скапливается гораздо большее количество частиц, нежели в основном объёме. Это происходит из-за того, что алгоритм SPH пытается достигнуть нулевой плотности во всех точках жидкости.

Рисунок не подписан. Договаривались рисунок текстом не обтекать.

Для решения этой проблемы стенки принято моделировать слоем (толщиной из 3 или 4 частиц) таких же частиц жидкости, только с константным положением.

Т.к. такой метод значительно увеличивает объём вычислений, (из-за увеличения количества частиц) он был изменен. Частицы за стенкой сосуда размещаются только там, где они необходимы. Таким образом, количество вычислений практически не увеличивается.

**2.4. Алгоритмы визуализации**

Для реализации визуализации принято решение использовать raycasting ввиду его довольно большого потенциала и скорости работы. Этот метод легко модифицируется до метода raytrace. Можно достаточно легко добавить отражение, прозрачность сложные объекты в сцену и т.п.

**Алгоритмы рейкастинга для визуализации пересекающихся сфер.**

Схемы находятся в приложении.

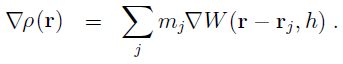
Пересечение луча ведётся с пресекающимися сферами. Сферы располагаются в местах расположения частиц жидкости.

Алгоритм сглаженных сфер отличаются от обычного методом подсчета нормали в точке пересечения луча и сферы. Для алгоритма пересекающихся сфер используется нормаль к поверхности сферы, с которой было найдено пересечение луча. Для алгоритма сглаженных сфер нормаль считается на основе градиента поля плотности.

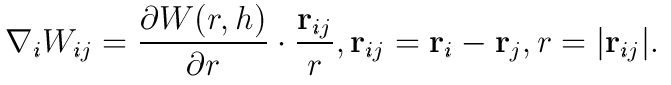
**Алгоритм подсчёта нормали в точке пересечения луча и изоповерхности.**

Чтобы найти нормаль в точке r на изоповерхности необходимо подсчитать градиент поля в этой точке.

Поле плотности рассчитывается по формуле (15). Градиент этого поля будет рассчитываться по следующей формуле:



, где градиент функции W рассчитывается по формуле:



**Алгоритм рейкастинга для визуализации изоповерхностей (метасфер).**

Подсчёт нормали в этом алгоритме совпадает с методом подсчета нормали в алгоритме сглаженных сфер. Но поиск пересечения ведётся не со сферами, а с полем плотности. Точнее: с изоповерхностью поля плотности.

В профессиональных системах визуализации изоповерхностей, для ускорения процесса визуализации, производится вокселизация сцены.

Проанализировав тот объем вычислений, который необходимо выполнить для того, чтобы вокселизовать всю сцену в нормальном разрешении я решил отказаться от этого метода ускорения (что-то я не понял) работы. Был придуман (разработан) следующий способ:

Опять одинарный интервал. До этого еще в паре мест встречался.

Луч задается вектором.

Сцена разворачивается так, что наблюдатель находится в точке (0, 0, 0).

Частицы сортируются по удалённости от наблюдателя.

Частицы заменяются сферами, с радиусом, равным радиусу сглаживания.

Производится поиск ближайшей сферы (частицы) и самой удалённой от наблюдателя. Вместе с этими данными сохраняются данные о точках пересечения луча и сфер (MinPoint и MaxPoint).

Затем на промежутке от MinPoint до MaxPoint производится вычисление значений плотности, с некоторым задаваемым шагом и если плотность оказывается выше пороговой плотности, считается, что найдена точка пересечения луча и изоповерхности.

И что? А рисовать когда будем? Или это алгоритм расчета изоповерхности? Пронумеруйте шаги алгоритма.

1. **Технологический раздел**

Каждый раздел начинается с новой страницы.

Для реализации поставленной задачи был выбран язык программирования C#.

Технология программирования – ООП.

**3.1. Структура программы**

**Диаграмма классов**

Structures

clsParticles

frmMain

clsScene

clsKernel

clsRendering

frmRendering

Стрелочки какое отношение показывают (наследование, агрегирование, что-то еще)?

**3.2. Описание классов**

**frmMain**

Основное окно проекта.

Предоставляет графический интерфейс для настройки параметров жидкости и визуализации. Также выводит на экран схематическую модель жидкости в реальном времени.

**clsScene**

Класс объединяющий в себе все настройки и алгоритмы для работы жидкости, камеры, визуализации и физики.

**clsRender**

В этом классе находятся алгоритмы визуализации жидкости.

**clsParticles**

Параметры системы частиц. И алгоритмы физики системы частиц.

**clsKernel**

Класс с функциями ядра.

**Structures**

Класс со структурами данных.

**frmRendering**

Второстепенное окно для вывода результата рендеринга.

**3.3. Интерфейс**

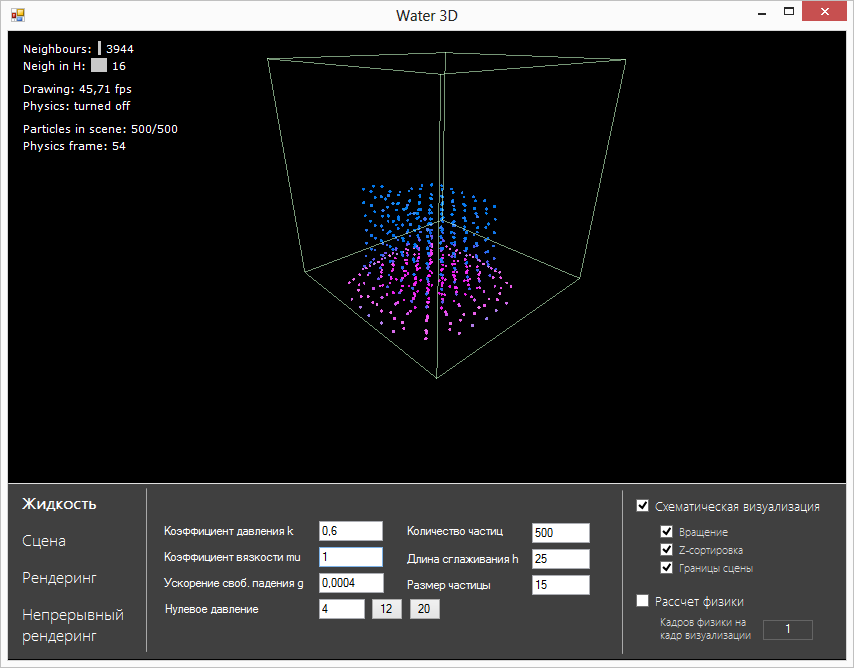


Рисунок не подписан.

**Панель настроек**

Самые часто используемые настройки были расположены статически в правом нижнем углу. Остальные были классифицированы на 4 группы: настройки жидкости, сцены, рендеринга, и непрерывного рендеринга. Эти настройки появляются при наведении мыши на название группы.

**Настройки жидкости** (~~активны на скриншоте~~ показаны на рисунке номер такой-то)

Настройка физических параметров жидкости. Установка максимального количества частиц в сцене.

«Коэффициент давления k» и «нулевое давление» – параметр из формулы 12. Это максимально возможный модуль скорости частицы в модели жидкости (скорость звука) и давление, к которому давление в каждой точке жидкости должно стремиться соответственно.

«Коэффициент вязкости mu» – параметр из формулы 10.

«Длина сглаживания h» - коэффициент функции ядра.

«Размер частицы» - расстояние, на котором необходимо размещать частицы, для того, чтобы достигалось нулевое давление.

**Настройки сцены**

Настройка размеров сцены + настройка «поведения» жидкости. Возможность задать источник частиц или задать начальное положение частиц в сцене для моделирования падения капли воды и т.п.

**Настройки рендеринга**

Установка размеров изображения.

Установка метода визуализации и параметров визуализации

**Настройки непрерывного рендеринга**

Эта область настроек относится к возможностям программы генерировать анимации жидкости. Настраиваются параметры вращения сцены, слежения за жидкостью камерой, также выставляется ограничение количества кадров и возможность завершения работы компьютера после рендеринга анимации.

**Область схематической визуализации**

Моделирование жидкости необходимо каким-либо образом показывать пользователю. В большей части окна приложения отображается система частиц.

Область схематической визуализации позволяет вращать и приближать (отдалять) сцену.

Таким образом выбирается ракурс для последующей качественной визуализации жидкости.

**Текущая информация о работе программы**

В левом верхнем углу выводится информация о скорости работы визуализации и расчетов физики.

Информация о количестве соседей в системе частиц и о максимальном количестве соседей в области сглаживания.

Также: какое количество частиц задействовано в сцене и сколько «кадров» физики было рассчитано.

1. **Исследовательский раздел**

**4.1. Исследование поведения жидкости в зависимости от выбранных функций ядра.**

Необходимо обеспечить как можно более стабильное и реалистичное поведение жидкости. Поэтому были исследованы различные функции ядер. Для подсчёта разных параметров возможно использование различных функций.

На рисунке 4, на обоих изображениях количество частиц и все остальные параметры жидкости (давление, вязкость, длина сглаживания) одинаковые. Различие – в выборе функции ядра для подсчёта сил давления.

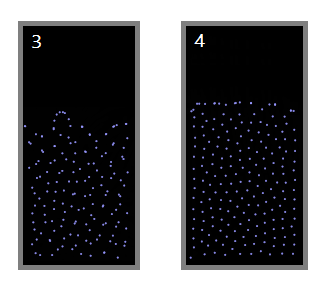
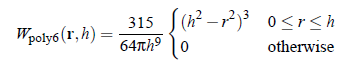


Рисунок 4.1

В первом случае использовалась стандартная функция (W poly6).

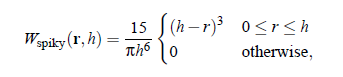
Её недостаток в том, что градиент вблизи нуля почти равен нулю. Из-за этого частицы начинают «склеиваться».

(4.1)



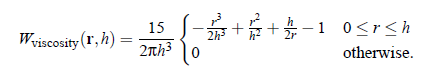
Во втором – «заостренная функция» (W spiky). Она решает проблему функции poly6 и делает расположение частиц гораздо более равномерным, а поведение – более стабильным.

(4.2)



**Для формулы вязкости** возможно использование функций ядра 4.1 или 4.2, но лучше всего работает функция 4.3. Она порекомендована статьей [1].

(4.3)



Влияние функций ядра на поведение жидкости хорошо видно в тестовой программе.

**4.2. Исследование скорости вычисления физики жидкости.**



График 4.1

На графике две кривые т.к. замеры производились для различных «состояний» жидкости.

Тот, который выше – частицы были хаотично разбросаны по сцене.

Который ниже – частицы сформированы в «каплю». За счёт этого возросло количество «соседей» и соответственно количество необходимых вычислений.

**4.3. Исследование скорости генерации изображения в зависимости от кол-ва частиц каждым алгоритмом.**



График 4.2

Из графиков видно, что скорость генерации изображения линейно зависит от количества частиц. И то, что метод генерации сглаженных сфер и метод изоповерхности отличаются в 2 - 2,5 раза, в зависимости от расположения частиц в сцене.

**4.4. Исследование реалистичности визуализации.**

**Сравнение алгоритма пересекающихся сфер и сглаженных сфер.**

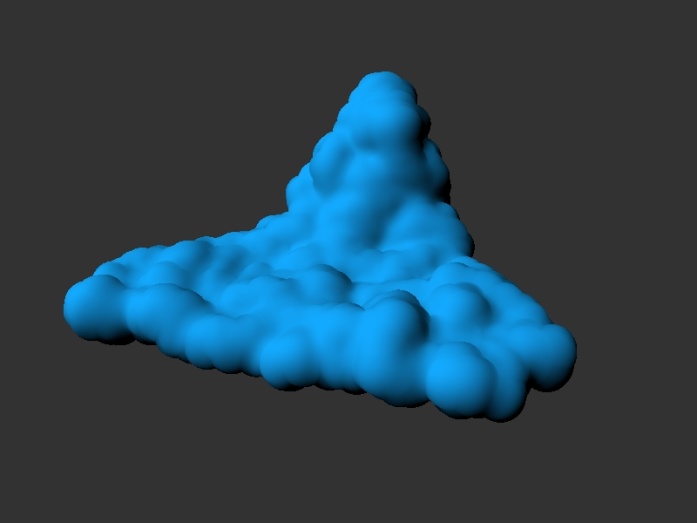
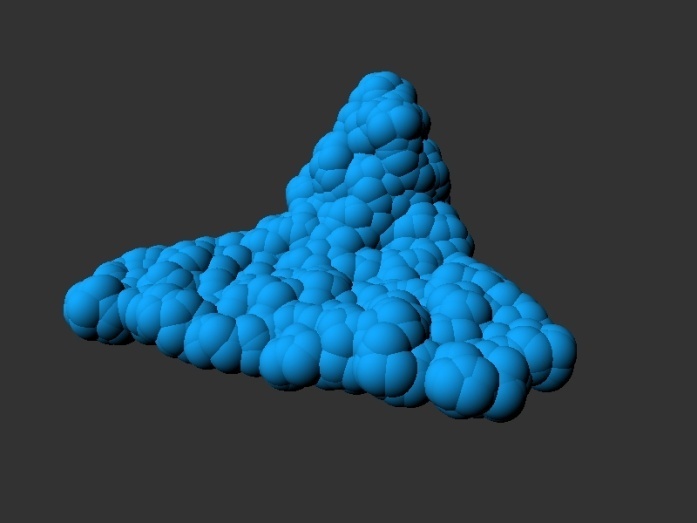


Рисунок 4.1

При размере сфер примерно равном удвоенному размеру частиц сглаживание получается вполне адекватное и красивое. Заметно улучшается качество картинки.

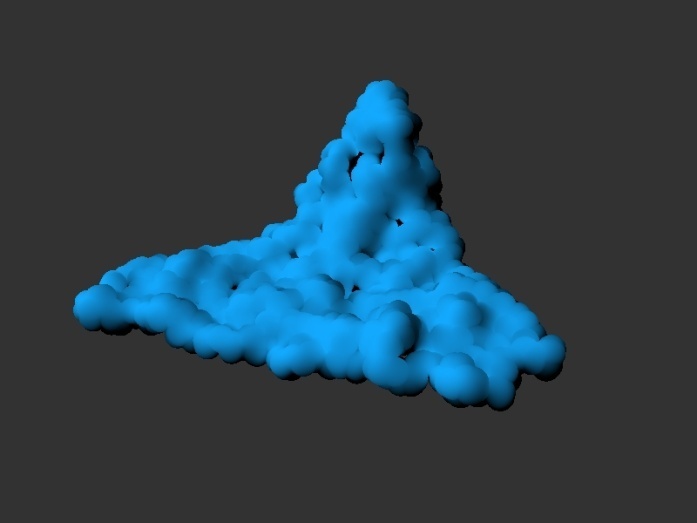
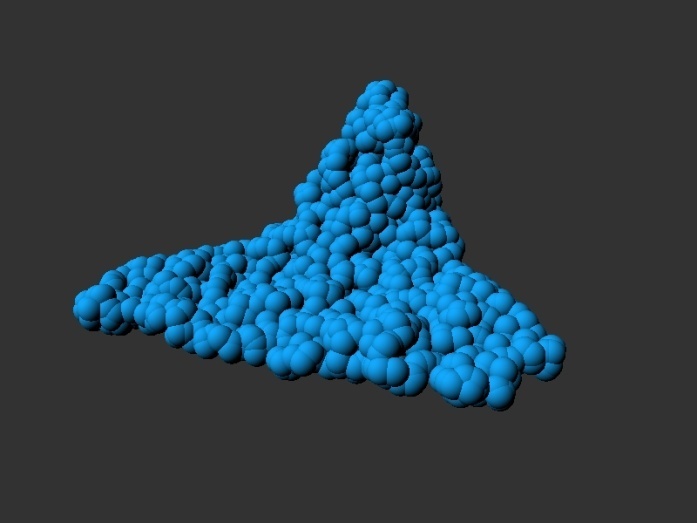


Рисунок 4.2

Сравнение при меньшем размере сфер. (размер сфер = размер частиц)

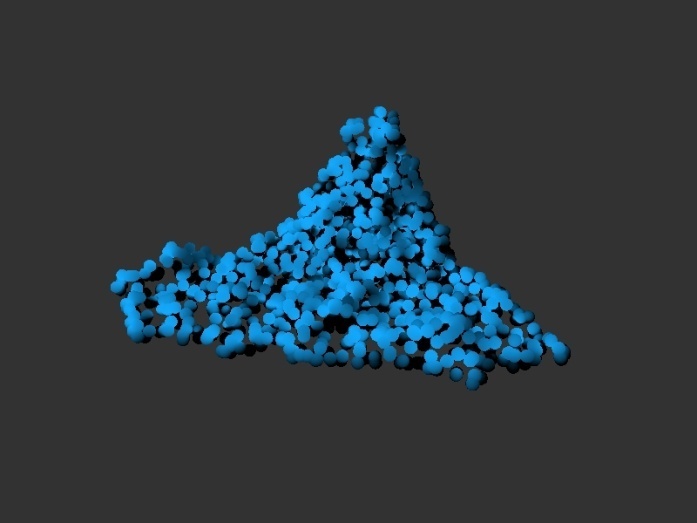
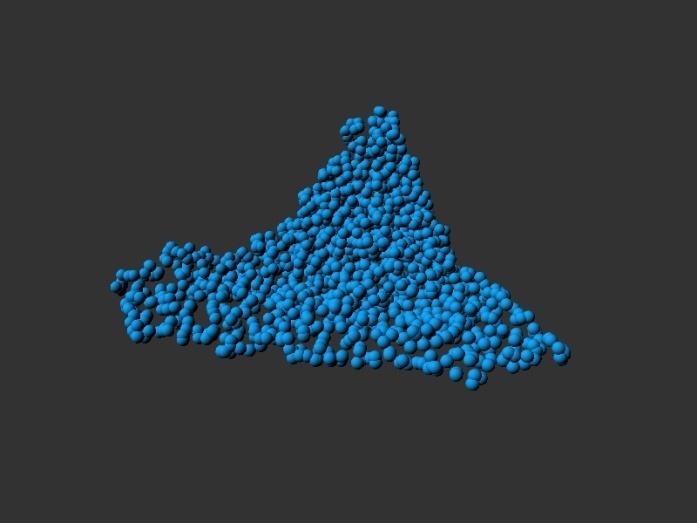


Рисунок 4.3

Размер сфер = размер частиц / 2

Как видно на изображениях выше. При размере сфер, равном размеру частиц в некоторых местах, где сферы недостаточно перекрывают друг друга, появляются артефакты в виде «дырок». Артефакты ещё более заметны при размере сфер вдвое меньшем, нежели размер частиц.

**Сравнение алгоритма сглаженных сфер и алгоритма, визуализирующего изоповерхность.**

Для удобства приведены 4 изображения.

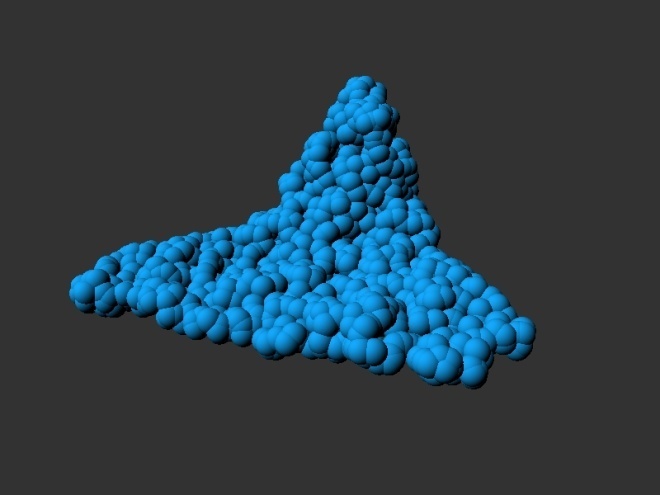
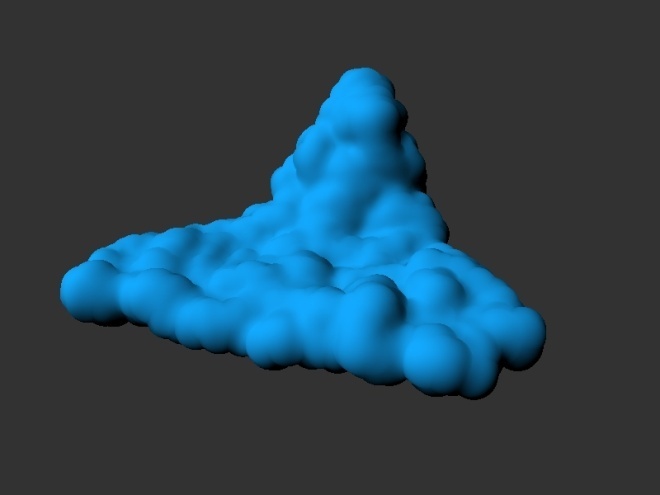
а. Сглаженные сферы. Размер сфер = Размер частиц \* 2

б. Пересекающиеся сферы. Размер сфер = Размер частиц

в. Сглаженные сферы. Размер сфер = Размер частиц

г. Метод изоповерхности. Пороговое значение = 0.8 \* Нулевая плотность.

а. б.



в. г.

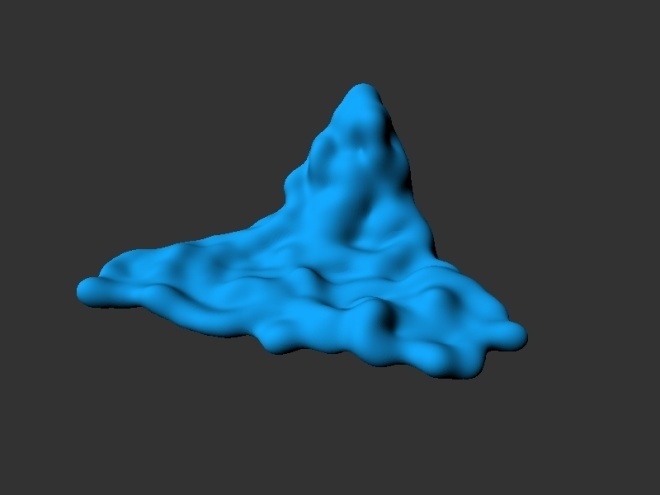
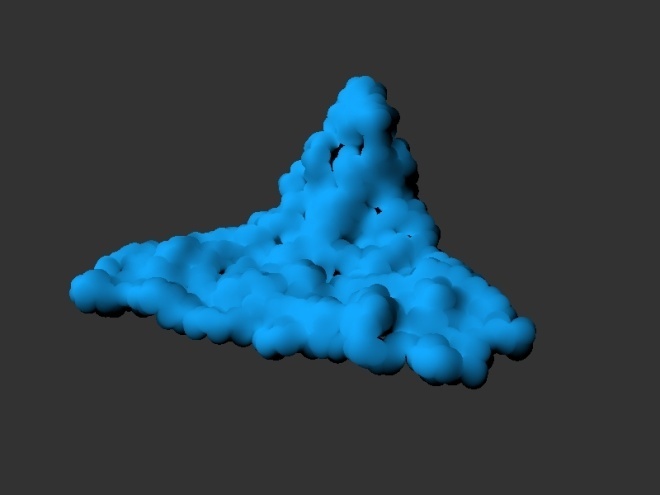


Рисунок 4.4

Сразу заметно, что метод визуализации изоповерхности гораздо более реалистичный.

Метод визуализации изоповерхности в 2 раза медленнее метода сглаженных сфер. Для получения изображения, похожего на 4.4г методом сглаженных сфер необходимо уменьшить размер сфер примерно вдвое, следовательно, увеличить плотность вдвое. Тогда количество частиц необходимо увеличить в 23=8. Если обратиться к графику зависимости времени генерации изображения от количества частиц, то становится ясно, что времени потребуется в 8 раз больше.

Таким образом получается, что метод визуализации изоповерхности эффективнее (примерно в 4 раза) метода сглаженных сфер.

**Заключение**

Разработано программное обеспечение для моделирования жидкости. ПО предоставляет возможности гибкой настройки параметров жидкости. Как физических, так и параметров её визуализации.

**Плюсы**

Реалистичное поведение жидкости и её визуализация

**Минусы**

Необходимость в очень большом количестве частиц для реалистичного поведения жидкости.

Очень долгое время построения качественного изображения жидкости.

**~~Дальнейшие планы.~~**

Это можно не писать.

1. ~~Ускорить скорость работы алгоритмов моделирования физики частиц.~~
2. ~~Ускорить рендеринг (попробовать скомбинировать существующий метод визуализации изоповерхности с вокселизацией сцены)~~
3. ~~Добавить прозрачность и отражение.~~
4. ~~Добавить возможность добавления твёрдых объектов в сцену.~~

**Список литературы**

[1] «Particle-Based Fluid Simulation for Interactive Applications» Matthias Müller, David Charypar and Markus Gross.

[2] «Smoothed particle hydrodynamics» *Monaghan J*.*J*.

[3] «Simulating free surface flows with SPH» *Monaghan J*.*J*.

[4] «Метод гидродинамики сглаженных частиц» И.В. Абрамов, М.А. Алексеев, Д.О. Левченко, С.А. Моргунов.

[5] «Реализация метода SPH на CUDA для моделирования несжимаемых жидкостей» Суравикин А. Ю.

[6] «Simulation and Rendering of a Viscous Fluid using Smoothed Particle Hydrodynamics» Marcus Vesterlund.

[7] «Ray Tracing Implicit Surfaces» John C. Hart

[8] «МОДЕЛИРОВАНИЕ ВОЛНОВЫХ ПРОЦЕССОВ МЕТОДОМ СГЛАЖЕННЫХ ЧАСТИЦ (SPH)» © *2009 г. А.П. Потапов, С.И. Ройз, И.Б. Петров*

[9] «Interactive Ray Tracing for Isosurface Rendering»

Steven Parker Peter Shirley Yarden Livnat Charles Hansen Peter-Pike Sloan

[10] «Wavelet-based Multiresolution Isosurface Rendering» Markus Steinberger and Markus Grabner

[11] «Smoothed particle hydrodynamics, a meshfree particle method.» Liu G. R., Liu M. B.

[12] URL: http://ru.wikipedia.org/wiki/Гидродинамика\_сглаженных\_частиц

[13] URL: http://ru.wikipedia.org/wiki/Метод\_решёточных\_уравнений\_Больцмана

[14] URL: http://ru.wikipedia.org/wiki/Metaballs

[15] URL: http://ru.wikipedia.org/wiki/Marching\_cubes

[16] Landau L.D., Lifshits E.M. *Teoreticheskaia fizika. T. 6. Gidrodinamika* [Theoretical physics. Vol. 6. Hydrodynamics]. Moscow, Fizmatlit Publ., 2006. 736 p.

[17] URL: http://ru.wikipedia.org/wiki/Моделирование\_жидкости

[18] URL: <http://ru.wikipedia.org/wiki/Ray_casting>

[19] «Real-time Rendering of Point BasedWater Surfaces» KeiI wasaki1, Yoshinori Dobashi2, Fujiichi Yoshimoto1, and Tomoyuki Nishita3