Методы визуализации и анимации моделей протяженных водных поверхностей в системах виртуальной реальности

Н. А. Елыков, И. В. Белаго, С.А. Кузиковский
Институт Автоматики и Электрометрии СО РАН
Новосибирск, Россия

Аннотация

Предложена техника для генерации и анимации протяженных водных поверхностей с использованием *GPU* (*Graphics Processing Unit*). Детально описаны методы симуляции поведения водной поверхности. Описывается метод непрерывной детализации геометрической модели водной поверхности. Так же представлена техника генерации и анимации текстур водной поверхности.

Ключевые слова: системы виртуальной реальности, симуляция водных поверхностей, преломление волн.

1. ВВЕДЕНИЕ.

Бурный рост производительности и качества графических средств персональных компьютеров в последние годы и глобальная стандартизация графических программных интерфейсов существенно расширила применение PC в качестве аппаратной платформы для различных графических приложений, таких как приложения виртуальной реальности и симуляторы.

Моделирование визуальной обстановки в настоящее время является одной из наиболее динамичных и бурно развивающихся областей в компьютерной индустрии. Программные модели виртуального мира приобретают все большую схожесть с миром реальным. Благодаря присущей ему сложности, реалистичное моделирование природных феноменов является одной из наиболее вычислительно емких областей компьютерной графики. Один из таких феноменов – это морские сцены. Данная статья посвящена исследованию методов визуальной имитации морских поверхностей в реальном времени.

Существующие методы визуализации протяженных моделей водных поверхностей производят большую часть вычислений на *CPU*, например, решение уравнений *Navier-Stokes* [1] или применение преобразования Фурье [2,3]. Авторами в [4] был предложен метод визуализации водных поверхностей в реальном времени учитывающий эволюцию морской поверхности во времени в зависимости от направления и скорости ветра, формы морского дна и береговой линии. К сожалению, данный метод производил большинство вычислений, требующихся для генерации и анимации водной поверхности на *CPU*. В данной статье рассматривается адоптация предложенного метода для переноса большинства расчетов на *GPU*, для того, что бы оставить *CPU* для других вычислений необходимых в системах виртуальной реальности.

Описываемая методика генерации изображения морской поверхности состоит из трех частей. Первая часть представляет собой алгоритм, позволяющий получить форму поверхности моря в зависимости от времени и других факторов. Вторая часть позволяет построить оптимальную, с точки зрения качества/производительность, триангуляцию. Третья часть посвящена методам получения текстур для морской поверхности.

2. АНИМАЦИЯ ВЕРШИН ВОДНОЙ ПОВЕРХНОСТИ.

Поверхность моря можно представить в виде некоторой функции $H(x,y,t)=z$, заданной на двухмерном наборе данных. Наиболее простой метод представление данной функции на GPU - это суперпозиция нескольких синусоидальных волн с собственной амплитудой, периодом, начальной фазой и направлением распространения [5]:

$$H(x,y,t)=\sum A_i \times \sin(D_i \cdot (x,y) \times \omega_i + t \times \varphi_i)$$

при этом сама точка на поверхности воды - $P(x,y,t)=(x,y,H(x,y,t))$, а касательные и нормаль к поверхности в данной точке воды задаются следующим образом:

$$B(x,y,t)=\left(\frac{\partial x}{\partial x},\frac{\partial y}{\partial x},\frac{\partial}{\partial x}H(x,y,t)\right)=\left(1,0,\frac{\partial H}{\partial x}\right)$$

$$T(x,y,y)=\left(\frac{\partial x}{\partial y},\frac{\partial y}{\partial y},\frac{\partial}{\partial y}H(x,y,t)\right)=\left(0,1,\frac{\partial H}{\partial y}\right)$$

$$N(x,y,t)=B(x,y,t)\times T(x,y,t)=\left(-\frac{\partial H}{\partial x},-\frac{\partial H}{\partial y},1\right)$$

Текущее время, фаза, амплитуда и направление распространения волны при этом передаются в вершинный шейдер в виде констант. К сожалению, данное представление задает достаточно “плавные” волны (см. Рис. 1), что соответствует достаточно спокойной, далёкой от берега водной поверхности. Для более реалистичного моделирования поведения воды необходимо контролировать “остроту” волн.

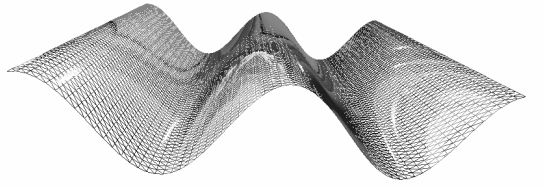

Рис. 1. Суперпозиция синусов

В [4] для управления “остротой” волн было преложено использовать линейную интерполяцию двух функций $w(u)=\cos(2\pi u)$ и $w(u)=8(u-0.5)^2-1$, где $u\in[0..1)$ и $w(0)=w(1)$. Вместо этого для уменьшения количества вычислений и достижения большего реализма, была использована формула *Gerstner* -модификация суперпозиции синусов, которая как бы “сдвигает” вершины волны (см. Рис 2.):

$$P=\begin{pmatrix} x+\sum Q_i A_i D_{ix}\cos(\omega_i D_i\cdot(x,y)+\varphi_i t)\\ y+\sum Q_i A_i D_{iy}\cos(\omega_i D_i\cdot(x,y)+\varphi_i t)\\ \sum A_i\sin(\omega_i D_i\cdot(x,y)+\varphi_i t)\end{pmatrix},$$ где Q_i это параметр контролирующий “остроту” волн. При $Q_i=0$, волна становится обычной “плавной” волной, а при $Q_i=\frac{1}{\omega_i A_i}$, волна становится максимально острой, при $Q_i>\frac{1}{\omega_i A_i}$, волна выродится, и вместо гребней волн появятся циклы. Таким образом, мы можем задавать остроту волны в виде параметра $Q\in[0..1]$, при этом $Q_i=\frac{Q}{\omega_i A_i n}$, где *n*-количество волн.

Касательные и нормаль в некоторой точке поверхности при этом принимаю следующий вид:

$$B=\begin{pmatrix} 1-\sum Q_i D_{ix}^2 A_i\omega_i\sin(\omega_i D_i\cdot(x,y)+\varphi_i t)\\ -\sum Q_i A_i\omega_i D_{ix}D_{iy}\sin(\omega_i D_i\cdot(x,y)+\varphi_i t)\\ \sum A_i\omega_i D_{ix}\cos(\omega_i D_i\cdot(x,y)+\varphi_i t)\end{pmatrix}$$

$$T=\begin{pmatrix} -\sum Q_i A_i\omega_i D_{ix}D_{iy}\sin(\omega_i D_i\cdot(x,y)+\varphi_i t)\\ 1-\sum Q_i D_{iy}^2 A_i\omega_i\sin(\omega_i D_i\cdot(x,y)+\varphi_i t)\\ \sum A_i\omega_i D_{ix}\cos(\omega_i D_i\cdot(x,y)+\varphi_i t)\end{pmatrix}$$

$$N=\begin{pmatrix} -\sum \omega_i A_i D_{ix}\cos(\omega_i D_i\cdot(x,y)+\varphi_i t)\\ -\sum \omega_i A_i D_{ix}\cos(\omega_i D_i\cdot(x,y)+\varphi_i t)\\ 1-\sum Q_i\omega_i A_i\sin(\omega_i D_i\cdot(x,y)+\varphi_i t)\end{pmatrix}$$

Рис. 2. Волны *Gerstner*.

Текущее время, фаза, амплитуда, острота и направление распространения волны при этом передаются в вершинный шейдер в виде констант. К сожалению, данная формула не учитывает форму дна и берега. Введем зависимость скорости распространения волны в зависимости от глубины воды, как функцию фазы от координат $\varphi_i(x,y)$. Пространственная компонента фазы позволяет ввести зависимость от поверхности дна, что делает возможным моделирование таких эффектов как рефракция волн.

Хорошим приближением для случая, когда период волны почти не зависит от глубины, является трансцендентное выражение[6]: $k\tanh(kh(x,y))=k_\infty$,где $h(x,y)$ – глубина в некоторой точке, k – волновое число, а k_∞ - волновое число на бесконечной глубине. Это уравнение легко решается для предельных значений h: для малой глубины ($h<0.05L$, где L - период волны) имеем $k=\sqrt{\frac{k_\infty}{h}}$ для глубокого моря ($h>0.25L$) имеем $k=\frac{k_\infty}{\sqrt{\tanh(kh)}}$.

Соответственно находим пространственную компоненту фазы:

$$\varphi_i(\vec{x}_i)=\int_0^{\vec{x}_i}k_i(u)du=\sum_0^{\vec{x}_i}\frac{k_\infty}{\sqrt{\tanh(k_\infty h(x_i))}}\Delta x,$$ где осуществлен переход к численному интегрированию. Нужно отметить, что выполнять численное интегрирование для данной формы дна, длинны волны и направления распространения только один раз на этапе подготовки, а не отдельно на каждом кадре. На рисунке 3 представлен пляж с набегающими на него волнами, причем береговая линия представлена ломанной, а глубина меняется линейно с увеличением расстояния от берега.

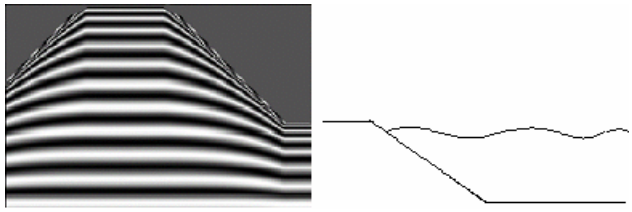

Рис. 3. Берег с линейно изменяющейся глубиной и набегающие на берег волны (слева вид с верху, справа срез). Хорошо видна рефракция волн.

Для достижения большего реализма авторами также использовалась зависимость коэффициента “остроты” и амплитуды волны от глубины моря и длинны волны - $A_i(x,y)$, $Q(x,y)$, так что бы $Q(x,y)$ было близко к 0 для “глубокой воды”, и к 1 для прибрежных волн.

Таким образом, имея функцию глубины, можно на этапе подготовки для каждой вершины моря предвычислять и сохранять пространственную компоненту фазы $\varphi_i(x,y)$ и коэффициент “остроты” волны $Q(x,y)$. Далее сохранив данные параметры в качестве атрибутов вершины используя вершинный шейдер можно вычислять позицию вершины

$P(x,y,t)$, нормаль $N(x,y,t)$ и касательные к поверхности $B(x,y,t)$ $T(x,y,t)$ на *GPU*. Таким образом, полностью освободить *CPU* от вычислений связанных с генерацией и анимацией вершин водной поверхности.

На видео ускорителе класса *GeForce3* возможно аппаратно вычислять суперпозицию до четырех волн. Нормали и касательные к поверхности необходимы для вычисления отражения и преломления в пиксельном шейдере. Для упрощения вычислении в вершинном шейдере считается что изменение пространственной компоненты фазы, параметра "остроты" и амплитуды волны малы по сравнению с длинной волны и $\frac{\partial \varphi_i(x,y)}{\partial x}, \frac{\partial \varphi_i(x,y)}{\partial y}, \frac{\partial Q_i(x,y)}{\partial x}, \frac{\partial Q_i(x,y)}{\partial y}, \frac{\partial A_i(x,y)}{\partial x}, \frac{\partial A_i(x,y)}{\partial x}$ малы и не вносят существенный вклад в касательные и нормаль к поверхности моря.

3. ОПТИМАЛЬНАЯ ТРИАНГУЛЯЦИЯ МОРСКОЙ ПОВЕРХНОСТИ В ЗАВИСИМОСТИ ОТ ПОЛОЖЕНИЯ НАБЛЮДАТЕЛЯ.

Необходимость интерактивно генерировать и визуализировать большие площади морской поверхности, состоящие из многих миллионов треугольников, вынуждает для выполнения условий реального времени использовать алгоритм управления детализацией, который модифицирует геометрическую модель для уменьшения числа анимируемых и визуализируемых треугольников (вершин).

На алгоритм управления детализацией можно наложить несколько формальных требований [7]:

i. Триангуляция должна удовлетворять требованию минимальной избыточности, т.е. важные с точки зрения текущего кадра части морской поверхности должны быть представлены большим количеством треугольников.

ii. Динамические изменения сетки, влекущие за собой изменение параметров поверхности или ее геометрии, не должны влиять на производительность.

iii. Высокочастотные особенности поверхности, такие как локальные выпуклости или вогнутости, не должны вести к глобальному усложнению модели.

iv. Небольшие изменения положения наблюдателя не должны вести к значительному изменению геометрии модели, во избежание значительных изменений получаемого изображения кадр от кадра и для сохранения скорости генерации одного кадра почти постоянной.

v. Алгоритм должен иметь возможность заранее задавать сложность получаемой модели (т.е. иметь возможность управлять изменениями качества).

В настоящее время наличие аппаратного ускорения компьютерной графики стало стандартом в индустрии. Поэтому необходимы алгоритмы, которые наиболее подходят для аппаратного ускорения. Современные 3D ускорители способны обрабатывать и отображать большое количество примитивов, таким образом, алгоритмы основанные на упрощении треугольников проигрывают алгоритмам основанных на упрощении сущностей более высокого уровня.

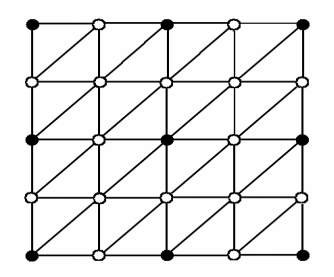

Рис 4. Пример построения первого уровня детализации для блока размером 5x5. Черным выделены вершины, попавшие в первый уровень детализации.

В качестве алгоритма для управления детализацией геометрической модели морской поверхности выбран метод регулярного прореживания. В данном методе водная поверхность представляется в виде равномерной решетки, в узлах которой находятся вершины представляющие геометрическую модель. Такое представление модели просто в реализации и требует минимальных затрат памяти. Расстояние между узлами решетки постоянно и зависит от характерной длинны волны, а размер решетки *N*x*N*, где *N* [1,∞], таким образом, получается (*N* – *1*)*(*N* - *1*) четырехугольников, каждый из которых в свою очередь состоит из двух треугольников, которые и задают геометрическую модель водной поверхности. Далее решетка разбивается на блоки размером *M*x*M* (например, 256 x 256). В каждом из блоков можно хранить охватывающую сферу для отбраковки блоков, не попадающих в пирамиду видимости.

Блоки находящиеся вдали от наблюдателя необходимо отображать с более низким уровнем детализации, чем блоки находящиеся вблизи. Для получения блока с более низкой детализацией исключим из первоначального блока (уровень детализации 0) каждый второй столбец и строку, в результате будем иметь блок с более низкой детализацией (уровень детализации 1). Каждый следующий уровень детализации получается из предыдущего подобным образом. Так можно создать полную последовательность уровней детализации для каждого из блоков задающих геометрическую модель водой поверхности. На рисунке 4 представлен первый уровень детализации блока 5x5 вершин, заметим, что так же может быть получен и второй уровень детализации.

При отображении водной поверхности на основе информации о видимости данного участка водной поверхности, принимается решение об отображении блока и на основе ошибки в экранном пространстве и положении наблюдателя выбирается тот или иной уровень детализации для данного блока. Заметим, что из-за особенностей поведения водной поверхности, в качестве ошибки можно выбрать площадь сферы с радиусом суммарной амплитуды волн. Когда два соседних блока имеют различный уровень

детализации, то на границах блока могут появляться различного рода нестыковки геометрии. Для решения данной проблемы можно вставлять дополнительные вершины вдоль границы таких блоков, но это ведет к тому, что придется динамически изменять вершины и их соединения для данного уровня детализации и делать это каждый раз, когда соседний блок будет менять свою детализацию, что приведет к падению производительности. В [8] предложен оригинальный алгоритм для решения данной проблемы.

На рисунке 5 показан стык двух блоков с разной детализацией (граница блоков указана стрелочкой), вершины, которые выделены серым цветом, отсутствуют в левом блоке, и поэтому в этом месте возникает нестыковка. Что бы этого избежать, соединение вершин в блоке с более высоким разрешением изменяются, так что бы исключить "проблемные" вершины (вершины в которых возникает нестыковка) из геометрической модели, тем не менее, оставляя их в самом блоке. Наиболее быстрый способ сделать это рисовать пограничные вершины, объединяя их в *triangle fan*, а остальные, рисуя обычным путем. Сами же блоки для достижения наибольшей производительность рисуются как *triangle strips*.

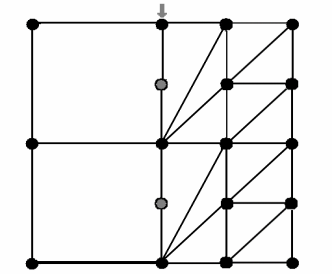

Рис 5. Пример стыковки блоков с разным уровнем детализации. Стрелочкой показана граница блока. Слева находится блок с более низкой детализацией. Серым выделены вершины, которые нарушают стыковку блоков.

Алгоритм обладает следующими достоинствами:

- Алгоритм хорошо удовлетворяет требованиям, предъявляемым современными видео ускорителями.
- Простота реализации и высокая скорость работы.
- Алгоритм отображает различные районы морской поверхности с различным уровнем детализации.
- Алгоритм поддерживает устойчивую скорость генерации одного кадра.
- Алгоритм автоматически поддерживает искусственную ликвидацию разрывов между блоками с различной детализацией.
- На основе охватывающих параллелепипедов, алгоритм отбраковывает невидимые блоки.
- Алгоритм поддерживает генерацию больших последовательностей *triangle strip*, а так же позволяет проводить оптимизации для *Post T'n'L* кэш.
- Алгоритм поддерживает динамическое управление качеством получаемой геометрической модели.

Алгоритм обладает одним недостатком - при переключении детализации блоков, происходит сильное изменение в геометрии по всем блоку, а так же изменяется освещение на стыках между блоками. Для уменьшения артефактов ("*popping*") при переключении детальности блока, выполняется по вершинный блендинг от одного уровня детализации к другому [9]

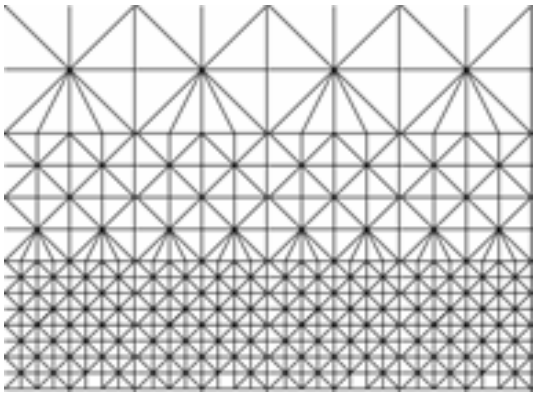

Рис. 6. Пример работы алгоритма регулярного прореживания

Таким образом, благодаря простоте реализации, высокой скорости работы алгоритма и то, что алгоритм хорошо оптимизируется для 3D акселераторов последнего поколения, этот алгоритм возможно использовать для визуализации моделей протяженной водной поверхности. На рисунке 6 представлен пример работы алгоритма.

4. ТЕХНИКА ГЕНЕРАЦИИ И АНИМАЦИИ ТЕКСТУР.

Волны на воде обладают "фрактальными" свойствами, т.е. при различном увеличении они выглядят почти одинаково (подобно). Исходя из свойства "само подобия" часть высокочастотных особенностей воды можно выполнить в виде текстурных эффектов. Для этого используется несколько текстурных карт нормалей представляющих собой волны с разной частотой, амплитудой, скоростью и направлением распространения, которые суммируются друг с другом. Эти карты нормалей могут быть нарисованы дизайнером или могут быть построены исходя из формул для анимации вершин водной поверхности. Для того, что бы уменьшить результирующую заливку, суммирование выполняется в специальную текстуру накопитель один раз на каждом кадре. Количество проходов необходимых для этой операции зависит от возможностей конкретного видео ускорителя.

Полученная результирующая карта нормалей $\vec{N} = \sum \vec{N}_i$ используется для построения вектора отражения в каждой точке (пикселе) поверхности по следующей формуле: $\vec{R} = 2\vec{N} \cdot (\vec{V} \cdot \vec{N}) - \vec{V}$, где $\vec{V}$ - нормированный вектор направления на наблюдателя для

данной точки. Причем т.к. $\vec{N}$ задан в локальной системе координат поверхности, то и $\vec{V}$ должен быть задан в той же системе координат. Для перевода из одной системы координат в другую служат касательные $B(x,y,t)$, $T(x,y,t)$ и нормаль $N(x,y,t)$ к поверхности воды, вычисленные в каждой вершине водной поверхности с помощью вершинного шейдера. Полученный вектор используется для сэмплирования текстуры отражения. Текстуры отражения и преломления могут быть как предпросчитанными [4], или строится на каждом кадре [10].

Для смешивания получившегося отражения и преломления используется приближение формулы Френеля:

$R(\alpha) = R(0) + (1 - R(0)) \cdot (1 - \cos(\alpha))^5$, где $R(0) = \frac{(n_1 - n_2)^2}{(n_1 + n_2)^2}$, n_1 и n_2 коэффициенты преломления соответствующих сред, а $\cos(\alpha) = \vec{V} \cdot \vec{N}$. Для воздуха и воды $R(0) = 0.02037$. [10]

Для достижения большего реализма авторами использовалась зависимость прозрачности и цвета воды в зависимости от глубины, а так же для анимации текстурных координат карты нормалей, представляющей поведение поверхности воды применялась шумовая функция Перлина [11] (*Perlin Noise function*)

Т.к. производится суммирование нескольких карт нормалей в отдельную текстуру то данный метод без учета преломления может быть выполнен на видео ускорителях класса *GeForce 3*.

5. ЗАКЛЮЧЕНИЕ.

Авторами данной работы были разработаны методы визуализации и анимации моделей протяженных водных поверхностей в системах виртуальной реальности, включающие в себя генерацию и анимацию вершин и текстур водной поверхности. Предложенные алгоритмы учитывают особенности архитектуры современных видео ускорителей. Они позволяют проводить большую часть вычислений на GPU, оставляя при этом CPU практически свободным для других вычислений необходимых в системах виртуальной реальности, таких как обсчет физики, AI и пр. Так же было предложено использовать метод регулярного прореживания для динамического упрощения геометрической модели водной поверхности с целью достичь максимально возможной производительности на конкретной аппаратной конфигурации. Предложенные алгоритмы позволяют пользователю регулировать различные параметры, влияющие на визуальное качество получаемых изображений и на скорость генерации одного кадра (количество треугольников в сцене, степень детализации, время работы подсистемы упрощения геометрической модели и пр.), а также позволяющий регулировать такие “физические” параметры как скорость и направление ветра, рельеф морского дна и линию берега, ширина полосы прибоя и пр.

Реализация алгоритмов показала, что визуализация участка океанской поверхности 5000x5000 м^2 га машине Pentium IV – 1 GHz; RAM - 256Mb; с видео ускорителем NVidia GeForce 4 происходило со скоростью ≈55 кадров в секунду. Такой скорости в большинстве случаев оказывается достаточно для эффективного применения библиотеки в системах синтеза визуальной обстановки.

6. ЛИТЕРАТУРА:

[1] Foster, N. and Metaxas, D. “*Realistic Animation of Liquids. Proceedings*” GI '96, pp. 204-212

[2] Gray A. Mastyn, Peter Watterberg and John Mareda *"Fourier Synthesis of Ocean Scenes."* IEEE Computer Graphics and Applications Mar.1987 pp.16-23

[3] Tessendorf, J. “*Simulating Ocean Water.*” SIGGRAPH 2001 Course notes. http://home1.gte.net/tssndrf/index.html.

[4] Н.А. Елыков, И.В. Белаго, М.М. Лаврентьев Ю.Ю. Некрасов “*Генерация изображения водной поверхности в реальном времени*” // Труды конференции GraphiCon’2000, с. 268-275, 2000

[5] Mark Finch “Effective Water Simulation from Physical Models” GPU Gems 2004

[6] D.R. Peachey *"Modeling Waves and Surfaces."* Computer Graphics (Proc. SIGGRAPH'86) Aug. 1986 pp. 65-74

[7] P. Lindstrom, D. Koller, W. Ribarsky, L. F. Hodges, N. Faust and G. A. Turner, *Real-time, continuous level of detail rendering of height fields*, In Proc. SIGGRAPH ’96, pages 109-118,Aug. 1996.

[8] Willem H. de Boer,*Fast Terrain Rendering Using Geometrical MipMapping*,E-mersion Project, October 2000, http://www.connectii.net/emersion

[9] Daniel Wagner “*Terrain Geomorphing in the Vertex Shader*” ShaderX2 2004

[10] Kurt Pelzer “*Advanced Water Effects*” ShaderX2 2004

[11] "*Using MMX™ Instructions for Procedural Texture Mapping. (Based on Perlin's Noise Function).*" Intel Developer Relations Group © 1996

Об авторах:

Елыков Николай Александрович– м.н.с. ИАиЭ СО РАН

E-mail: **Nicolas@sl.iae.nsk.su**

Белаго Игорь Викторович– н.с. ИАиЭ СО РАН

Кузиковский Станислав Александрович– н.с. ИАиЭ СО РАН

High realistic visualization and animation of open water surfaces in virtual reality systems

Abstract

A technique for rendering and simulation of large water surfaces on GPU is proposed. Methods of ocean surface behavior simulation are described in details. Geometry detail selection algorithm, based on the camera position, is described. A technique to generate and animate the texture used for the ocean surface is also provided.

Keywords: *real-time rendering, water surface simulation, ocean wave refraction.*

Elykov Nikolay – Institute of Automation and Electometry SB RAS. E-mail: **Nicolas@sl.iae.nsk.su**

Belago Igor– Institute of Automation and Electometry SB RAS.

Kuzikowski Stanislaw – Institute of Automation and Electometry SB RAS.