Реализация образовательных инструментов в виртуальных 3D-средах с использованием потоковых процессоров

Андрей Юрьевич Сморкалов
аспирант,
Марийский государственный технический университет,
пл.Ленина, 3, г.Йошкар-Ола, 424000, +7-836-2-686090
asmorkalov@mail.ru

АННОТАЦИЯ

Статья рассматривает реализацию образовательных инструментов в виртуальных мирах с использованием вычислительных возможностей графических потоковых процессоров. Большинство инструментов в образовательной 3D-среде так или иначе требуют реализации вычислительно сложных операций обработки изображений. Использование потоковых процессоров позволяет освободить для выполнения других необходимых для функционирования образовательного виртуального пространства расчетов центральный процессор, а также реализовать сами инструменты на более высоком уровне в плане графического отображения, с расширенным числом возможностей и поддерживать большее количество работающих одновременно с инструментами учеников и преподавателей.

Paper is devoted to development of educational tools in virtual worlds with using computing power of graphics stream processors. Realization of complex image processing operations is required for a lot of tools in educational 3D environment. Stream processors using give ability to free CPU for performing other tasks, which needed for work of virtual educational space. Also it makes possible to develop educational tools with advanced graphics and additional features and to support more simultaneous teachers and students working with tools.

Ключевые слова

обработка изображений, образовательные инструменты, графические потоковые процессоры, виртуальные образовательные среды;
image processing, educational tools, graphics stream processors, virtual educational environments.

Введение

На текущий момент обучение в виртуальных образовательных средах перестало быть педагогическим экспериментом и перешло в разряд постоянной деятельности, которую ведут десятки университетов из самых разных уголков мира. Так, в 2010 году New Media Consortium, некоммерческий консорциум 250-ти учебных заведений, музеев и исследовательских центров со всего мира, сделал заявление, что образовательные видеоигры и виртуальные мировые классные комнаты станут основными учебными методами и пособиями уже в ближайшие пять

лет. Крупные компании, такие как IBM и Microsoft, уже активно занимаются разработкой виртуальных 3D-сред для образовательных целей.

Существует несколько видов образовательных виртуальных миров, включая виртуальные музеи и художественные галереи, виртуальные театры, виртуальные среды для изучения определенных учебных тем [1], виртуальные библиотеки и виртуальные языковые среды, а также виртуальные кампусы [2]. Хотя музеи и галереи можно посетить через обычные мультимедийные программы, в виртуальном мире-музее возможно совместное посещение и обсуждение в реальном времени.

Возможности полноценных виртуальных миров делают их уникальным и мощным средством для образования в целом [3]. Помимо этого, технологии виртуальной реальности подходят и для дистанционного образования, делая его действительно эффективным [4]. Основными достоинствами дистанционного обучения с помощью виртуальных миров является значительно меньшие требования к скорости интернет-соединения, чем при передаче видео, и возможность совместной деятельности, которую лишь в существенно ограниченной форме предоставляют вебинары.

На сегодня существует множество различных образовательных виртуальных сред, все из которых обладают следующими свойствами:

1. совместное пространство, в котором одновременно находятся и действуют множество пользователей.
2. графический пользовательский интерфейс: виртуальный мир отображается в 3D или 2D-форме с различным стилем передачи изображения: от мультипликационного до максимально реалистично приближенного к фотографическому.
3. общение в режиме реального времени: все коммуникации осуществляются практически мгновенно.
4. интерактивность: пользователи так или иначе взаимодействуют с контентом 3D-мира, возможно даже изменяют, добавляют и удаляют 3D-объекты.
5. независимость существования: существование мира продолжается независимо от того, находятся ли отдельные пользователи в системе.

Пользователи виртуальной образовательной среды приобретают формы и свойства аватара — визуального представления человека на экране монитора [5]. Аватаром они перемещаются по виртуальному миру и видимы другим пользователям. Как правило, в качестве аватара в виртуальной образовательной среде используется человекоподобное существо. Человеческий мозг легко ассоциирует себя с аватаром, и то, что случается в виртуальном мире, подсознательно воспринимается как реальность, происходящая с самим человеком. Показано, в частности, что если аватар имеет вид заинтересованного и внимательно слушающего лекцию человека, то обучаемый склонен переносить это состояние на себя и уделять лекции больше внимания [6].

В отличие от виртуальных музеев, галерей, планетариев, полноценные образовательные 3D-среды не ограничены тематикой. Пользователи сами получают возможность создать или выбрать наиболее приемлемую для них тему занятий и получают для этого все необходимые средства [7].

Для поддержки обучения в виртуальной 3D-среде, а также для развития конкурентных преимуществ виртуальных образовательных сред по сравнению с традиционными методиками обучения необходимы образовательные инструменты [8]. Такие инструменты также имеют сходство с реальным миром и чаще всего привязаны к виртуальной доске. Виртуальная доска имеет некоторое визуальное сходство с обычной школьной доской, а её содержимое напрямую зависит от используемого инструмента.

Среди наиболее востребованных инструментов:

1. отображение на виртуальной доске слайдов презентации;
2. отображение заранее загруженных изображений;
3. отображение изображения из буфера обмена;
4. отображения текста из буфера обмена;
5. отображение части экрана учителя в виде статичной картинки;
6. отображение части экрана учителя в динамике;
7. отображения видео-изображения с веб-камеры;
8. отображение рукописных рисунков/графических аннотаций поверх размещенных статичных и меняющихся изображений.
9. голосовая связь и текстовый чат

Рассмотрим описанные образовательные инструменты подробнее.

Отображение слайдов презентации позволяют осуществить поддержку распространенной формы подачи материала в виде набора слайдов.

Отображение заранее загруженных изображений и изображений из буфера обмена позволяет показывать подготовленный материал в графической форме.

Отображение части экрана учителя в динамике дает возможность преподавателю показать ученикам учебный материал в браузере, видеоматериал, электронную книгу, интерактивные образовательные ресурсы, показать на практике работу с какой-либо программой и т.д.

Графические аннотации позволяют преподавателю на лету прокомментировать заранее подготовленный материал, иллюстрируя свою мысль более подробно или отвечая на заданный учениками вопрос.

Сочетание аватара и образовательных инструментов позволяет добиться улучшения восприятия учебного материала. Чтобы проиллюстрировать это, рассмотрим три модели обучения: аудио, видео и кинестетическую.

Аудио-обучающиеся нуждаются в том, чтобы информация была ими услышана, чтобы понять ее. Визуалы должны увидеть информацию, чтобы они стала для них ясной. Кинестетики, чтобы учиться, должны что-то делать, переживать какие-то ощущения. Это не является абсолютным ни для одного человека но, как правило, в различные моменты времени люди имеют предпочтения к той или иной модели.

При разговоре с веб-камерой визуальный канал ученика перегружен, но не догружен кинестетический. Аудио-учащиеся могут не заметить этой проблемы, но она может существовать для тех, с кем они ведут разговор.

В противоположность этому занятия в виртуальных 3D-средах работают задействуя все каналы восприятия. Видео-обучающиеся могут смотреть на аватара преподавателя, динамический или статический контект на виртуальных досках. Аудио-обучающиеся могут просто слушать или обращаться, изредка просматривая текст или слайды. Кинестетики могут передвигаться, рисовать на виртуальной доске или размещать на ней презентации и изображения, перемещать объекты, жестикулировать.

Таким образом, применимость виртуальных миров в образовании напрямую зависит от качества реализации образовательных инструментов и инструментов управления аватаром. Если аватарам и способам управления аватаром посвящено множество научных исследований, то тема реализации образовательных инструментов в виртуальных трехмерных образовательных средах практически не затрагивалась, несмотря на свою безусловную актуальность.

Теоретическая часть

Описанные образовательные инструменты можно разбить на группы:

1. Инструменты, требующие поддержки фильтрации изображений

2. Инструменты, требующие поддержки возможности приема/передачи динамической картинки

3. Инструменты, требующие поддержки операций рисования

Для инструментов, требующих поддержки возможности приема/передачи динамического изображения прежде всего следует определить способ упаковки/распаковки изображения. При этом надо учитывать, что в образовательных целях передача динамического изображения должно происходить в реальном времени, таким образом стандартные алгоритмы сжатия потокового видео сразу исключаются из рассмотрения. В настоящее время широкое распространение для сжатия изображений получили алгоритмы DCT (digital cosine transform - дискретное косинусное преобразование) и DWT (digital wavelet transform - дискретное вейвлет-преобразование).

Дискретное косинусное преобразование - одно из ортогональных преобразований, вариант косинусного преобразования для вектора действительных чисел. Применяется в алгоритмах сжатия информации с потерями, например, MPEG и JPEG. Это преобразование тесно связано с дискретным преобразованием Фурье и является гомоморфизмом его векторного пространства.

Дискретные вейвлет-преобразования относятся к вейвлет-преобразованиям, в которых вейвлеты представлены дискретными сигналами. Преобразование можно выполнить за $n * \log_2 n$ операций. Свойства преобразования используются для уменьшения избыточности в представлении дискретных сигналов как первый этап в компрессии данных.

При сравнении двух подходов, были выделены преимущества вейвлет-преобразований для реализации механизма передачи динамического изображения в образовательных инструментах:

1) Вейвлет-преобразования до 40% эффективнее по степени сжатия.

2) При сжатии DWT текст, размещенный на изображении, передается значительно лучше с точки зрения читабельности (рис. 1).

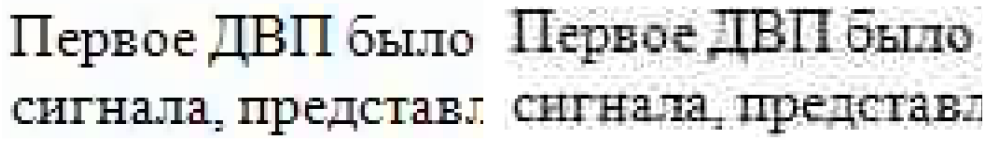

Рис. 1. Сравнение увеличенного изображения текста, сжатого алгоритмом DCT (справа) и с помощью вейвлет-преобразований (слева).

При этом у вейвлет-преобразований есть серьезный минус, заключающийся в значительно больших вычислительных затратах на сжатие и распаковку.

Фильтрация изображений и поддержка операций рисования, как и вейвлет-преобразования, требуют значительных вычислительных затрат. В образовательных виртуальных средах центральный процессор оказывается загружен множеством задач различной сложности, которые не могут быть выполнены на других вычислительных устройствах. Между тем, задачи обработки изображений могут быть перенесены для выполнения на графические потоковые процессоры.

Например, в работе [9] была предложена реализация вейвлет-преобразования на основе шейдера, а в работе [10] было проведено сравнение применения схемы каскада банков и схемы лифтинга для реализации дискретного вейвлет-преобразования на графических потоковых процессорах. Однако, в перечисленных работах было достигнуто умеренное ускорение (до 6 раз) по сравнению с реализацией на центральном процессоре, а сама реализация требовала обширных знаний, специфичных для графических потоковых процессоров.

Графические потоковые процессоры характеризуются большой степенью параллелизма, обладая сотнями и даже тысячами вычислительных ядер, но накладывают на выполняемые алгоритмы большое число ограничений (в основном,

из-за изначальной ориентированности на задачи 3D-графики), которые делают невозможным прямой перенос алгоритмов обработки изображений для выполнения на потоковых процессорах. Среди этих ограничений отсутствие рекурсии, прямого доступа к памяти на запись, неопределенный результат чтения из результирующего изображения и др.

Тем не менее, превосходство потоковых над обычными процессорами в десятки раз по вычислительной мощности делает перспективным обработку двумерных изображений именно на потоковых процессорах. Преимущество потоковым процессорам при обработке изображений для образовательных инструментов в виртуальной 3D-среде даст еще и то, что обрабатываемые изображения будут находиться в видеопамяти и результат также должен будет оказаться в видеопамяти. Между тем, центральный процессор не имеет прямого доступа к видеопамяти. В связи с этим потребовалась бы пересылка данных по шине AGP/PCI-E, которая требует значительного времени.

В настоящее время виртуальные миры представлены как в виде отдельных клиент-серверных приложений, которые устанавливаются пользователем как отдельный программный продукт, так и в виде браузерных приложений, не требующих установки.

Не менее перспективной является идея использования потоковых процессоров для обработки изображений в браузерных приложениях в связи с их ограниченным доступом к вычислительным возможностям центрального процессора. Прямое исполнение кода на центральном процессоре из браузерного приложения невозможно без использования специально разработанного дополнительного модуля для браузера (плагина). Механизм поддержки плагинов варьируется от браузера к браузеру, что вносит дополнительные трудности и требует разработки плагинов под каждый поддерживаемый браузер.

Долгое время обработку изображений в браузерных приложениях можно было реализовать с помощью двух основных подходов: с использованием Java-апплета и с использованием Flash-компонента. Так, например, для обработки изображений в java-апплетах широко используется библиотека ImageJ [11].

При использовании обоих подходов требуется установка специальных плагинов, что является во многих случаях нежелательным или невозможным (по соображениям безопасности, ввиду отсутствия плагина для нужной платформы или браузера, ввиду недостаточной квалификации пользователя для установки, а также по другим соображениям). Кроме того, в обоих случаях обработка изображений производится байт-кодом, выполняемым на виртуальной машине Flash или Java. Таким образом, скорость обработки изображений будет значительно уступать скорости программ на компилируемых языках, таких как C/C++ и тем более скорости обработки изображений на графических потоковых процессорах.

С появлением стандарта HTML5 стала возможна обработка изображений непосредственно на языке javascript с использованием элемента canvas [12]. Преимущество метода заключается в отсутствии необходимости установки каких-либо плагинов, однако

1. Программа обработки изображений выполняется на виртуальной машине javascript.

2. Использование возможностей многоядерных процессоров затруднено.

Эти два фактора негативно отражаются на производительности подхода. По приводимому в [12] графику видно, что скорость описываемого подхода в несколько раз уступает скорости обработки изображений с помощью ImageJ (рис. 2).

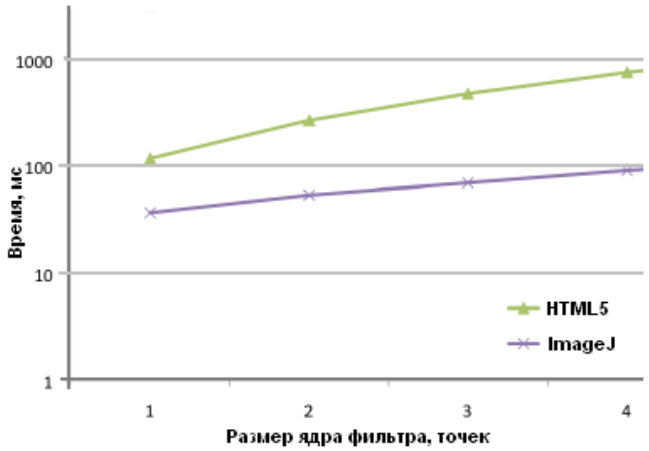

Рис 2. Сравнение производительности box-фильтра с помощью HTML5 canvas и ImageJ.

В настоящее время все большее распространение получает технология PixelBender [13], реализованная в Adobe Flash начиная с версии 10.0. Эта технология позволяет в ограниченном объеме использовать возможности потоковых процессоров, разрабатывая программы для них на специальном языке, специфичном для Flash.

Существенные минусы PixelBender:

1. Технология реализована внутри Flash, а значит требует установки плагина и имеет все перечисленные проблемы технологий, требующих плагинов.

2. Специальный язык ограничивает доступ к вычислительным возможностям потоковых процессоров, ограничивает простор для оптимизации и т.д.

10 февраля 2011 года была опубликована спецификация открытого стандарта WebGL [14] – реализации OpenGL ES 2.0 для использования в браузерных приложениях. WebGL дает возможность разрабатывать OpenGL-программы непосредственно на javascript, включая использование вершинных и пиксельных шейдеров. Таким образом, становится доступной вся функциональность потоковых процессоров.

Для использования WebGL не требуется установки дополнительных модулей, поддержка технологии реализуется непосредственно в браузере. Перечисленные качества делают технологию перспективной для организации обработки изображений в браузерных приложениях.

Таким образом, предлагаемый подход подходит как для реализации образовательных инструментов в виртуальных мирах, выполненных как отдельное приложение, так и виртуальных средах, работающих непосредственно в браузере. При этом одним из важнейших качеств предлагаемых системы должно быть то, что она не будет требовать от использующего его программиста знаний особенностей архитектуры и подходов программирования графических потоковых процессоров.

Реализация (практическая часть)

МАТЕМАТИЧЕСКАЯ МОДЕЛЬ

Составим математическую модель обработки изображений, опираясь на особенности и возможности OpenGL/WebGL. Будем рассматривать изображение в цветовой модели RGBA:

$U(x, y) = \{f_R(x, y), f_G(x, y), f_B(x, y), f_A(x, y)\}$ (1),

где $f_R(x, y)$, $f_G(x, y)$, $f_B(x, y)$, $f_A(x, y)$ — это дискретные функции, заданные табличным методом. $f_R(x, y)$ представляет красный канал изображения, $f_G(x, y)$ - зеленый, $f_B(x, y)$ - синий, $f_A(x, y)$ - альфа-канал. Значения этих функций лежат в диапазоне [0, 1].

Результатом преобразования G изображения A на основе изображения B будем называть

$R = G(A, B, x, y)$ (2),

где A — это исходное изображение, B – растеризуемая фигура, G — преобразующая функция.

Цветовой маской будем называть множество

$M = \{M_R, M_G, M_B, M_A\}$ (3),

где M_R, M_G, M_B, M_A могут принимать значения из множества {0, 1}. Результатом преобразования G изображения A на основе изображения B с учетом цветовой маски M будем называть

$R_q = G_q(A, B, M, x, y) = G_q(A, B, x, y) * M_q + A * (1 – M_q)$ (4),

где q – любой из цветовых каналов.

Геометрическая фигура $S = \{V, F, \{r, g, b, a\}\}$ (5) задается множеством двумерных векторов вершин V и множеством индексов вершин F, а также цветом фигуры {r, g, b, a}. Растеризация представляет собой преобразование, результатом которого является изображение:

$U(x, y) = G_R(G_P(S, M_P))$ (6),

где G_R – растеризующее преобразование, M_P – матрица проецирования, G_P — проецирующее преобразование.

В результате проецирующего преобразования G_P получается спроецированная фигура

$S_P = G_P(S, M_P) = \{P, F, \{r, g, b, a\}\}$ (7)

Спроецированная фигура S_P задается множеством двумерных векторов спроецированных вершин $P = \{P_1,\ldots, P_N\}$, множеством индексов вершин $F = \{F_1, ..F_{3*M}\}$, а также цветом фигуры {r, g, b, a}. Отметим, что для всех $i \in \{1..N\}$

$P_i = V_i * M_P$ (8)

$G_R(x, y) = \{\{r, g, b, a\}$, если существует тройка $(F_{3*k+1}, F_{3*k+2}, F_{3*k+3})$, где k – целое число от 1 до M / 3, такая, что (x, y) принадлежит треугольнику, образованному вершинами P_{i1}, P_{i2}, P_{i3}, где $i1 = F_{3*k+1}$, $i2 = F_{3*k+2}$, $i3 = F_{3*k+3}$, {0.0, 0.0, 0.0, 0.0} в противном случае} (9).

МОДЕЛЬ ФИЛЬТРАЦИИ ИЗОБРАЖЕНИЙ

На основе анализа архитектуры и особенностей потоковых процессоров [15] и возможностей OpenGL/WebGL была разработана следующая модель фильтрации изображений. В основе модели лежат четыре основные понятия: Texture, Drawing Target, Filter и Filter Sequence. Texture - это изображение в форме (1), хранимое в памяти графического потокового процессора. Drawing Target - это объект, который определяет изображение-результат и цветовую маску (3).

Filter - задает преобразование изображения (2) и в общем случае представляет собой функцию с набором предопределенных и пользовательских параметров, возвращающую цвет точки для заданных координат. Функция, например, может быть задана на GLSL-подобном языке, расширенном дополнительными функциями для обработки изображений. Параметры функции строго типизированы, имеют уникальные имена и определяются описанием в форме XML. В качестве растеризуемой фигуры B используется триангулированный прямоугольник с размером совпадающим с размером результирующего изображения R.

```
<filter>
  <function name="GetColor">
    <arguments>
      <image>buffer</image>
```

```
      <image>oldBuffer</image>
    </arguments>
    <body>
      vec4 newColor = (buffer.getOfsPixel(-1.0, -1.0) +
      buffer.getOfsPixel(-1.0, 0.0) + buffer.getOfsPixel(-1.0, 1.0) +
      buffer.getOfsPixel(0.0, -1.0) + buffer.getOfsPixel(0.0, 1.0) +
      buffer.getOfsPixel(1.0, -1.0) + buffer.getOfsPixel(1.0, 0.0) +
      buffer.getOfsPixel(1.0, 1.0)) / 4.0;
      vec4 oldColor = oldBuffer.getCurrentPixel();
      float clr = newColor.r - oldColor.r;
      if (clr less 0.32)
        clr = 0.32;
      if (clr more 1.0)
        clr = mod(clr, 1.0);
      return vec4(clr, clr, clr, 1.0);
    </body>
  </function>
</filter>
```

Листинг 1: Пример фильтра для расчета кадра дождевой поверхности

В листинге 1 приведен пример фильтра, заданного в предлагаемой форме. Функция GetColor должна возвратить цвет текущего обрабатываемого пикселя, аргументами функции являются два изображения buffer и oldBuffer. Функция getCurrentPixel возвращает цвет текущего пикселя в заданном изображении, а getOfsPixel цвет пикселя со смещением относительно текущего. Типы используемых вне декларативного объявления переменных соответствуют типам данных языка GLSL.

```
<filter>
  <function name="GetColor">
  <arguments>
    <image>background</image>
    <image>rain</image>
  </arguments>
  <body>
      vec4 newColor = background.getCurrentPixel();
      vec4 oldColor = rain.getPixel(%x, %height - %y) * 0.5;
      vec4 oldColor2 = oldColor + 0.5;
      return vec4(newColor.r * oldColor2.r + oldColor.r, newColor.g * oldColor2.g +
oldColor.g, newColor.b * oldColor2.b + oldColor.b, 1.0);
  </body>
  </function>
</filter>
```

Листинг 2: Пример фильтра для применения рассчитанной дождевой поверхности на произвольное изображение.

В листинге 2 приведен еще один пример фильтра. В этом фильтре осуществляется прямой доступ к пикселям изображения с помощью функции getPixel, а координаты рассчитываются согласно значениям предопределенных переменных %x и %y (координаты текущего пикселя), %width и %height (размеры результирующего изображения).

В основе модели фильтрации лежит условие: изображение-результат и изображения-параметры одного и того же фильтра не могут совпадать, т.е.

$$F(X) \neq X \qquad (10)$$

Это условие заложено в модель в целях обеспечения корректности выполнения преобразований на потоковых процессорах, т.е. для организации независимой параллельной обработки фрагментов изображения.

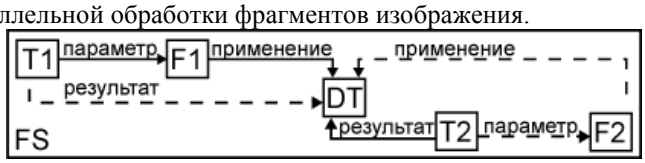

Рис.3: Взаимоотношения объектов в системе на пример filter sequence FS типа "пинг-понг".

Filter Sequence - задает последовательность нескольких фильтров с их параметрами, в общем случае позволяя организовать конвейер обработки, на основе которого можно реализовать сложное преобразование для обработки изображений. Таким образом, Filter Sequence представляет собой композицию преобразований изображений и в общем случае выражается формулой

$G(X) = G_1(G_2(..G_i(X)))$ (11),

где G(X) - преобразование, задающее Filter Sequence, а G_1, G_2... G_i - составляющие его преобразования-фильтры.

На примере в рис. 3. текстура T1 является параметром фильтра F1, который применяется к drawing target DT. Текстура T2 является результатом применения фильтра F1 к DT. В тоже время текстура T2 является параметром фильтра F2, который применяется к DT. Текстура T1 является результатом применения фильтра F2 к DT. Сначала применяется F1, затем F2, исходное изображение берется из T1, результат оказывается там же. Сплошными линиями показаны связи, существующие во время применения первого фильтра, пунктирными - второго.

МОДЕЛЬ ПОДДЕРЖКИ ОПЕРАЦИЙ РИСОВАНИЯ

Операции рисования являются отдельным классом задач обработки изображений как при реализации на центральном процессоре, так и на потоковых. Способы реализации поддержки рисования в системах обработки изображения на центральном процессоре детально проработаны. Однако, при использовании потоковых процессоров классические алгоритмы (например, алгоритм Брезентхема) оказываются неприменимы из-за архитектурных ограничений, что требует разработки методов поддержки операций рисования специально для использования с применением потоковых процессоров.

Теоретически, геометрическая фигура может быть отрисована непосредственно по аналитическому описанию с применением шейдерных программ, что свело бы эту задачу к задачам фильтрации. Однако, эффективность такого подхода невелика, т.к. фактическая площадь отрисовываемой фигуры может быть на порядки меньше, чем площадь растеризовываемого триангулированного прямоугольника. Даже если в качестве данного прямоугольника брать ограничивающий объем фигуры (bounding box), неэффективность подхода остается значительной (см. рис. 4).

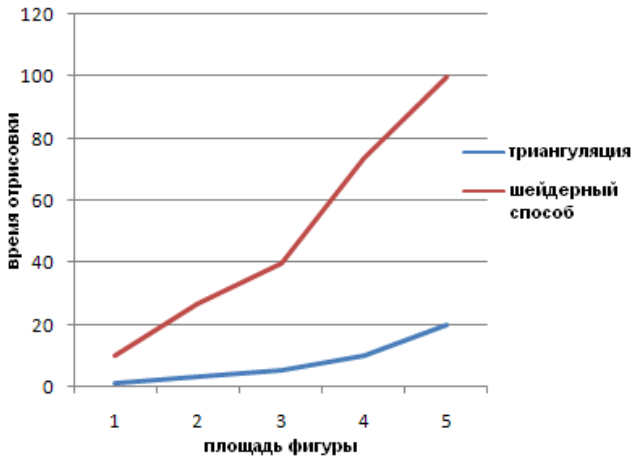

Рис. 4: Замер производительности поддержки операций рисования с помощью растеризации по аналитическому описанию и с помощью триангуляции.

Поэтому для более эффективной реализации необходимо триангулировать геометрические фигуры, т.е. представлять их в форме выражения (5). Для триангуляции каждой фигуры предлагается использовать частный, наиболее оптимальный для нее метод триангуляции, т.к. триангуляция в общем виде избыточно сложна.

Программная архитектура

Программная архитектура (рис. 5) предполагает поддержку некоторой абстрактной палитры инструментов, которая может быть расширена новыми инструментами для реализации поддержки требуемых операций. Инструменты самостоятельно выполняют триангуляцию фигур и передают на растеризацию триангулированную фигуру с установленными параметрами цвета и прозрачности.

Для каждого изображения с поддержкой операций рисования создается объект палитры инструментов, а также Drawing Target. Он содержит в себе основную текстуру, временную текстуру и буфер глубины. Размер основной и временной текстуры совпадают. Реализация текущего инструмента может запросить сохранение во временную текстуру изображения из основной текстуры перед очередным этапом работы. До окончания этапа содержимое временной текстуры может быть скопировано в основную, а поверх растеризована актуальная рисуемая фигура. Под этапом понимается полный цикл рисования одной фигуры (линии, прямоугольника и т.д).

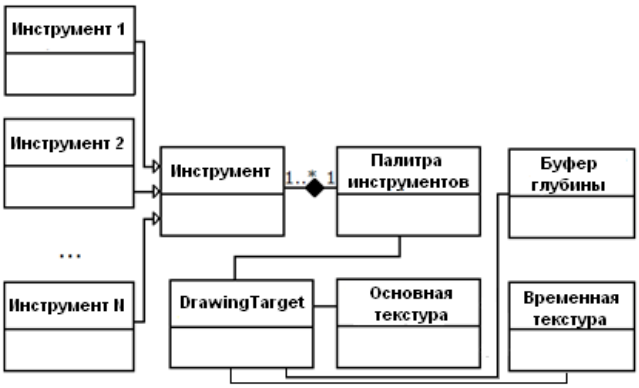

Рис. 5: Архитектура инструмента

Объект палитры инструментов содержит в себе коллекцию инструментов и хранит информацию о том, какой из инструментов активен в текущий момент. Этот

объект получает уведомления о событиях, инициирующих процесс отрисовки (например, от мыши, от управляющего сервера и т.д.), и передает их текущему инструменту. Все инструменты удовлетворяют одному и тому же программному интерфейсу, так что палитра инструментов взаимодействует с любым конкретным инструментом без учета того, как этот инструмент устроен внутри.

Поддержка прозрачности

Для поддержки прозрачности предлагается использовать композицию из двух преобразований. Первый проход выражается преобразованием

$$G(C, D, x, y) = \min(C(x, y) + D(x, y), 1.0) \quad (12),$$

где C и D представляют собой функции одного и того же канала изображения. Это преобразование применяется только к альфа-каналу с помощью маски (3). Оно соответствует поддерживаемому аппаратно преобразованию с параметрами source factor = GL_ONE, destination factor = GL_ONE, blend mode = GL_FUNC_ADD.

Второй проход выражается преобразованием

$$G(A, B, x, y)_q = \min(A(x, y)_q * B(x, y)A + (1 - B(x, y)A) * B(x, y)_q), 1) \quad (13)$$

Преобразование (13) применяется только к RGB-каналам изображения. Оно соответствует поддерживаемому аппаратно преобразованию с параметрами source factor = GL_SRC_ALPHA, destination factor = GL_ONE_MINUS_SRC_ALPHA, blend mode = GL_FUNC_ADD. Полученное преобразование удовлетворяет предъявляемым к прозрачности требованиям:

1. Смешивание RGB-каналов должно производиться с учетом значения альфа-канала растеризуемой фигуры.
2. При растеризации фигуры на абсолютно непрозрачную точку изображения, значение альфа-канала не должно измениться.
3. При растеризации фигуры на абсолютно прозрачную точку изображения, значение альфа-канала должно увеличиться.

МОДЕЛЬ КОМПРЕССИИ/ДЕКОМПРЕССИИ ДИНАМИЧЕСКИХ ИЗОБРАЖЕНИЙ

Определение изменившихся областей

Чаще всего динамическое изображение меняется от кадра к кадру незначительно или частично. Это значит, что для передачи нового кадра можно использовать только набор изменившихся областей. Динамическое изображение делится на NxM прямоугольных областей, причем размеры каждой области составляют:

$$w' = w / N \quad (14),$$

$$h' = h / M \quad (15),$$

где w’ – ширина прямоугольной области, h’ – высота прямоугольной области, w – ширина всего изображения, h – высота всего изображения.

Определение изменившихся областей осуществляется с помощью фильтрации прямоугольных областей специальным фильтром с occlusion query. Фильтр применяется к каждой прямоугольной области отдельно и записывает в результирующее изображение точку, если соответствующие точки (с совпадающими координатами) i-ого и (i + 1)-го кадра не совпадают. В противном случае точка не записывается (операция discard). Occlusion query возвращает число изменившихся пикселей внутри прямоугольной области. Таким образом, по количеству изменившихся пикселей можно принять решение о том, изменилась прямоугольная область или нет.

В результате описанной проверки получается множество изменившихся прямоугольных областей изображения:

$$O = \{O_1, O_2, \ldots, O_N\} \quad (16)$$

Множество полученных прямоугольных областей обрабатывается с помощью дискретного вейвлет-преобразования, а затем полученный результат запаковывается с помощью одного из стандартных алгоритмов компрессии данных без потерь.

Дискретное вейвлет-преобразование

В его качестве было выбрано 1D вейвлет-преобразование по схеме лифтинга, т.к. аналогичное 2D вейвлет-преобразование на тестовом наборе изображений приводило к небольшому уменьшению размера данных после паковки используемым методом архивирования в то время как значительно увеличивало вычислительную сложность, что было особенно критично для варианта решения задачи на центральном процессоре.

Вейвлет-преобразование осуществлялось для цветового пространства YUV, для чего исходный цвет конвертировался из цветового пространства RGB в YUV. После этого часть изображения, содержащая каналы UV, уменьшалась в N раз с осреднением цвета (т.к. человеческий глаз слабо различает цветовые составляющие по сравнению с яркостной) и проходила квантование с коэффициентом D по формулам (17).

$$U = U / D,\ V = V / D \qquad (17)$$

Для результата применялось вышеописанное дискретное вейвлет-преобразование с K проходами по схеме лифтинга. Критерием отброса пикселя как незначительного на I-ом проходе лифтинга было неравенство (18).

$$| C_x - (C_{x-2^{I+1}} + C_{x+2^{I+1}}) | < E \qquad (18)$$

где Cx — цвет точки с абсциссой x, а E — допустимое отклонение аппроксимированного значения цвета пикселя от точного.

При подготовке к распаковке часть изображения, содержащая UV-каналы увеличивалась в N раз, проходила квантование с коэффициентом 1 / D, а цвет пикселя определялся по формуле (19).

$$C_x = \begin{cases} C_x, \text{ если } C_x > 0 \\ (C_{x-2^{I+1}} + C_{x+2^{I+1}}) / 2, \text{ если } C_x = 0 \end{cases} \qquad (19),$$

где I удовлетворяет равенству (20).

$$x \bmod 2^{I+1} = 2^I \qquad (20)$$

Прямое дискретное вейвлет-преобразование

Подготовка изображений к упаковке происходит в два преобразования. В ходе первого преобразования применяется фильтр, который переводит цвет из пространства RGB в YUV и осуществляет уменьшение изображения по каналам UV с одновременным их квантованием. Используются функции системы для квантования цвета, оптимизированного осреднения цвета и выборки текущего пикселя.

Второе преобразование представляет собой непосредственно прямое дискретное преобразование, в котором одновременно выполняются все шаги лифтинга. Для этого для каждого пикселя в цикле прямым перебором находится шаг лифтинга I, удовлетворяющий равенству (20), что оправдано т.к. стоимость избыточных арифметический операций значительно меньше стоимости дополнительных проходов. При осуществлении проверки неравенства (2) пересечения с другими шагами лифтинга не происходит, т.к. согласно (10) изображение-результат и изображение-параметр различаются. Используются функции системы для выборок из текстуры со смещением относительно текущего пикселя, выборки текущего пикселя и встроенные свойства текущего пикселя, зависящие от используемой текстуры.

Обратное дискретное вейвлет-преобразование

Подготовка изображения к распаковке происходит в K + 1 этапов. Первые K этапов представляют собой последовательность фильтров, реализующую K шагов лифтинга обратного ДВП. В связи с необходимостью использовать на более поздних шагах результаты более ранних и ограничением (10), объединить дискретное вейвлет-преобразование в 1 проход не удалось.

Вместо этого используется filter sequence типа «пинг-понг» (см. рис. 2), при котором для всех K этапов используется один и тот же фильтр с разным коэффициентом шага лифтинга и чередующимися изображением-результатом и

исходным изображением. Используются функции системы для выборок из текстуры со смещением относительно текущего пикселя и возможности попиксельной адресации.

На последнем этапе происходит деквантизация каналов UV, восстановление исходных размеров изображения по этим каналам и преобразование цвета из пространства YUV в RGB. Используются функции системы для квантования цвета, выборки текущего пикселя и выборки с произвольной попиксельной адресацией.

Анализ и оценка разработки

На основе моделей обработки изображений была разработана система обработки изображений, которая была интегрирована в виртуальную образовательную 3D-среду «Виртуальная академия» [16].

В табл. 1 и 2 приведено усредненное время подготовки изображений к упаковке и распаковке с помощью вейвлет-преобразования [17]. В качестве ГПП использовался NVidia GF8800 GT. В качестве ЦП выступал Core 2 Quad 2.4 ГГц (4 ядра). Реализация алгоритма под ЦП использовала многоядерность. На основе вейвлет-преобразования в Виртуальной академии работают отображение части экрана учителя в динамике и отображения видео-изображения с веб-камеры (рис. 6).

Табл. 1. Время подготовки к паковке изображения (в мс).

	128x128	256x256	512x512	1024x1024	2048x2048
ГПП	0,3	0,4	0,5	1,6	6,2
ЦП	5,1	10,3	31,4	120,1	479,2

Табл. 2. Время подготовки к распаковке изображения (в мс).

	128x128	256x256	512x512	1024x1024	2048x2048
ГПП	1	1,1	1,5	2,7	10,5
ЦП	3,5	7,1	21,1	84,3	331,9

Из анализа табличных данных видно, что преимущество потоковых процессоров при распаковке достигает ~75 раз, а при распаковке ~44 раз. Максимальное преимущество достигается при больших размерах изображения. Более низкая скорость работы графических потоковых процессоров при распаковке объясняется большим чем при паковке числом проходов. Существенной стоимостью прохода рендеринга как такового объясняется и нелинейная зависимость времени обработки от размера изображения.

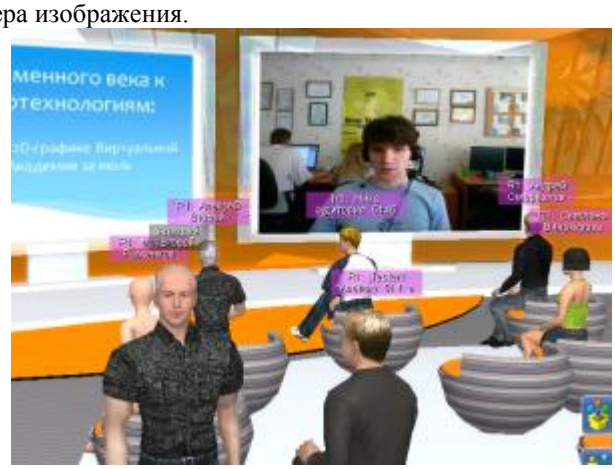

Рис. 6: Отображение видео-изображения с веб-камеры с помощью вейвлет-преобразований.

Экспериментальные замеры производительности операций рисования показали преимущество описанного подхода в 10-50 раз в зависимости от аппаратных конфигураций и условий тестирования [18]. Такое использование прежде всего достигается за счет использования возможностей системы в рамках 3D-приложения (рис. 7), т.к. содержимое виртуальной доски хранится в видеопамяти как текстура, чтобы отображать его в трехмерном пространстве. Это делает неэффективной работу образовательных инструментов на центральном процессоре из-за необходимости постоянной пересылки данных по шине данных AGP/PCI-E и соответственно дает преимущество описанному подходу.

Рис. 7: Инструмент графического комментирования материала в «Виртуальной Академии».

Был предложен перспективный подход к обработке изображений в браузерных приложениях, который позволяет осуществлять довольно сложные преобразования в режиме реального времени. Подход к обработке изображений (фильтрация) в браузерных приложениях успешно апробирован на ряде реальных примеров (рис. 8) и в ходе экспериментальных разработок плеера записанных виртуальных занятий (рис. 9).

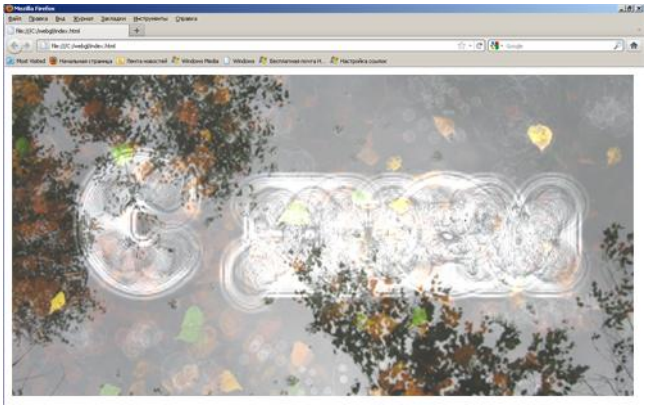

Рис. 8: Пример фильтра водной поверхности, работающего в реальном времени на потоковых процессорах в браузерном приложении в браузере Firefox 4.

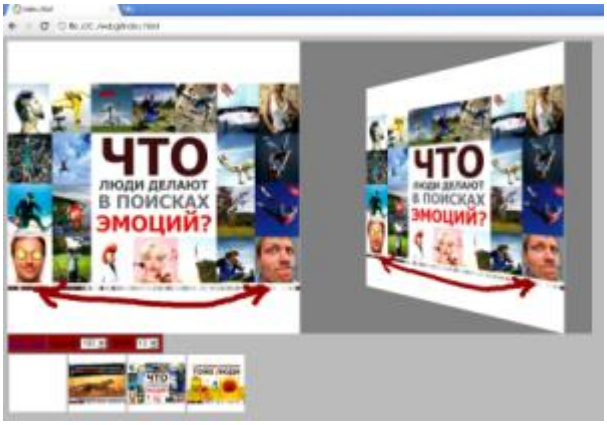

Рис. 9: Пример браузерного приложения, использующего фильтрацию и операции рисования в браузере Chrome.

Метод показывает высокую производительность. Так box-фильтр и фрактал Мандельброта рассчитываются предлагаемым методом в ~15-110 раз быстрее, чем при использовании подхода [5], в зависимости от выбранного сочетания центрального и графического потокового процессоров.

Заключение

Были предложены математические и программные модели для реализации образовательных инструментов в виртуальных образовательных 3D-средах с использованием обработки изображений на основе вычислительных возможностей графических потоковых процессоров. Эти модели были реализованы в программной системе обработки изображений, интегрированной в виртуальный образовательный мир «Виртуальная академия» и показали свою эффективность с точки зрения производительности на практике.

Предложенные методы и подходы могут быть использованы при разработке других образовательных виртуальных миров, в том числе работающих внутри браузера, что является преимуществом метода относительно стандартных алгоритмов обработки изображений на центральном процессоре.

В дальнейшем математическая и программная модель обработки изображений могут быть доработаны для дальнейшего повышения производительности, улучшения гибкости применяемых подходов и обхода части архитектурных ограничений графических потоковых процессоров.

Литература

1. Morozov, M., Tanakov, A. The Virtual Laboratory as an Active Learning Environment // eAdoption and the Knowledge Economy: Issues, Applications, Case Studies. Part 2, Amsterdam: IOS press, ISBN 1-58603-470-7, 2004 -1742-1748 pp

2. Fominykh, M. Virtual Campus in the Context of an educational Virtual City // The International conference on Educational Multimedia, Hypermedia & Telecommunications - 2009. - 22-26 pp.

3. Зеленко, Л.С. Применение технологии виртуальных миров при построении интерактивных обучающих систем // Третья международная научно-практическая конференция "Электронная Казань 2011" - 2011. - 127-133 с.- Т. 1-2.

4. Dickey, M. D. Three-dimensional virtual worlds and distance learning: two case studies of Active Worlds as a medium for distance education/ M. D. Dickey // British Journal of Educational Technology - Vol. 3 No. 36, 2005. - 439-451 pp.
5. Герасимов, А.В., Морозов, М.Н. Взаимодействие с персонажами в виртуальных образовательных средах.// Материалы IV Всероссийской научно-практической конференции "Применение информационно-коммуникационных технологий в образовании" - Йошкар-Ола: МарГТУ, 2007.- 108-111 с.
6. Алишева, Д.А. Обучение в виртуальных мирах, 2010 [электронный ресурс] / Режим доступа: http://anatom.paxgrid.ru/article1.php.свободный. - Загл. с экрана.
7. Diener S. This will change everything. Virtual Worlds in Education / S. Diener // Angel learning white paper - July 2009. - 30-38 pp.
8. Cross, J., O'Driscoll, T., Trondsen, E. Another life virtual worlds as tools for learning [Electronic resource] / J.Cross, T. O'Driscol, E. Trondsen // eLearn Magazine - Electronic data. - Mode access: http://www.elearnmag.org/subpage.cfm?section=articles&article=44-1, free.
9. Tien-Tsin Wong, Chi-Sing Leung, Pheng-Ann Heng, and Jianqing Wang. Discrete Wavelet Transform on Consumer-Level Graphics Hardware. IEEE Trans. on Multimedia, Vol. 9, No. 3, April 2007, pp. 668-673.
10. Tenllado, C., Setoain, J., Prieto, M., Pinuel, L., Tirado, F. Parallel Implementation of the 2D Discrete Wavelet Transform on Graphics Processing Units: Filter Bank versus Lifting. IEEE Trans. Parallel And Distributed Systems, vol. 19 no. 3, March 2008, pp. 299-310
11. Abramoff, M.D., Magelhaes, P.J., Ram, S.J. "Image Processing with ImageJ". Biophotonics International, volume 11, issue 7, pp. 36-42, 2004.
12. Kai Uwe Barthel, Karsten Schulz, ImageJ in the web - Image processing in the browser using HTML5. In proceedings of ImageJ Conference 2010.
13. Adobe Pixel Bender Guide. http://www.adobe.com/content/dam/Adobe/en/devnet/pixelbender/pdfs/pixelbender_guide.pdf
14. The Khronos Group. WebGL 1.0 Specification. https://www.khronos.org/registry/webgl/specs/1.0/
15. Fatahalian Kayvon, Houston Mike. A closer look at GPUs. Communications of the ACM, October 2008, Vol. 51, NO. 10.
16. www.vacademia.com
17. Сморкалов, А.Ю., Морозов, М.Н. Поддержка дискретных вейвлет-преобразований для компрессии в системе обработки изображений на потоковых графических процессорах. Сборник материалов Всероссийской научно-практической конференции с международным участием "Информационные технологии в профессиональной деятельности и научной работе". - Йошкар-Ола: МарГТУ, 2010.- С. 91-96.
18. Сморкалов, А.Ю. Поддержка операций рисования в системе обработки растровых изображений на потоковых процессорах. Сборник материалов Всероссийской научно-практической конференции "Информационные технологии в профессиональной деятельности и научной работе". - Йошкар-Ола: МарГТУ, 2011.- С. 201-206.