Физическое моделирование деформируемых поверхностей в игровых и анимационных приложениях. Анимация водной поверхности.

Маркичев А. С., Супиков А.М.
Нижегородская Лаборатория Программных Технологий, ЗАО, Нижний Новгород, Россия

Аннотация

В данной работе рассматривается вопрос о физическом подходе к моделированию деформируемых поверхностей с целью анимации в режиме реального времени различных явлений, таких как, например, волны на поверхности воды.

На примере моделирования водной поверхности показывается, что если требуется получить не физически точную, реальную модель рассматриваемых явлений, а лишь приемлемую в смысле зрительного восприятия как, например, в игровых приложениях, то во многих случаях можно получить хороший результат, вводя более простую и менее дорогую по вычислениям динамическую модель явления, похожего на исходное, но имеющего, может быть, другую физическую природу.

В качестве основной модели в данной работе берется конечно-элементная линейно-упругая динамическая модель, являющаяся, по сути, системой пружин с демпферами, связывающих точечные узлы с сосредоточенной в них массой (spring-mass, пружинно-массовая модель). Она является одной из наиболее подходящих, в силу ее простоты и как следствие небольшой вычислительной сложности. В то же время все преимущества физического подхода сохраняются.

Используя указанный метод моделирования, мы получаем описание динамики изображаемой поверхности в терминах механики (силы, скорости, ускорения), что позволяет сравнительно просто и быстро менять ее текущее состояние (внося возмущения по скорости и/или положению), накладывать ограничения на движение каких-либо точек поверхности (напр., фиксировать) и т.д.

Нами был осуществлен подбор параметров (массы, сосредоточенные в узлах, коэффициенты упругости связывающих эти узлы пружинок и т.д.) пружинно-массовой модели для визуализации движения водной поверхности. При этом, несмотря на существенное отличие процесса, описываемого данной моделью и реальной водой, удалось получить такие эффекты как распространение круговых волн, прохождение волн друг сквозь друга, их отражение от «береговой линии» и т.д.

Для достижения производительности, необходимой в приложениях реального времени, был разработан специально оптимизированный алгоритм, обеспечивающий высокую производительность вычислительной части и зависимое от положения точки зрения упрощение полигональной модели, передаваемой на отображение в 3D API. Реализованное нами на основе алгоритма демонстрационное приложение показывает высокую общую производительность (расчет+отображение) и хорошее визуальное качество.

Ключевые слова: *физическое моделирование, деформируемая поверхность, компьютерная анимация, поверхность воды*

1. ВВЕДЕНИЕ

В настоящее время уровень развития вычислительной техники достиг той отметки, когда физическое моделирование различных явлений перестало быть уделом супердорогих и даже просто дорогих компьютеров. Более того, расчеты физических моделей могут выполняться в реальном времени на обычном среднего уровня домашнем компьютере. Это повлекло за собой всплеск интереса к физическому моделированию в индустрии компьютерных игр, и постепенно использование физических моделей становится здесь стандартом.

Преимущества физического, а точнее динамического подхода к моделированию водной поверхности, кожи живых существ или одежды в анимации очевидны. Описав поведение системы в виде соответствующих дифференциальных уравнений, мы получаем возможность оперировать с моделью в физических терминах, таких как силы, скорости, ускорения. В момент инициализации от нас требуется задать начальные и граничные условия, а при переходе от одного кадра анимации к другому мы просто осуществляем один или несколько шагов численного интегрирования исходной дифференциальной системы, получая соответствующее шагу модельного времени состояние поверхности. Самое же интересное заключается в том, что формирование правых частей уравнения, так же как и начальных условий для

следующего шага интегрирования можно осуществлять непосредственно перед очередным шагом, что дает возможность получить адекватный ответ системы на неизвестные заранее внешние воздействия, возникающие либо в результате изменения в окружающей обстановке, либо в результате ввода пользователя. Причем, в случае физического подхода к моделированию мы получаем довольно большую свободу в выборе представления таких воздействий: позиции, скорости, ускорения либо силы.

Приведем пару простых примеров. Хорошо известен подход к моделированию поверхности кожи с использованием метода применения линейной комбинации преобразований (Character Studio, Kinetix). Этот метод позволяет получить гладкую поверхность, допускает эффективную реализацию, и поддержка его включена даже в DirectX7.0, но метод практически не позволяет анимировать такие эффекты, как складки, или прогиб в результате удара, скажем, какого-либо предмета. Существует большое количество методов анимации поверхности воды, но большинство из них обладают одним из двух недостатков: либо они очень дороги с вычислительной точки зрения, либо эти методы не допускают отработки произвольных заранее неизвестных воздействий, что сказывается на ощущении реалистичности. Но об этом чуть позже.

Итак, преимущества физического подхода к моделированию в анимации и в областях, связанных с виртуальной реальностью, очевидны. Основными аргументами против его использования могут служить с одной стороны сложность реализации, а с другой … относительно высокая вычислительная сложность соответствующих моделей и, как следствие, низкая производительность. Это действительно так в случае научных расчетов, когда требуется высокая точность результатов. В развлекательных же анимационных, игровых и других подобного рода приложениях требуется лишь, чтобы зритель воспринимал это явление, как соответствующее; чтобы, взглянув на экран, он был бы уверен, что это, например, вода, а это – кожа, а вот это - ткань и т.д. В подобных случаях баланса между производительностью и реалистичностью можно достигнуть, выбрав простую и менее дорогую по вычислениям динамическую модель явления, похожего на исходное, но имеющего, может быть, другую физическую природу. При этом все достоинства именно физического подхода сохраняются.

Далее мы достаточно детально покажем это на примере моделирования поверхности воды.

2. АНИМАЦИЯ ПОВЕРХНОСТИ ВОДЫ

Как мы говорили ранее, существует много подходов к моделированию и анимации поверхности воды. Большая часть из них основана на аналитической модели суперпозиции волн, и решение здесь либо задается сразу в виде линейной комбинации тригонометрических функций со специально подобранными коэффициентами, либо получается в результате обратного преобразования Фурье со специально заданным спектром [3]. Выбор спектра и определяет сложность, реалистичность и детализацию модели. Такие модели могут быть как очень сложными и дорогими с вычислительной точки зрения, так и довольно простыми. Все же, чтобы получить достаточно детальную и реалистичную картину, требуется большое количество вычислений, и поэтому в приложениях реального времени стараются провести такие вычисления заранее, получив, например, набор растровых изображений, которые последовательно меняют, чтобы получить анимацию. В более современных работах применяются так называемые методы непериодического покрытия (aperiodic tiling), когда изображаемая поверхность, как из плиток, составляется из набора заранее рассчитанных растровых изображений, а для получения анимации эти изображения меняются местами по специальным законам [4].

Вообще же, сами авторы признают, что метод Фурье-синтеза дает только базовую картину волн на поверхности воды, поэтому хорошо применим для анимации больших участков поверхности воды (морей, больших озер и т.д.). Анимация же небольших участков поверхности, а тем более маленьких водоемов (небольшие озера, бассейны, каналы), находящихся на близком расстоянии от зрителя имеет свою специфику. Для того, чтобы учитывать такие явления, как отражение волн от береговой линии, большие волны, профиль которых не является однозначной функцией (девятый вал и т.д.), волны, возникающие в результате попадания в воду больших объектов или их движения на поверхности (лодки, пловцы, и т.д.), используются всевозможные ухищрения, которые существенно усложняют модель, что в первую очередь сказывается на производительности. Кроме того, набор описанных в литературе эффектов, которые можно получить, модифицируя базовый метод Фурье-синтеза, достаточно ограничен. Получение новых эффектов требует от разработчика в первую очередь хорошей математической подготовки и довольно длительной работы по подбору параметров.

Выбирая подходящую модель, мы ориентировались именно на небольшие водоемы и на достижение анимации характерных базовых эффектов в их динамике: распространение, наложение, отражение волн, прохождение их друг сквозь друга, визуально адекватная реакция на заранее неизвестное воздействие объектов окружающего мира, волны неоднозначного профиля.

Требуется отметить, что поскольку мы не ставили перед собой задачу точного моделирования распространения волн на поверхности именно воды, то модель, которую мы используем, абсолютно не является какой-либо аппроксимацией ни

стохастической модели, ни решения классических дифференциальных уравнений.

Элемент модели деформируемой поверхности.

Используемая нами модель имеет в своей основе структуру узлов, связанных линейно-упругой пружиной, представляющую, по сути, одноосный конечный элемент (Рис. 1). Каждый узел $\mathbf{p_i}$ обладает следующим набором характеристик:

$\mathbf{m_i}$ - масса узла;

$\bar{x}_i(t) = \{x_i(t),\ y_i(t),\ z_i(t)\}$ - декартовы координаты узла;

$\bar{v}_i(t) = d\,\bar{x}_i / dt$ - скорость узла;

$\bar{a}_i(t) = d^2\,\bar{x}_i / dt^2$ - ускорение узла;

$\bar{f}_i(t)$ - сумма всех сил, действующих на узел.

В свою очередь пружина $\mathbf{S_k}$, $\mathbf{k=k(p_i,\ p_j)}$ соединяющая узлы $\mathbf{p_i}$ и $\mathbf{p_j}$, характеризуется следующими величинами: l_k^r - длина пружины в состоянии покоя; l_k - текущая длина пружины; c_k - коэффициент упругости пружины.

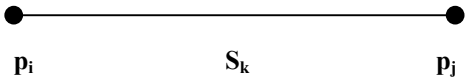

Рис. 1

Отсюда, собственно и следует набор полей для базовой структуры данных.

Выражение для силы пружины $\mathbf{S_k}$, действующей на узел $\mathbf{p_i}$, просто следует из закона Гука:

$$\bar{f}_i^k(t) = c_k\left(l_k - l_k^r\right)\bar{s}_k \qquad (1)$$

где l_k^r, $l_k = \left\|\bar{x}_j - \bar{x}_i\right\|$ - длина покоя и текущая длина пружины соответственно; $\bar{s}_k = \left(\bar{x}_j - \bar{x}_i\right)/l_k$ - единичный вектор в направлении пружины.

На базе таких элементов строятся различные модели: кожа живых существ, слои подкожных тканей, материалы одежды и т.д. Это определяет способ, которым подобные элементы объединяются в графовую структуру. Это могут быть, например, треугольные сетки или слои сеток, связанные между собой, в состоянии покоя они могут быть как регулярными, так и нерегулярными.

Для моделирования поверхности воды мы выбрали следующую структуру.

Структура модели деформируемой поверхности.

Для упрощения дальнейших рассуждений, предположим, что моделируемая поверхность воды в состоянии покоя представляет собой квадрат с введенной регулярной прямоугольной сеткой с некоторым шагом **h** и содержащей **N** x **N** узлов (Рис.2). Узлами нашей модели будут являться узлы сетки, а вот пружинами, соединяющими узлы, необходимо взять не только стороны квадратов сетки, но и их диагонали (Рис.2). Во-первых, в такой модели естественным образом вводится нумерация узлов $(i,j) \mapsto p_{ij}$, $i,j = \overline{1,N}$ (Рис.2).

Во-вторых, структура данных для представления этой модели становится чрезвычайно простой, поскольку структура связности графа пружин и узлов легко восстанавливается, и, следовательно, хранение ее не требуется. То есть для хранения информации о состоянии узлов достаточно двумерного массива, что выгодно отличает данную модель.

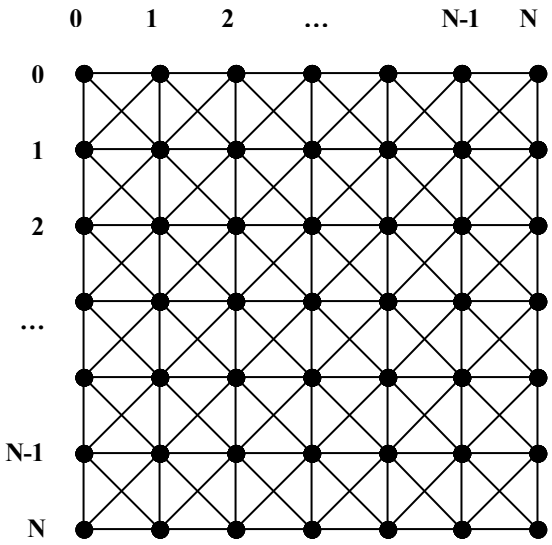

Рис. 2

Во время анимации узлы движутся и их координаты меняются, поэтому геометрическую “регулярность” мы теряем, но “регулярность” структуры связности останется, так как сама структура связности остается неизменной.

Вышеуказанная структура не описывает полностью геометрию поверхности, она задает только набор узлов – точек принадлежащих поверхности и физические связи между узлами, поэтому, вообще говоря, требуется применение алгоритма восстановления поверхности по набору точек. В данном же случае восстановление поверхности сводится к простейшему случаю построения триангуляции по множеству точек. Для получения поверхности мы просто берем четыре точки $\mathbf{p_{ij}}$ $\mathbf{p_{i+1j}}$ $\mathbf{p_{i+1j+1}}$ $\mathbf{p_{ij+1}}$ и формируем на них два треугольника: $\mathbf{p_{ij}p_{i+1j}p_{ij+1}}$ и $\mathbf{p_{ij+1}p_{i+1j}p_{i+1j+1}}$ со смежным ребром $\mathbf{p_{i+1j}p_{ij+1}}$(Рис.3). Таким образом поступаем для всех $i,j = \overline{1,N-1}$. Очевидно, что можно выбирать и другие два треугольника, со смежным ребром $\mathbf{p_{ij}p_{i+1j+1}}$, но в нашем случае это не является принципиальным. Более того, выбор единообразного способа получения двух треугольников по четырем точкам позволяет в

некоторых случаях получать более простые и производительные алгоритмы. Теперь, когда мы по физической модели полностью определили поверхность, следует отметить, что нумерация узлов сетки очевидным образом позволяет ввести параметризацию поверхности, гомеоморфизм квадрата $[0,N]\times[0,N]$ на нашу поверхность. Это свойство является очень полезным, и оно пригодится нам в дальнейшем.

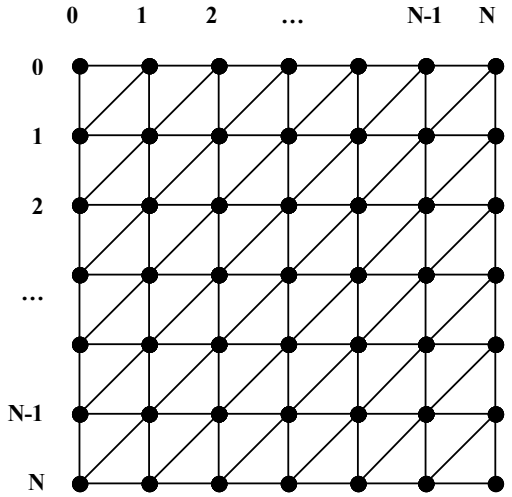

Рис. 3

Теперь можно перейти к динамике.

Уравнение движения модели.

Уравнение движения поверхности в нашей модели есть система обыкновенных дифференциальных уравнений второго порядка, каждое из которых является уравнением движения отдельного узла модели. Уравнение движения (в векторной форме) отдельного узла $\mathbf{p_{ij}}$, связывающего координаты, скорости и ускорения, легко выводится из закона движения Ньютона, входящего в школьный курс физики:

$$m_{ij}\frac{d^2\bar{x}_{ij}}{dt^2}+\gamma_{ij}\frac{d\bar{x}_{ij}}{dt}+\bar{f}_{ij}=\bar{f}_{ij}^{ext}\text{, где}$$

m_{ij} - масса узла;

$\bar{f}_{ij}$ - сумма сил пружин, прикрепленных к узлу;

$\bar{f}_{ij}^{ext}$ - сумма внешних сил, приложенных к узлу;

γ_{ij} - коэффициент демпфирования, характеризующий сопротивление среды, в которой движется узел.

Численное интегрирование уравнений.

Для получения приближенного решения вышеуказанной системы уравнений используется хорошо известная явная схема Эйлера. Пусть шаг интегрирования равен Δt, тогда на очередном шаге алгоритма для значения времени $t+\Delta t$ мы рассчитываем значение ускорения на момент времени t, после чего дважды интегрируем, чтобы получить координаты узла на момент времени $t+\Delta t$.

$$\bar{a}_{ij}(t)=\frac{1}{m_{ij}}\left(\bar{f}_{ij}^{ext}(t)-\gamma_{ij}\bar{v}_{ij}(t)-\bar{f}_{ij}(t)\right)$$
$$\bar{v}_{ij}(t+\Delta t)=\bar{v}_{ij}(t)+\bar{a}_{ij}(t)\Delta t$$
$$\bar{x}_{ij}(t+\Delta t)=\bar{x}_{ij}(t)+\bar{v}_{ij}(t+\Delta t)\Delta t$$

Начальные параметры.

Нами были подобраны параметры системы, которые делают движение поверхности похожим на движение поверхности воды.

Шаг сетки, **H**	Масса узла, $\mathbf{m_{ij}}$	Коэффициент упругости, $\mathbf{c_к}$	Демпфирование, γ_{ij}	Шаг интегрирования, Δt
1.0	**4.0**	**7.0**	**0.05**	**0.4**

Табл. 1

Из таблицы видно, что массы узлов и коэффициенты упругости пружин выбираются одинаковыми. В принципе для получения базовых эффектов этого достаточно, и структуры данных можно упростить, не выделяя памяти под эти параметры для каждого узла.

Граничные условия.

По соглашению, в нашей модели поверхность в состоянии покоя располагается в некоторой горизонтальной плоскости, поэтому далее, чтобы упростить выкладки, мы будем говорить о горизонтальном или вертикальном направлении именно в таком смысле.

Чтобы во время движения поверхность не смещалась от места, определенного для водоема, и для получения эффекта отражения волн от границ, мы вводим два типа условий на границе:

- узлы полностью закреплены, и смещение этих узлов на каждом шаге полагается равным нулю;
- допускается только вертикальное движение узлов, т.е. горизонтальное смещение полагается равным нулю.

Первый тип удобен для моделирования береговой линии, а второй дает очень хороший результат в случае моделирования бассейнов и каких-либо других емкостей с водой, создается эффект движения воды вдоль стенок.

Управление движением, возмущения.

Наиболее интересным и простым подходом к управлению движением поверхности является гибридный подход, когда контроль осуществляется не только посредством сил, но и посредством прямой модификации скоростей/позиций. В этом случае перед очередным шагом интегрирования вместо сил, с

которыми происходит воздействие, мы можем сразу указать результат воздействия, что существенно облегчает задачу получения необходимого движения.

Для описания точечных возмущений, например, попадания небольшого предмета в воду, можно воспользоваться тремя способами. Во-первых, задать внешнюю силу, но этот путь мы не используем. Во-вторых, можно изменить скорость в узле и некоторой его окрестности. В-третьих, аналогично можно поступить с позициями. Мы пользуемся либо вторым, либо третьим путем, в зависимости от того, какой визуальный результат нам требуется. Мы просто задаем возмущение, имеющее форму гауссовского распределения с центром в точке возмущения и с некоторым заранее подобранным радиусом. Значения функции рассчитываются с шагом сетки **h** заранее и хранятся в таблице. Как правило, используются аддитивные возмущения, хотя мы реализовали и другие способы.

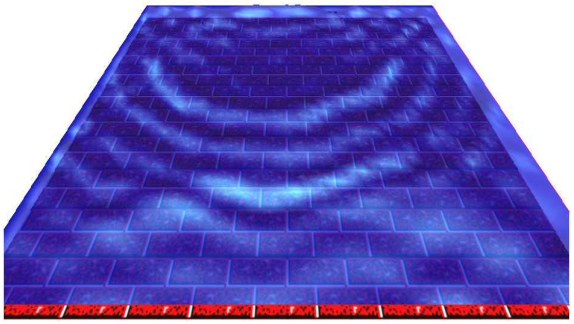

Рис. 4

На Рис.4 изображена поверхность с закрепленной полностью границей и заданным некоторое время назад возмущением скоростей.

При задании большого по модулю возмущения можно получить эффект неоднозначности профиля волны с большой амплитудой, что по виду будет очень похоже на подводный взрыв (Рис.5).

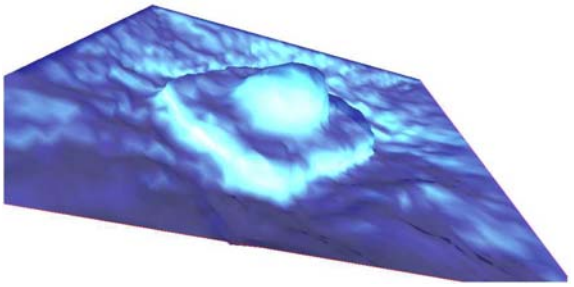

Рис. 5

Для получения требуемого движения также можно воспользоваться наложением ограничений (типа а) или б)) не только на граничные, но и на любые другие узлы. Наложив ограничения типа а) на некоторые узлы поверхности, можно смоделировать, например, островок. Интересный результат дает например наложение ограничений типа б) (запрет горизонтальных перемещений) на все узлы нашей поверхности. В этом случае эффект прохождения волн друг сквозь друга сильно подчеркивается (Рис.6).

Эффект от попадания или нахождения в воде большого предмета легко может быть достигнут соответственным определением ограничений или возмущений на границе пересечения объекта и поверхности.

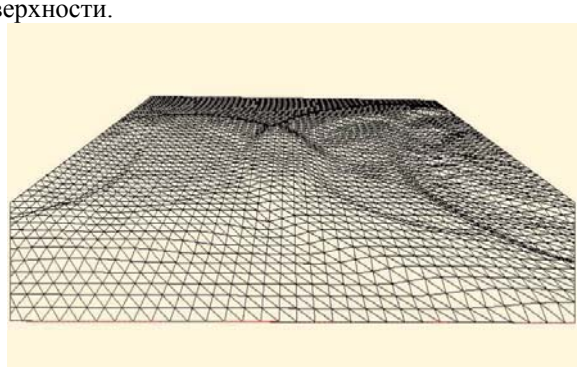

Рис. 6

Оценка нормалей к поверхности.

Для отображения поверхности на экране мы используем упрощенную модель диффузного освещения. Она требует оценки нормалей к поверхности в вершинах треугольников. Хорошо показал себя метод вычисления нормалей по четырем смежным вершинам, как показано на Рис.7.

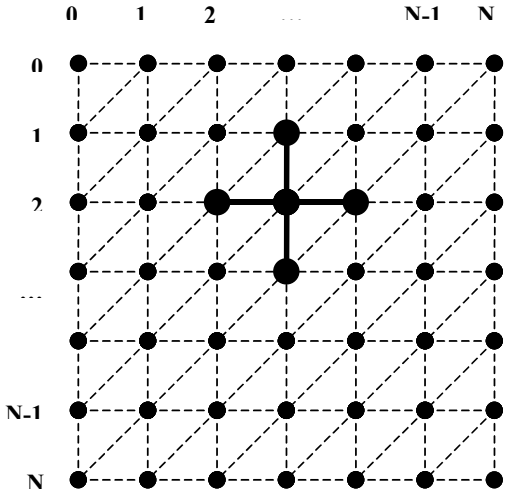

Рис. 7

Оптимальная реализация алгоритма.

Основная вычислительная нагрузка в алгоритме расчета движения падает на вычисление сил пружин. Вернемся к формуле (1) и перепишем ее в другом виде.

$$\bar{f}_i^k(t) = c_k\left(\bar{x}_j - \bar{x}_i\right)\left(1 - l_k^r \frac{1}{l_k}\right) \qquad (1)'$$

Во-первых, бросается в глаза факт, что сила той же пружины $\mathbf{S_k}$ для узла $\mathbf{p_j}$ есть взятая с противоположным знаком сила $\bar{f}_i^k(t)$ для узла $\mathbf{p_i}$. Поэтому на каждом шаге сила для пружины вычисляется один раз и складывается с соответствующим знаком с результирующей силой соответствующих узлов.

Наиболее уязвимым с точки зрения производительности является вычисление сомножителя $\frac{1}{l_k}$, который на самом деле рассчитывается по формуле $\frac{1}{\sqrt{l_k{}^2}}$, где $l_k{}^2$ получается как сумма квадратов компонент соответствующего вектора. Для расчетов мы пользуемся приближенной формулой Ньютона-Гольдшмита для функции $\frac{1}{\sqrt{x}}$, реализация которой получилась у нас примерно в два раза быстрее, чем композиция машинных корня квадратного и деления. Мы не будем останавливаться на деталях приближенного вычисления такой функции, так как различные его варианты часто встречаются в литературе.

Следует особо упомянуть о низкоуровневых архитектурно-ориентированных аспектах оптимизации. Алгоритм реализовывался специально для архитектуры компании Intel (IA-32) и для выполнения на процессоре Pentium Pro и старше. Во-первых, чтобы минимизировать обращения к кэш-памяти для данных, мы разработали однопроходный алгоритм, который для каждого узла вычисляет сразу все величины для очередного шага, от результирующей силы до нормали и цвета в вершине. Во-вторых, код реализован так, чтобы уменьшить взаимозависимости между вычислительными операциями и воспользоваться тем самым всеми преимуществами динамического исполнения команд.

В результате при базовой сетке из 50x50 узлов время вычисления одного шага составляет в среднем 4.8 миллисекунды на процессоре Intel Pentium II 300Mhz. Следует заметить также, что часто бывает достаточно существенно меньшего количества узлов.

Алгоритм расчета движения поверхности допускает явный параллелизм, что дает возможность получить очень эффективную реализацию на многопроцессорной архитектуре, либо, используя векторную реализацию, существенно повысить производительность за счет использования SIMD-инструкций (напр., SSE Pentium III).

Повышение эффективности отображения.

При отображении поверхности на экране с помощью трехмерного графического API (OpenGL, DirectX) на отображение передается большое количество треугольников, что на самом деле излишне. Детально и с большим количеством граней необходимо отображать только ближний план, а на дальнем плане поверхность можно огрубить, уменьшая количество граней.

Мы используем адаптивный к положению камеры алгоритм формирования триангуляции поверхности по набору точек. За основу был взят алгоритм отображения полей высот, разработанный Finch&Bishop и довольно хорошо изложенный в [2].

Два момента следует отметить особо.

Во-первых, изначально алгоритм ориентирован на поля высот, которые допускают параметризацию вида z = z (x, y), где точки (x, y) принадлежат некоторой прямоугольной области. В нашем же случае область изменения параметров не является проекцией поверхности на координатную плоскость. Поэтому пришлось доработать алгоритм, чтобы он работал с параметризацией вида $\bar{x} = \bar{x}(u,v), (u,v) \in [0,N] \times [0,N]$ и подобрать другую функцию для вычисления ошибки.

Во-вторых, в работе не было дано точного рецепта обработки границ поверхности и границ между уровнями детализации, и эту проблему пришлось решать нам самим.

На Рис.8а изображена поверхность без применения алгоритма упрощения (4802 треугольника), а на Рис.8б – с использованием алгоритма упрощения (2431 треугольник).

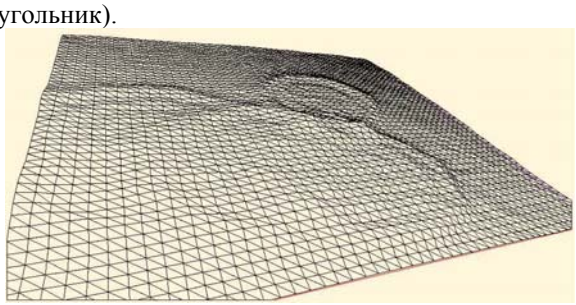

Рис. 8а

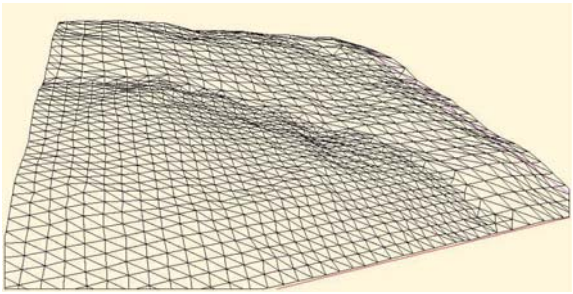

Рис. 8б

Рисунки 9а и 9б изображают соответственно неупрощенную (4802 треугольника) и упрощенную (2431 треугольник) поверхности с освещением и в цвете.

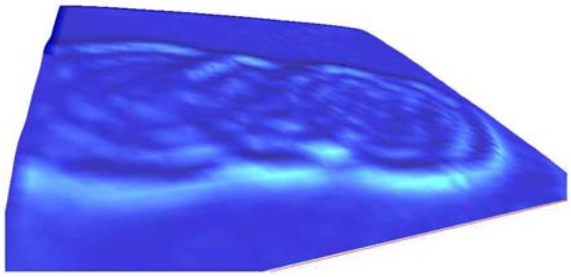

Рис. 9а

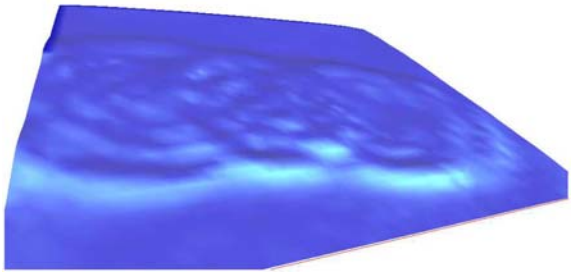

Рис. 9б

3. ЗАКЛЮЧЕНИЕ И ДАЛЬНЕЙШИЕ ИССЛЕДОВАНИЯ.

Итак, мы получили реализацию физической модели, позволяющей эффективно имитировать движение поверхности воды. На модели достигнута производительность, позволяющая использовать ее в приложениях реального времени. Кроме того, мы дополнили модель адаптивным механизмом, позволяющим разгрузить геометрический конвейер трехмерного графического API.

Будущие исследования будут вестись в двух направлениях.

Во-первых, дальнейшее увеличение производительности как расчетов за счет применения векторных подходов и использования более быстрых методов приближения, так и отображения за счет уменьшения выходного количества треугольников. Уменьшение количества треугольников возможно при подборе соответствующих параметров нашего алгоритма упрощения, мы просто еще не провели эту работу, хотя здесь есть большой потенциал.

Во-вторых, увеличение реалистичности и детальности изображения поверхности воды. Это направление открывает широкий простор для творчества. В качестве примеров приведу три идеи, реализация которых авторами уже обдумана и будет воплощена в жизнь в ближайшее время. Для увеличения реалистичности необходимо добавить механизм моделирования брызг. Также очень хорошо будет, по мнению авторов, смотреться преломление изображения дна. А для увеличения детализации требуется использовать механизм текстурирования, предложенный в [4]. В заключение хотелось бы упомянуть об идее использования модели для анимации больших участков водной поверхности, когда на ближних планах можно применять нашу трехмерную динамическую модель, а на дальних – один из двумерных псевдодинамических подходов.

Литература.

[1] Yuencheng Lee, Demetri Terzopoulos and Keith Waters, *Realistic Modeling for Facial Animation.* Computer Graphics Proceedings, Annual Conference Series, 1995, ACM SIGGRAPH, pp.55-62

[2] Mark Finch, Lars Bishop. *Sorted Height Field Rendering.* White paper of NDL company. http://www.ndl.com/ndlter.pdf

[3] Alan H. Watt, Mark Watt, *Advanced Animation and Rendering Techniques : Theory and Practice,* 1992, Addison-Wesley Pub Co, pp.421-426, Animating water waves

[4] Jos Stam, *Aperiodic Texture Mapping*, 1997,

Об авторах

Маркичев Александр – (e-mail: mark@nstl.nnov.ru),
Супиков Алексей – (e-mail: sam@nstl.nnov.ru).

Abstract

Physically-based simulation of deformable surfaces in gaming and animation applications. Water surface animation.

A. Markhichev, A. Soupikov

Paper describes physically-based simulation of deformable surfaces in application to the real-time water surface simulation. We present our research results of using the uniaxial finite-element model (namely, spring-mass model) in animation the basic features of simulated water surface. The algorithm we developed allows realistic looking animation with various visual effects (splashes, boundary reflection, waves propagation caused by large objects and others) on one hand and performance acceptable for real-time VR and gaming applications on the other hand. Since the formulation we use is physically based it keeps all advantages of the physically based modeling approach.

To reduce rendering pipeline load we use view-dependent adaptive mesh generation algorithm.