УДК 004.032.24

МЕТОД МАРШРУТИЗАЦИИ С ПРЕПЯТСТВИЯМИ НА ОСНОВЕ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ

Заева К.А., Семенов А.Б.
Кафедра информационных технологий

Поступила в редакцию 29.03.2016, после переработки 15.05.2016.

В работе описан эффективный метод нахождения оптимального геодезического маршрута в плоской среде со стационарными полигональными препятствиями. Рассмотрен непрерывный вариант решения задачи с использованием графа видимости для полигональных объектов-препятствий. Решается задача векторизации и аппроксимации с контролируемой точностью бинарного изображения для получения полигональных областей. Применяется ускорение вычислений с использованием графических процессоров.

Ключевые слова: поиск минимального пути, среда с препятствиями, векторизация изображения, граф видимости, графические процессоры, технология CUDA.

Вестник ТвГУ. Серия: Прикладная математика. 2016. № 3. С. 85-95.

Введение

Задача поиска кратчайшего пути появилась довольно давно и включает в себя большое количество алгоритмов, методов и вариантов решения. Выбор методов решения данной задачи зависит от конкретной постановки, от области применения, инструментов, доступных для разработки. Поиск новых методов не теряет своей актуальности из-за постоянного развития приложений, которые требуют более быстрых и точных решений. Большинство уже существующих методов поиска пути предполагают, что пространство разбито на квадратные или шестиугольные ячейки. Для решения задачи непрерывное пространство сводится к нескольким дискретным вариантам, к которым уже применяются различные алгоритмы поиска маршрута [5]. Однако дискретное решение имеет множество недостатков, основным из которых является появление ошибок накопления в ходе вычислений. Кроме того, большинство существующих алгоритмов и методов решения задачи нахождения маршрута имеют чрезмерную вычислительную сложность, а также требуют точных алгебраических моделей препятствий.

В данной работе рассматривается задача в терминах непрерывной полигональной геометрии. Целью исследования является разработка метода маршрутизации, дающего результаты с высокой точностью за оптимально возможное время работы. Для ускорения времени работы алгоритмов в качестве вычислительного эксперимента используется параллельно-вычислительный подход, в частности графические процессоры.

1. Постановка задачи

Дана плоская среда с заданными на ней двумя точками – точка A (начало пути) и точка B (конец пути), и задан набор полигональных объектов, выступающих в роли препятствий. Требуется найти минимально возможный маршрут между двумя заданными точками (Рис. 1).

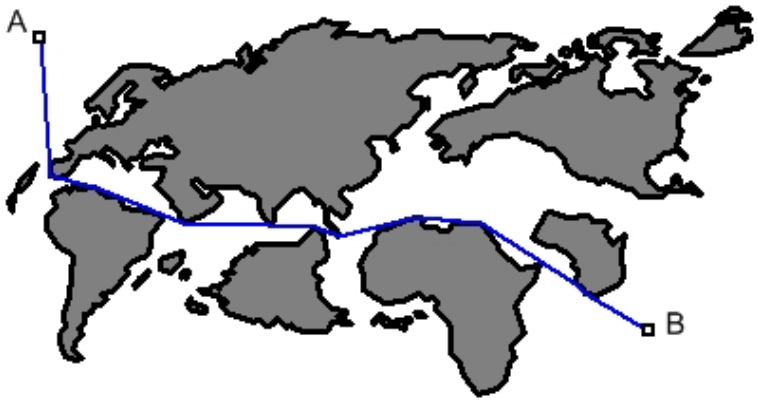

Рис. 1: Поиск минимального пути в среде с полигональными препятствиями

Рассматривается статичная двумерная плоскость, где набор полигональных объектов – это препятствия на местности, сквозь которые невозможно пройти. Данный набор представляет собой многоугольники без самопересечений, имеющие ненулевую площадь. Начальный и конечный пункты пути могут быть заданы произвольными точками на карте, которые не являются частью препятствий. Заметим, что искомый путь может проходить вдоль границы объектов, но не может их пересекать.

2. Обработка изображения. Получение объектов-препятствий

Исходные данные могут быть представлены в двух формах. Непрерывное (векторное) описание препятствий подразумевает, что полигоны заданы путем явного перечисления вершин с их координатами. Дискретное представление исходных данных описывает набор полигонов как матрицу черных точек на белом фоне. Таким образом, в качестве среды с препятствиями может быть использовано произвольное бинарное растровое изображение, которое подается алгоритму на вход. Для получения набора полигонов нам необходимо преобразовать контуры изображений из дискретного представления в непрерывное представление [8].

2.1 Векторизация изображения

На вход задачи подается бинарное изображение, в котором присутствуют только два цвета: 1 – белый, 0 – черный. На выходе имеем последовательность граничных точек. Вычисление границы растрового бинарного образа осуществляется

через направленный перебор точек с целочисленными координатами вдоль границы объекта [10].

Алгоритм оконтуривания применяется последовательно ко всем объектам – препятствиям входного изображения. На Рис. 2 представлен пример решения задачи векторизации, где слева – исходное бинарное растровое изображение, справа – полученный контур объектов. Для иллюстрации произведено прореживание граничных точек – крупным выделена каждая 30-я точка граничной последовательности.

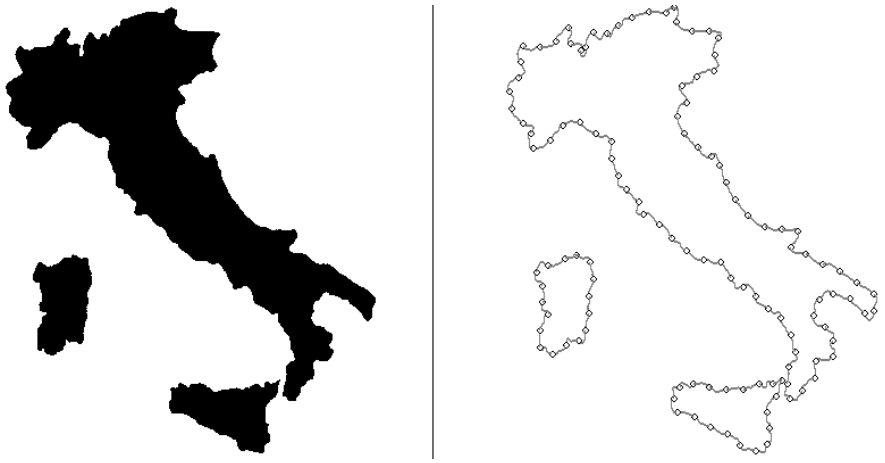

Рис. 2: Выделение контура объектов путем решения задачи векторизации

2.2 Аппроксимация границ объектов

Результатом выполнения алгоритма векторизации бинарного изображения является набор граничных точек. В том случае, когда на вход мы получаем изображение существенного размера, граничная последовательность точек имеет заведомо большую размерность описания (количество вершин). Применять какие-либо вычислительные операции к такому списку точек нерационально, поэтому для дальнейших действий необходимо уменьшить размерность описания набора граничных точек объектов путем решения задачи аппроксимации.

На вход алгоритма аппроксимации мы получаем пронумерованную последовательность точек. Для решения задачи будем использовать простой метод кусочно-линейной аппроксимации, названный итеративным подбором концевых точек [12].

На выходе алгоритма мы получаем набор связных отрезков, которыми аппроксимируется заданная фигура. Вершины этих отрезков и составляют искомое нами множество граничных точек, по набору которых в дальнейшем будут строиться полигональные препятствия. Количество получаемых отрезков (соответственно, вершин) прямо пропорционально зависит от выбранной точности аппроксимации. В качестве примера на Рис. 3 отображен результат аппроксимации граничной последовательности точек, полученной после векторизации изображения, с точностью в 5 пикселей. Изображение, полученное на вход, содержало 1786 граничных точек, в результате аппроксимации количество вершин сократилось до 76.

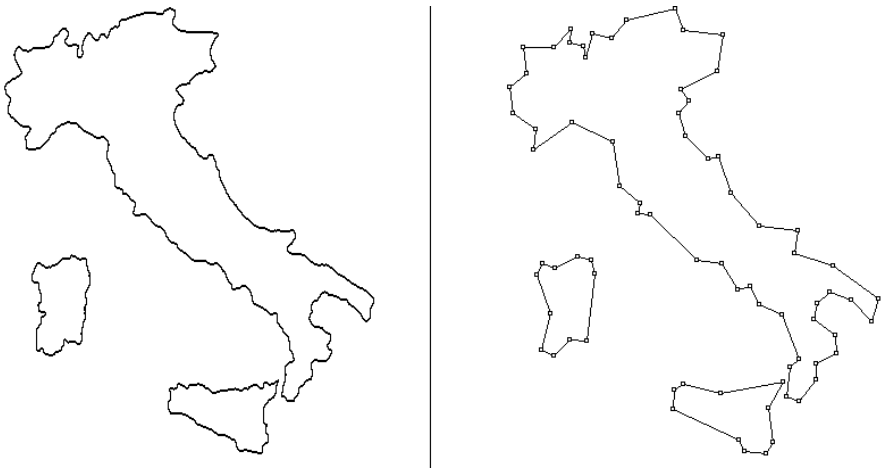

Рис. 3: Аппроксимация границ меньшим количеством точек

3. Построение графа видимости

Решение исходной задачи осуществляется путем предварительного построения графа видимости вершин с последующим применением алгоритма поиска пути на графе. Будем называть две точки взаимовидимыми, если отрезок, их соединяющий, не содержит в себе внутренних точек препятствий-многоугольников [1]. На Рис. 4 отрезками соединены все взаимовидимые вершины полигональных объектов. Заметим, что смежные вершины ребер полигонов всегда являются взаимовидимыми. Вершинами графа видимости являются вершины препятствий-многоугольников. Ребрами графа видимости являются отрезки, соединяющие взаимовидимые вершины графа.

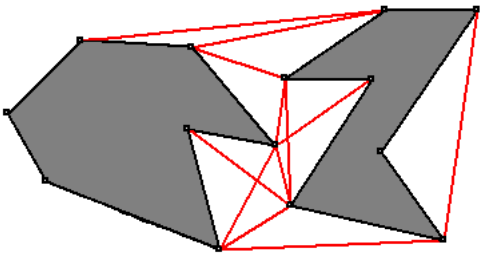

Рис. 4: Взаимовидимые вершины многоугольников

Для решения задачи поиска минимального пути между двумя точками граф видимости расширяется – добавляются начальная и конечная точки поиска пути к списку вершин, после чего отрезками соединяются взаимовидимые с ними вершины. По полученному графу ищется минимальный маршрут одним из стандартных алгоритмов поиска пути на графе [9]. Найденный кратчайший путь, проходящий

по графу видимости, и будет являться искомым минимальным маршрутом между двумя точками [2].

Существует достаточное количество алгоритмов построения графа видимости [3], далее рассмотрим самые известные из них.

Алгоритм Ли основан на сортировке ребер графа видимости по полярному углу. Проверка на пересечение с препятствиями-многоугольниками ведется по полученному отсортированному списку, что позволяет достигнуть времени работы построения графа $O(n^2 \log n)$, где n – общее количество вершин графа. Для реализации алгоритма используются АВЛ-деревья (сбалансированные по высоте двоичные деревья поиска).

Алгоритм Овермарса и Велзла базируется на концепции вращающихся деревьев (rotation trees [6]), что позволяет произвести вычисления за $O(n^2)$.

В алгоритме Гхоша и Маунта используется идея заметающей прямой с построением триангуляций Мельхорна [4]. Время работы такого подхода $O(|e| + n \log n)$, где $|e|$ – общее количество ребер графа видимости.

4. Оптимизация работы метода с использованием GPU

Основной целью данной работы является разработка эффективного метода маршрутизации. В связи с этим необходимо достигнуть минимально возможного времени работы алгоритма. Быстродействие вычислительного процесса может возрастать с увеличением числа параллельно работающих ядер. Таким образом, одним из способов ускорения вычислений алгоритмов является использование параллельно-вычислительного подхода. В рамках параллельно-вычислительного подхода могут применяться многоядерные системы, вычислительные кластеры, а также в качестве параллельной архитектуры можно использовать графические процессоры.

Многие ресурсоемкие вычислительные задачи достаточно хорошо «ложатся» на архитектуру GPU, позволяя заметно ускорить их численное решение. Основным условием пригодности задачи для реализации на GPU является наличие возможности распараллеливания алгоритма [7]. Сравнительный анализ показал, что описанные в предыдущем разделе методы построения графа видимости не могут быть реализованы с использованием параллельно-вычислительного подхода, так как в них задействованы структуры данных, которые априори не пригодны для параллельных процессов. В связи с этим использование данных алгоритмов на устройствах, обладающий параллелизмом, не представляется возможным. С другой стороны, существует наивный алгоритм [3], работа которого заключается в полном переборе вершин и ребер препятствий-многоугольников. Такой подход выполняется большим количеством итераций, что влечет за собой не совсем оптимальное время реализации, а именно $O(n^3)$, где n – общее количество вершин графа. Однако наивный алгоритм не использует никаких сложных структур для реализации и может быть весьма эффективно распараллелен при достаточном количестве обрабатывающих устройств. С точки зрения пользователя, приложение на GPU просто будет работать значительно быстрее, выполняя те же функции, что и раньше. Для решения подобного вида задач могут быть использованы программно-аппаратные решения на основе технологии CUDA компании NVIDIA.

4.1 Технология CUDA

В рамках технологии CUDA графический процессор выступает в роли массивно-параллельного сопроцессора к центральному процессору. CUDA строится на концепции, согласно которой GPU, называемый устройством, выступает в роли массивно-параллельного сопроцессора к CPU. Таким образом, программа задействует как CPU, так и GPU – обычный (непараллельный) код выполняется на CPU, а код для параллельных вычислений – на GPU, как набор одновременно выполняющихся потоков.

Технология CUDA реализует SIMD-архитектуру [11](Single Instruction Multiple Data), все параллельные процессоры одного типа одновременно способны выполнять одну и ту же операцию над многими данными. SIMD-процессор получает на вход поток данных и параллельно их обрабатывает, порождая на выходе результирующий поток данных. Обработка элементов осуществляется так называемым ядром (kernel). Количество CUDA-ядер зависит от технических характеристик графического процессора, установленного на вычислительной машине. Особенностью архитектуры CUDA является блочно-сеточная организация, необычная для многопоточных приложений (все нити, выполняющие ядро, объединяются в блоки, а блоки, в свою очередь, объединяются в сетку), при этом драйвер CUDA самостоятельно распределяет ресурсы устройства между нитями. Преимуществом такой организации данных является то, что исходная задача разбивается на наборы отдельных подзадач, решаемых независимо друг от друга, и помещается в свой блок нитей. Ключевой момент архитектуры CUDA – легкая масштабируемость. Единожды написанный код будет запускаться на всех устройствах, поддерживающих данную технологию.

4.2 Анализ и сравнение результатов

В рамках данной задачи были проведены вычислительные эксперименты работы алгоритма вычисления матрицы видимости для вершин препятствий-многоугольников на различных устройствах CPU и GPU с применением описанной выше технологии CUDA.

Устройства, на которых проводились исследования, обладают следующими техническими характеристиками:

1. CPU: Intel(R) Core(TM)2 Duo CPU E7200 @ 2.53GHz, GPU: GeForce 9600 GT (64 CUDA ядра);
2. CPU: Intel(R) Core(TM) i5-2500K CPU @ 3.30GHz, GPU: GeForce GTX 560 Ti (384 CUDA ядер).

В Таблице 1 приведены результаты вычисления матрицы видимости, где для расчетов задействован либо центральный процессор, либо графический процессор, соответствующего устройства. Мы видим, что применение параллельно-вычислительного подхода позволяет увеличить производительность почти в 10 раз. Данный параллельный алгоритм хорошо масштабируем. Усовершенствовав графический процессор на вычислительном устройстве, можно получить крайне высокие результаты времени работы алгоритма – в разы сократить время вычислений для решения задачи.

Таблица 1: Время вычисления матрицы видимости (в секундах)

Устройство	Количество точек			
	159	**1108**	**1939**	**4366**
CPU.1	0.04	10.4	37.56	344.19
CPU.2	0.02	4.56	17.53	164.67
GPU.1	0.016	3.58	17.53	187.68
GPU.2	0.003	0.45	2.05	19.42

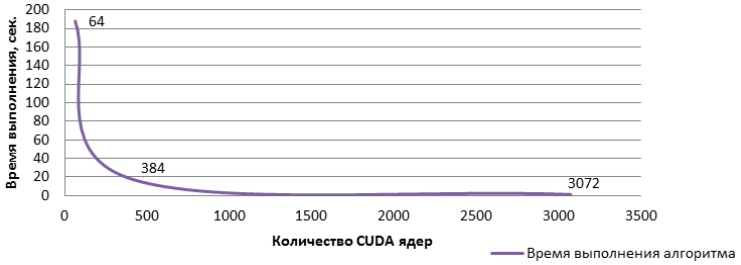

Рис. 5: Зависимость времени вычисления от мощности исполняемого графического процессора

Основной характеристикой, влияющей на быстродействие параллельно-вычисляемого CUDA-приложения, является количество ядер CUDA в устройстве GPU. На тех процессорах, где проводились тестирования алгоритма, количество ядер, соответственно, равно 64 и 384. Применяя метод экстраполяции к имеющимся реальным данным, можно предположить, как будет зависеть время выполнения программы с увеличением CUDA ядер, имеющихся на графическом процессоре. Допустим, если мы возьмем новейший графический процессор GeForce GTX Titan X, имеющий 3072 CUDA ядер, то можно предположить, что время работы алгоритма вычисления матрицы видимости для решения нашей задачи на таком устройстве будет крайне малым, расчеты будут производиться мгновенно (Рис. 5).

Заключение

В ходе работы был создан метод маршрутизации в среде с полигональными препятствиями, который может работать как с вручную заданными двумерными средами с объектами-многоугольниками, так и с произвольными бинарными растровыми изображениями в качестве «карты». Во втором случае «карта» преобразуется к набору вершин полигонов путем векторизации объектов и последующей аппроксимации граничных точек. Минимальный маршрут между точками вычисляется с использованием графа видимости полигональных объектов-препятствий. На выход мы получаем результат нахождения кратчайшего пути с высокой точностью вычисления. Разработанный прототип программы имеет возможность активно использовать ресурсы графического процессора. Применяемый параллельно-вычислительный подход позволяет рассчитывать на существенное увеличение про-

изводительности (скорости работы) на современных и будущих аппаратных платформах.

Список литературы

[1] Asano T., Asano T., Guibas L.J., Hershberger J., Imai H. Visibility of disjoint polygons // Algorithmica. 1986. Vol. 1, № 1. Pp. 49–63.

[2] Berg M., Cheong O., Kreveld M., Overmars M. Computational Geometry: Algorithms and Applications. Third Edition. Berlin: Springer-Verlag, 2008.

[3] Kitzinger J. The Visibility Graph Among Polygonal Obstacles: A Comparison of Algorithms. University of New Mexico, 2003. Pp. 6–22.

[4] Mehlhorn K. Data Structures and Algorithms. Searching and Computational Geometry. Vol. 3. Berlin: Springer-Verlag, 1984.

[5] Stout B. (оригинальная статья), Каменский М. (перевод) Алгоритмы поиска пути. [Электронный ресурс]. 2000. URL: http://pmg.org.ru/ai/stout.htm (дата обращения: 15.01.2016).

[6] Wikipedia contributors. Tree rotation [Electronic resource] // The Free Encyclopedia. 2015. URL: http://en.wikipedia.org/wiki/Treerotation (accessed at 16.03.2016).

[7] Боресков А.В., Харламов А.А. Основы работы с технологией CUDA. М.: ДМК Пресс, 2010.

[8] Заева К.А., Семенов А.Б. Система поиска минимального пути в среде с полигональными препятствиями // Труды 24-й международной конференции ГРАФИКОН-2014. Ростов-на-Дону, Южный федеральный университет, 2014. С. 163-166.

[9] Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ. 3-е издание. М.: ООО «И.Д.Вильямс», 2013. 1328 с.

[10] Местецкий Л.М. Непрерывный скелет бинарного растрового изображения // Труды 8-й международной конференции ГРАФИКОН'98. Москва, 1998.

[11] Орлов С.А., Цилькер Б.Я. Организация ЭВМ и систем. Учебник для вузов. 2-е издание. СПб.: Питер, 2011. С. 526-556.

[12] Семенов А.Б. Новый подход в конструировании рукописных шрифтов // Труды 18 международной конференции ГРАФИКОН-2008. Москва: МГУ, 2008.

Библиографическая ссылка

Заева К.А., Семенов А.Б. Метод маршрутизации с препятствиями на основе параллельных вычислений // Вестник ТвГУ. Серия: Прикладная математика. 2016. № 3. С. 85-95.

Сведения об авторах

1. **Заева Камила Андреевна**
магистрант кафедры информационных технологий Тверского госуниверситета.

Россия, 170100, г. Тверь, ул. Желябова, 33, ТвГУ, факультет ПМиК.
E-mail: kapitanka.ki@gmail.com.

2. **Семенов Андрей Борисович**
доцент кафедры информационных технологий Тверского госуниверситета.

Россия, 170100, г. Тверь, ул. Желябова, д. 33, ТвГУ, факультет ПМиК.
E-mail: semenov@tversu.ru.

METHOD OF ROUTING WITH OBSTACLES BASED ON PARALLEL COMPUTING

Zaeva Kamila Andreevna
Master student at Information Technologies department, Tver State University
Russia, 170100, Tver, 33 Zhelyabova str. E-mail: kapitanka.ki@gmail.com.

Semenov Andrey Borisovich
Associate professor at Information Technologies department, Tver State University
Russia, 170100, Tver, 33 Zhelyabova str.
E-mail: semenov@tversu.ru

Received 29.03.2016, revised 15.05.2016.

We propose a new effective method of finding the minimal geodesic path in a 2D environment with polygonal obstacles. We describe continuous solution that uses visibility graph for polygonal obstacles. The problem of binary image vectorization and approximation is solved with a controlled accuracy. GPU computation is used to speed up the calculations.

Keywords: pathfinding, environment with obstacles, vectorization of images, visibility graph, GPU, technology CUDA.

Bibliographic citation

Zaeva K.A., Semenov A.B. Method of routing with obstacles based on parallel computing. *Vestnik TvGU. Seriya: Prikladnaya Matematika* [Herald of Tver State University. Series: Applied Mathematics], 2016, no. 3, pp. 85-95. (in Russian)

References

[1] Asano T., Asano T., Guibas L.J., Hershberger J., Imai H. Visibility of disjoint polygons. *Algorithmica*, 1986, vol. 1 (1), pp. 49–63.

[2] Berg M., Cheong O., Kreveld M., Overmars M. *Computational Geometry: Algorithms and Applications*. Third Edition. Berlin: Springer-Verlag, 2008.

[3] Kitzinger J. *The Visibility Graph Among Polygonal Obstacles: A Comparison of Algorithms*. University of New Mexico, 2003. Pp. 6–22.

[4] Mehlhorn K. *Data Structures and Algorithms. Searching and Computational Geometry*. Vol. 3. Berlin: Springer-Verlag, 1984.

[5] Stout B. *Smart Move: Intelligent Path-Finding* [Electronic resource]. URL: http://www.gamasutra.com/view/feature/131724/smart_move_intelligent_.php (accessed at 16.03.2016).

[6] Wikipedia contributors. Tree rotation [Electronic resource]. *The Free Encyclopedia*. URL: http://en.wikipedia.org/wiki/Treerotation (accessed at 16.03.2016).

[7] Boreskov A.V., Kharlamov A.A. *Osnovy Raboty s Tekhnologiei CUDA* [Fundamentals of CUDA Technology]. DMK Press Publ., Moscow, 2010. (in Russian)

[8] Zaeva K.A., Semenov A.B. Search engine of the minimal path in an environment with polygonal obstacles. *Trudy 24-i Mezhdunarodnoi Konferentsii GRAFIKON-2014* [Proceedings of the 24th International Conference GraphiCon 2014]. Rostov-na-Donu, Yuzhnyi federal'nyi universitet, 2014. Pp. 163–166. (in Russian)

[9] Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. *Introduction to Algorithms*, 3rd Edition. The MIT Press, Cambridge, Massachusetts, 2009. 1313 p.

[10] Mestetskii L.M. Continuous skeleton of a binary raster image. *Trudy 8-i Mezhdunarodnoi Konferentsii GRAFIKON'98* [Proceedings of the 8th International Conference GraphiCon 98]. Moscow State University, Moscow, 1998. (in Russian)

[11] Orlov S.A., Tsil'ker B.Ya. *Organizatsiya EVM i Sistem. Uchebnik Dlya Vuzov* [Organization of computers and systems. Textbook for high schools]. Piter Publ., Saint-Petersburg, 2011. Pp. 526–556. (in Russian)

[12] Semenov A.B. A new approach in the design of script fonts. *Trudy 18 Mezhdunarodnoi Konferentsii GRAFIKON-2008* [Proceedings of the 18th International Conference GraphiCon 2008]. Moscow State University, Moscow, 2008. (in Russian)