Анализ алгоритмов оптимизации времени отображения электронных карт в формате HP-GL.

Ю.Л. Кетков
НИИ ПМК ННГУ
Нижний Новгород, Россия
ket@unn.ru

З.А. Матвеев
Intel Corporation
Нижний Новгород, Россия
zakhar.a.matveev@intel.com

Abstract

The visualization performance becomes critical when checking the digital maps completeness and quality; especially since typical map size is about 10-100 Mb of vector graphical data. In this paper two base performance improvement methods proposed. They are: data pre-processing technique and spatial indices-like data structure usage.

Generalization, hierarchical indexing, spatial selection criteria approaches; local offset optimization, special polyline 2-dimensional scissoring and task parallelism-based modification of algorithms introduced and analyzed as well.

Experimental results illustrate computational efficiency of proposed approaches in comparison with other visualization software.

Keywords: *Visualization, Indices, Digital map, Cartography, Scissoring.*

1. ВВЕДЕНИЕ

Бурное развитие цифровой картографии за последние годы привело к появлению большого числа ГИС различного назначения, в том числе автоматических и автоматизированных картографических систем (АКС). Одной из задач, решаемых с помощью профессиональных АКС, является осуществление контроля полноты и качества цифровых моделей топографических карт (ЦТК).

В большинстве случаев исходным материалом для создания ЦТК являются стандартные тиражные оттиски, при подготовке которых существует довольно много отступлений от регулярных правил. Например, допускается изменять характеристики шрифтов, отступать от типовых размеров дискретных знаков, варьировать шаг расположения заполнителей линейных знаков, выносить надписи в местах большой насыщенности документа и т.п. Кроме того в процессе оцифровки, автоматического распознования карты возникают самые разные ошибки, связанные с погрешностью позиционирования, деформацией исходных документов, проблемами классификации картографических объектов и их пространственно-логических связей.

В результате электронная версия картографического документа может существенно отличаться от оригинала. Поэтому как на этапе первоначального наполнения, так и при оперативном обновлении содержания, ЦТК обязательно должны контролироваться с помощью автоматических средств или путем визуального просмотра. Это особенно необходимо в случае подготовки издательских оригиналов цифровых карт, т.к. твердые (бумажные) копии еще не потеряли своей актуальности.

Требования, предъявляемые руководствами по изданию карт, достаточно высоки и для соблюдения установленных правил редактору-картографу приходится тщательно исследовать каждый фрагмент изображения. С учетом размеров листов карт (габариты морских карт могут достигать порядка 1000 мм) и габаритов просматриваемых объектов (минимальная высота некоторых надписей составляет 1–1.2 мм) на экране дисплея приходится отображать фрагменты электронной карты с увеличением в 4-8 раз. Когда размер файла с графическим изображением электронной карты достигает нескольких десятков и сотен Мбайт, быстродействие системы отображения становится решающим фактором, напрямую влияющим на эффективность внедрения АКС. Однако оперативность многих современных графических систем, работающих как с растровыми, так и с векторными изображениями такого размера, оставляет желать лучшего.

Просмотр содержимого графического файла, масштабирование различных его фрагментов и навигация по смежным фрагментам – операции массовые, требующие активного взаимодействия с пользователем. Для выполнения этих операций обычно используется специальная программа просмотра (визуализатор) графических файлов. В статье рассматриваются алгоритмы и методы, позволяющие повысить эффективность визуализаторов ипспользуемых в составе АКС для просмотра электронных карт.

2. ПРЕДВАРИТЕЛЬНЫЙ АНАЛИЗ

Экспериментальный анализ и апробация алгоритмов оптимизации выполнялись в процессе проектирования и разработки визуализатора векторных ЦТК в формате HP-GL/2 [7] (именно в этом формате формируются электронные карты современными версиями АКС, разрабатываемыми на базе НИИ ПМК Нижегородского государственного университета).

Формат HP-GL/2 (Hewlett-Packard Graphics Language) является на сегодняшний день de facto промышленным стандартом для производителей серийных большеформатных

плоттеров (применяемых для воспроизведения цветных оттисков) и специализированных прецизионных фотоплоттеров (используемых при подготовке издательских оригиналов и заливочных шаблонов).

Образ электронной карты в формате HP-GL представляет собой линейную графическую программу (без циклов и ветвлений), ориентированную на управление плоттером и содержащую достаточно длинную цепочку команд вида:

CD [p1,p2,...];

где CD – двухсимвольный код графической команды, а p1,p2,... – обязательные или необязательные числовые параметры, записанные в символьном формате. В качестве параметров могут выступать целочисленные и/или вещественные координаты, радиусы окружностей, углы, характеристики пишущего узла и др. Таким образом, электронная карта в формате HP-GL/2 это – обычный текстовый файл, каждая строка которого представлена графической командой.

Наиболее многочисленную группу в файле представляют команды воспроизведения различных графических примитивов – отрезков прямых той или иной толщины, залитых и не залитых окружностей, дуг, круговых и эллиптических секторов, фрагментов кубических парабол. Особенно много таких примитивов порождается при обработке надписей. Аппаратура плоттеров, как правило, не содержит графические описания кириллических шрифтов, поэтому каждую русскую букву приходится формировать из достаточно большого количества графических примитивов. При этом надо учитывать, что дизайн российских карт, отсчитывающий свою историю от времен правления Петра I, предполагает использование только для топографических карт 26 различных символьных наборов.

Лист топографической карты со средней нагрузкой может содержать порядка 10^6 - 10^7 графических команд. Получив от пользователя заказ на отображение соответствующего фрагмента листа, программа визуализации должна последовательно обработать все команды в графическом файле, и воспроизвести их на экране с соответствующим масштабным коэффициентом. На полном листе карты таких фрагментов от 100 до 1000, и к некоторым из них приходится обращаться не по одному разу. Поэтому лобовой вариант интерпретации команд графического файла (с повторным синтаксическим анализом и масштабированием для каждого очередного фрагмента) представляется крайне неэффективным.

Сформулируем ряд проблем, решение которых позволило бы повысить эффективность системы отображения при обработке графических файлов большого размера.

Внутренний формат хранения данных.

Необходимо ли при многократных просмотрах фрагментов одной и той же карты каждый раз заниматься синтаксическим анализом текстового файла и преобразованием графических команд из символьного представления в соответствующие машинные форматы? Очевидно, что это не целесообразно: такую интерпретацию следует проделать один раз и при просмотре новых фрагментов нужно обрабатывать информацию, полученную при первичном анализе. Реализация такого подхода связана с созданием походящего формата внутреннего представления, исключающего повторение уже проделанной работы.

Следует отметить, что введение подобного внутреннего формата позволяет заблаговременно (на этапе открытия файла) вычислить различные дополнительные характеристики объектов (площадь, взаимосвязи с другими объектами и пр.), которые при работе с фрагментом электронной карты позволяют ускорить его обработку.

Отбор данных для последующей визуализации.

Есть ли необходимость воспроизводить всё множество графических команд электронной карты при просмотре её фрагмента? С одной стороны, возможно при каждом изменении коэффициента масштабирования растеризовать всю карту целиком, сохраняя её в виде промежуточного растрового изображения. Однако в этом случае возрастёт объём оперативной памяти компьютера, используемой для хранения промежуточного растра. Кроме того, при любом изменении масштаба карты – будет выполняться генерация растрового представления всей карты заново, что повлечёт существенные временные потери.

Альтернативный подход основывается на идее динамического отбора графических команд, в зависимости от параметров запрашиваемого фрагмента (масштаб и положение на листе карты). Так, при каждом изменении масштаба можно обрабатывать только те объекты, которые пересекаются с указанным кадром (двумерное отсечение).

Допустимы и некоторые другие критерии отбора. Например, при малом коэффициенте увеличения вполне целесообразно отбрасывать те объекты, которые заведомо не будут различимы человеческим глазом (генерализация карты). Однако, непосредственное применение названных критериев не всегда достаточно эффективно.

Эффективный доступ к информации и индексирование.

Как ускорить доступ к нужной информации при отборе графических команд, отображающих объекты, принадлежащие заданной области? К решению этой проблемы ведут два пути.

Во-первых, в состав таблиц с внутренним представлением можно включить различные данные, упрощающие поиск. В роли таких данных обычно выступают некоторые характеристики графического объекта, например координаты его "точки привязки" (той точки, в которую пишущий узел переводится перед началом построения объекта) или минимальных окаймляющих прямоугольников (MBR – Minimal Bounding Rectangle).

Другой подход заключается в индексировании внутренних данных, т.е. в построении системы указателей (индексов), обеспечивающих прямой доступ к графическим данным, инцидентным окну просмотра.

Отметим, что при рассмотрении каждого из этих подходов, необходимо определиться с термином "объект". Нужно ли считать таковым объект в смысле классификатора соответствующей предметной области или абстрагироваться от такой содержательной трактовки и перейти на уровень простейших графических примитивов? В первом случае количество объектов существенно уменьшится, при этом появится тематическая зависимость. И хотя внедрение тематической зависимости является несомненным

преимуществом, - для её поддержки может потребоваться, чтобы программа подготовки графического файла перед формированием образа очередного картографического объекта заносила в HP-GL файл команду-комментарий (благо в версии HP-GL/2 такая команда с кодом CO появилась), по которой можно было автоматически определять точку привязки. Во втором случае точка привязки и другие характеристики определяются элементарно, но зато количество объектов возрастает на один-два порядка.

В целом, вопросы представления графических объектов, организации индексирования и внедрения дополнительных данных тесно взаимосвязаны между собой, воздействуя на производительность визуализатора самым существенным образом.

Ускорение отсечения ломанных

За счет чего можно ускорить процедуру отсечения графических команд, воспроизводящих объекты, не попадающие в кадр просмотра? Очевидно, что только точка привязки объекта не позволяет судить об его принадлежности к заданному окну просмотра. Картографический объект может быть линейным и достаточно протяженным или площадным с размерами, превышающими габариты окна просмотра, в которое попадают какие-то фрагменты объекта. Даже графический примитив, соответствующий длинному отрезку прямой, может начинаться за пределами окна просмотра, но, тем не менее, пересекать кадр. Существует довольно много алгоритмов отсечения, описанных в литературе, но в нашей задаче имеется достаточно важная специфика – приходится анализировать принадлежность окну просмотра не одного отрезка или его части, а множества мелких отрезков, образующих ребра ломаных или расположенных достаточно близко друг к другу (например, в случае аппроксимации контуров букв). В этом случае затраты на процедуру отсечения можно сократить примерно вдвое за счет того, что результаты анализа конца текущего отрезка можно учесть при анализе положения начала следующего отрезка.

Оптимизация просмотра смежных фрагментов.

Можно ли сэкономить время просмотра при переходе к соседнему фрагменту, граничащему с текущим по одной из четырех границ? Плавная навигация по соседним областям листа карты – один из достаточно распространенных режимов работы редактора-картографа. В простейшем случае можно разбить лист карты на фиксированное число прямоугольных фрагментов и вести просмотр любого из них. Однако это не очень удобно. Во-первых, часто требуется менять коэффициент масштабирования как в сторону увеличения, так и в сторону уменьшения. Во-вторых, скачкообразный переход к новому фрагменту не позволяет проанализировать структуру карты в районе границ смежных фрагментов. При плавной навигации возникает желание воспользоваться частью ранее полученного растрового изображения и заново воспроизвести только обновленную часть кадра. Таким образом можно примерно на 30% сократить время формирования следующего кадра и снизить неприятный эффект скачкообразного перехода к следующему изображению.

По результатам анализа перечисленных проблем были предложены различные методы и алгоритмы оптимизации, описанию которых посвящены следующие разделы статьи. Отметим, что большинство рассматриваемых алгоритмов применимо не только к конкретному формату HP-GL/2, но и к векторным графическим форматам вообще.

3. ВНУТРЕННЯЯ МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ДАННЫХ.

Как отмечалось выше, синтаксический анализ исходного файла имеет смысл выполнять однократно с целью создания внутреннего представления данных. Для решения этой задачи желательно использовать специальный компонент ПО, ответственный за предобработку графических данных. Одна из задач подобного компонента – распознавание символьного кода графической операции, преобразование значения параметров команды из символьного представления в целочисленный машинный формат. Вторая задача заключается в объединении смежных команд в группы, соответствующие понятию абстрактного графического объекта и формированию атрибутов этого объекта.

Понятие объекта может варьироваться от элементарного примитива, представленного единственной командой языка HP-GL до тематически зависимого картографического объекта со всеми его характеристиками. На наш взгляд, может быть предложена некоторая промежуточная модель, представленная группами следующих графических примитивов:

1. Группа установочных объектов, задающих текущие цвет, толщину и тип окончания линии, используемых при визуализации графических данных (для формата HP-GL/2 формируются командами SP).
2. Группа не залитых полигонов/ломанных (применительно к формату HP-GL/2 – речь идёт о командах PD, PU, PM, EP)
3. Группа "криволинейных" объектов типа дуга, окружность, сектор и др. Для представления объектов этой группы, не используются координаты точек контура. По этому признаку к этой группе можно также отнести объекты текстовых меток.
4. Подгруппа полигональных объектов, описывающих залитые области (для формата HP-GL/2 генерируются командами FP).
5. Подгруппа объектов координатной основы.

Группировка объектов в составе внутреннего формата осуществляется иерархически с использованием шаблона проектирования “Компоновщик” [1]. При этом, например, объекты координатной основы – являются дочерними по отношению к объектам из группы не залитых полигонов, а те в свою очередь – являются дочерними по отношению к установочным объектам.

В процессе построения описываемой структуры данных для каждого объекта (за исключением объектов нижнего уровня) заблаговременно вычисляются координаты точки привязки и MBR (окаймляющего прямоугольника – от англ. minimal bounding rectangle). При этом MBR объектов i-го уровня (узлов дерева глубины i-1) иерархии определяется как окаймляющий прямоугольник, включающий MBR всех дочерних объектов i+1 го уровня (узлов дерева глубины i).

Можно привести следующие аргументы в пользу предлагаемой модели представления данных (и их классификации):

- Использование установочных объектов на верхнем уровне иерархии позволяет оперировать с данными внутреннего формата как с картографическими слоями, облегчая генерализацию и минимизируя количество переключений палитры в процессе визуализации.
- Низкоуровневые объекты координатной основы отличаются от объектов других групп. Прежде всего, данные объекты всегда являются листовыми в древовидной структуре данных. Во-вторых, количество этих объектов очень велико. Так, например для типовых ЦТК в формате HP-GL/2 число объектов этой группы достигает 1-5 млн. единиц. Поэтому размер конкретного экземпляра структуры данных внутреннего формата определяется, в первую очередь, количеством объектов этого типа, а производительность визуализатора – определяется способом обработки этих объектов.
- Применение объектов второго типа может посодействовать более эффективному использованию процедур графического API. Известно, например, что обращение к попдрограмме воспроизведения ломаной линии Windows GDI Polyline намного эффективнее, чем использование многократных вызовов функции LineTo. Поэтому агрегирование цепочек отрезков в составе одного объекта может заметно повысить эффективность визуализации (в особенности когда заведомо известно, что весь объект целиком лежит внутри текущего окна просмотра).
- Хотя криволинейные объекты и объекты-надписи могут быть представлены с помощью объектов-ломаных (криволинейные – могут быть только апроксисимированы), они требуют специальных алгоритмов индексирования и растеризации. Поэтому их целесообразно выделить в отдельную подгруппу и не сопостовлять с ними координатную основу.
- Применение шаблона “Компоновщик” упрощает иерархическую организацию данных и обход полученной древовидной структуры.

4. АЛГОРИТМЫ ИНДЕКСИРОВАНИЯ.

Предварительная обработка данных позволяет выполнять дальнейшее эффективное индексирование картографических документов при просмотре увеличенных фрагментов карты. При этом в качестве выходных данных алгоритма индексирования выступает специализированная структура данных, называемая индексом, содержащая номера тех графических объектов, которые удовлетворяют одному или нескольким критериям отбора.

Индекс строится динамически. Это означает, что для каждого нового окна просмотра индекс очищается и заполняется заново на основе выбранного критерия.

При этом внутреннее представление индекса также является иерархическим, что позволяет в частности, минимизировать число операций отсечения объектов. В самом деле, если объект верхнего уровня не попадает в индекс (то есть лежит целиком вне окна просмотра) – отпадает необходимость в индексировании всего множества его дочерних объектов, т.к. все они также заведомо не попадают в индекс. Аналогично, если объект верхнего уровня, лежит целиком внутри окна отсечения, нет необходимости применять критерий отсечения для всех его дочерних объектов, т.к. очевидно, что каждый из них попадает в индекс (см. рис. 1).

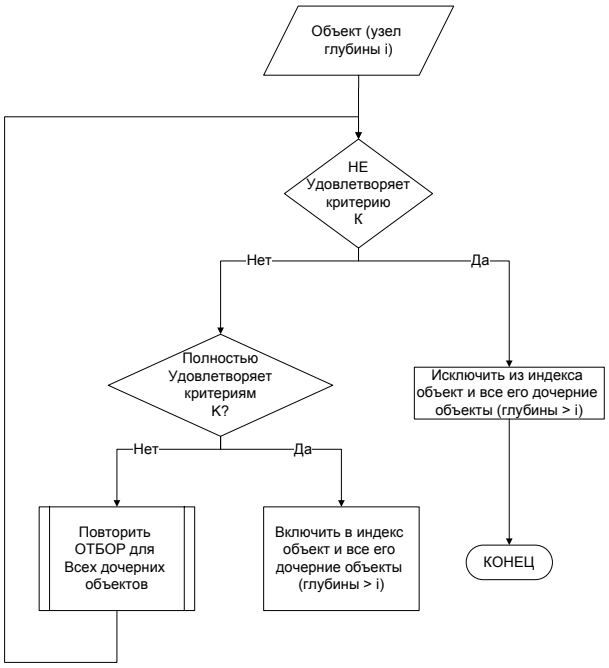

Рис. 1. Блок-схема иерархической индексации.

Для объектов, расположенных на первых двух уровнях иерархии основным критерием отбора является пересечение MBR объекта с текущим окном просмотра (“грубый критерий” отбора).

В случае если MBR объекта второго уровня лежит на границе окна просмотра – выполняется уточняющий шаг: двумерное отсечение дочерних объектов третьего уровня (отрезков). Для определения принадлежности очередной вершины отрезка окну просмотра используется информация о предыдущей точке, полученная на предыдущем шаге согласно упомянутому выше алгоритму отсечения ломанной.

Наряду с анализом взаимного расположения объектов и окна отсечения, при построении индекса учитываются относительные размеры объектов. В случае если при данном значении коэффициента масштабирования объект является неразличимым на экране дисплея (его площадь или периметр меньше некоторой величины, зависящей в свою очередь от коэффициента масштабирования) – он также не заносится в индекс. В данном случае уже используется количественный критерий генерализации.

Таким образом, индексирование выполняется на основе трёх критериев отбора информации: грубого пространственного, точного пространственного и количественного генерализационного.

На практике, в целях оптимизации, возможны некоторые отклонения от описанной схемы. Например, в случае если пользователь выполняет два последовательных запроса на отображение фрагментов с увеличением n и 2*n соответственно (без сдвига окна просмотра), при переходе от одного фрагмента к другому индекс не очищается, т.к. индекс, построенный при первом запросе, уже содержит все необходимые идентификаторы объектов - и остаётся лишь отбросить те, из них, которые не принадлежат новому фрагменту. Другая особенность используемого подхода заключается в том, что для залитых областей на практике не выполняется уточняющее индексирование, т.к. двумерное отсечение залитой области само по себе является трудоёмкой задачей.

5. ДОПОЛНИТЕЛЬНЫЕ СРЕДСТВА ОПТИМИЗАЦИИ ВРЕМЕНИ ОТОБРАЖЕНИЯ.

5.1 Оптимизация локальной навигации.

Не все разработанные алгоритмы оптимизации просмотра укладываются в схему предобработки и последующего индексирования карты. Так, например, существенный прирост производительности был достигнут за счёт реализации алгоритма локальной навигации (рис. 2).

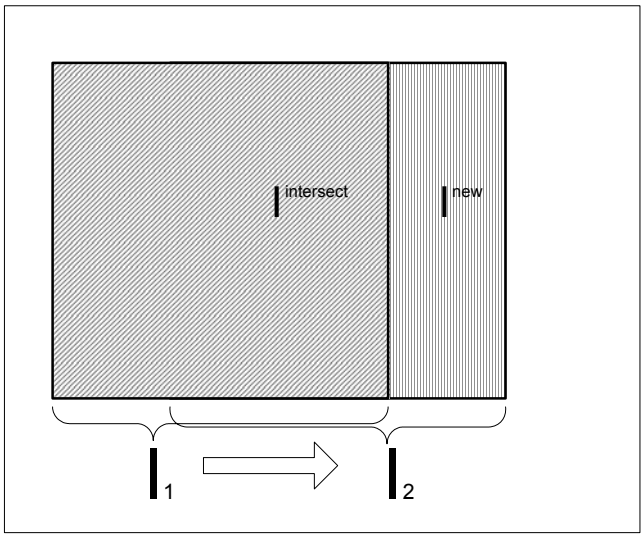

Рис. 2. Алгоритм локальной навигации.

Этот алгоритм применяется при работе в режимах панорамирования и фиксированного смещения без изменения масштаба. Оба названных режима подразумевают последовательное перемещение от одного фрагмента изображения (с окаймляющим прямоугольником I^1) к другому (I^2 соответственно) с тем, что часть предыдущего фрагмента видна на мониторе и после перемещения (т.е. фактически $I^{intersect} = I^1 \cap I^2 \neq \varnothing$). Причём если в режиме панорамирования площадь $I^{intersect}$ может быть очень мала (в случае, если пользователь интенсивно "перетаскивает" изображение), то в режиме фиксированного смещения площадь $I^{intersect}$ гарантированно составляет 50-75% от площади исходного I^1.

В связи с этим возникает идея дальнейшего сужения множества отображаемых объектов; в самом деле, если часть требуемого растрового изображения ($I^{intersect}$) уже была сгенерирована при предыдущем запросе пользователя – нет необходимости генерировать её заново; достаточно лишь растеризовать *новый* фрагмент ($I^{new} = I^2 / I^{intersect}$, $I^{new} \subset I^2$, см. также рис. 2).

5.2 Параллельная модификация алгоритма.

В связи с широким распространением ПК на базе многоядерных процессоров особенный интерес представляют параллельные модификации алгоритмов визуализации.

В ходе анализа и апробации различных способов декомпозиции алгоритмов индексирования и растеризации, автором была реализована схема функциональной декомпозиции (конвейерного типа), в результате чего удалось снизить время отображения на 20-25%.

Идея предложенного способа декомпозиции заключается в том, что в то время, как i -й слой (установочный объект) индексируется в первом потоке, – второй поток может выполнять преобразование координат для i-1 -го слоя, а третий поток – растеризацию i-2 -го слоя. Графически конвейерная схема проиллюстрирована на рис. 3. В роли общих данных для трёх потоков выступает индекс (взаимодействие 1 и 2 потоков) и дерево внутреннего формата (взаимодействие 2 и 3 потоков).

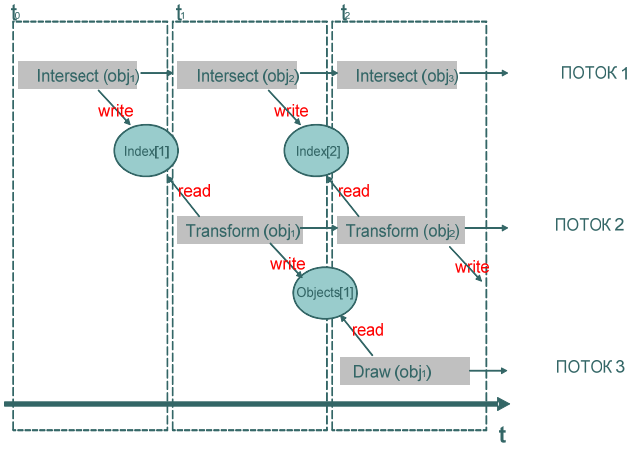

Рис. 3. Графическая иллюстрация алгоритма функциональной "конвейерной" декомпозиции (Intersect – подпрограмма индексирования, Transform – подпрограмма преобразования координат, Draw – подпрограмма растеризации).

Очевидным недостатком предложенной схемы является слабая масштабируемость алгоритма. В тоже время, в отличие от различных вариантов декомпозиции по данным, “конвейерный” алгоритм эффективно функционирует независимо от целевой аппаратной платформы и возможностей графического API.

В завершение упомянем, что на основе анализа узких мест программы были применены различные методы низкоуровневой оптимизации: использование внутреннего менеджера памяти; кэширование идентификаторов команд; избирательное применение целочисленной арифметики в процессе изменении масштаба карты. Каждый из этих методов оказался вполне адекватным поставленной задаче, обеспечивая прирост производительности на той или иной стадии обработки графической информации.

6. ЗАКЛЮЧЕНИЕ.

В ходе внедрения алгоритмов оптимизации, было выполнено несколько вычислительных экспериментов, позволивших проанализировать как вклад каждого алгоритма по отдельности, так и совокупный вклад всех рассмотренных алгоритмов оптимизации.

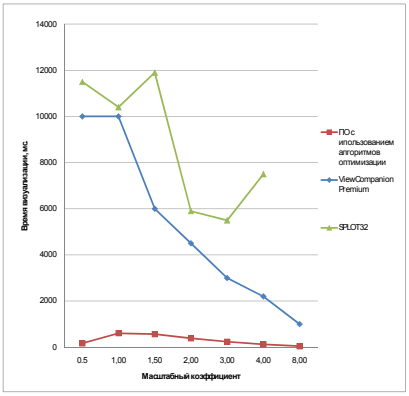

Рис. 4. Время отклика визуализатора в процессе интерактивной навигации (математическое ожидание времени отклика в с.).[1]

В роли экспериментальных данных использовались типовые топографические цифровые и морские навигационные карты в формате HP-GL/2 (количество объектов координатной основы в пределах 1-2 млн. единиц на карту).

Показатели, продемонстрированные ПО, разработанным с использованием рассмотренных алгоритмов [5], оказались значительно выше показателей сторонних коммерческих визуализаторов HP-GL/2 [6], [8] (рис. 4).

При этом по результатам сравнительного анализа рассмотренных в статье алгоритмов были выявлены следующие закономерности:

- Индексирование с использованием грубого пространственного критерия (окаймляющих прямоугольников) позволяет существенно ускорить визуализацию при коэффициенте масштабирования более 2 за счёт быстрого отсечения протяжённых объектов.
- В свою очередь, алгоритм генерализации и иерархической индексации обеспечивают прирост производительности при малых коэффициентах масштабирования (одно- – трёх- кратное увеличение листа ЦТК), в частности за счёт исключения небольших пространственных объектов.
- При использовании всей совокупности алгоритмов индексирования и генерализации – время, затрачиваемое на визуализацию фрагмента, монотонно убывает с ростом коэффициента увеличения.
- Применение внутреннего формата позволяет существенно повысить производительность визуализации на этапе интерактивной навигации по листу ЦТК (время визуализации уменьшается до 10 раз). В тоже время внедрение внутреннего формата ведёт к увеличению времени первоначальной загрузки исходного документа и требует большего объёма доступной оперативной памяти (порядка 80-100Мб для типовых ЦТК в формате HP-GL/2).
- Функциональная декомпозиция визуализации позволяет дополнительно повысить эффективность алгоритма визуализации на 10% – 25% практически при всех масштабных коэффициентах. Однако при 8-кратном увеличении полученный выигрыш начинает уравновешиваться накладными расходами, связанными с организацией многопоточных вычислений.

В целом, применение изложенных подходов позволило достичь высокой степени эффективности при интерактивном просмотре ЦТК.

7. ЛИТЕРАТУРА

[1] Гамма, Э. Приемы объектно-ориентированного проектирования: Паттерны проектирования [Текст] : пер. с англ./ Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. – СПб : Питер, 2004. – 366 с.

[2] Ласло, М. Вычислительная геометрия и компьютерная графика на C++ [Текст] / М. Ласло. – М. : БИНОМ, 1997. – 304 с.

[3] Фень, Ю. Программирование графики для Windows [Текст] / Ю. Фень. – СПб. : Питер, 2002. – 1072 с.

[4] Guting, R.H. An introduction to spatial database systems [Text] / R.H. Gutting. – VLDB Journal. – 1994. – Vol 3. – No 4. – P. 357-399.

[5] Matveev, Z. Indices Organization as Visualization System Performance Improvements Means [Text] / Z. Matveev // 9-th International Conference on Pattern Recognition and Image Analysis: New Information Technologies PRIA-9-2008 : conference proceedings. – Nizhni Novgorod, 2008. – vol.2. – P. 15-17.

[6] Novy A. SPlot V5.20 for Windows. User Guide [Electronic resource] / A. Novy. – Режим доступа: http://www.swplot.com , свободный – Загл. с экрана.

[7] The HP-GL/2 and HP RTL reference guide: a handbook for program developers [Text] / Hewlett-Packard. – MA : Addison-Wesley, 1997. – 3rd edition. – 544 p.

[8] View Companion. Premium Edition. Quick reference [Electronic resource] / Software Companions. – Режим доступа:

[1] Использовалась тестовая конфигурация клиентского ПК на базе процессора Intel Core Duo, 1024 Mb SDRAM ОЗУ, оборудованного графическим видеоакселератором Intel GMA 950. В качестве тестовых данных была выбрана навигационная карта побережья Северного моря, размер файла ЦТК – 28 Мб.

http://www.softwarecompanions.com/viewpremium.html, свободный. – Загл. с экрана.

Об авторах

Кетков Юлий Лазаревич – д.т.н., профессор кафедры МО ЭВМ факультета ВМК Нижегородского государственного университета им. Н.И. Лобачевского (ННГУ), зав. отделом НИИ Прикладной Математики и Кибернетики при ННГУ.

E-mail: ket@unn.ru

Захар Александрович Матвеев – сотрудник нижегородского отделения Intel Corporation (разработчик программного обеспечения), выпускник аспирантуры ННГУ.

E-mail: zakhar.a.matveev@intel.com