УДК 681.5.14

МЕТОД БИНАРНОГО ПОИСКА ЭЛЕМЕНТОВ ИЗОБРАЖЕНИЯ ФУНКЦИОНАЛЬНО ЗАДАННЫХ ОБЪЕКТОВ С ПРИМЕНЕНИЕМ ГРАФИЧЕСКИХ АКСЕЛЕРАТОРОВ

С. И. Вяткин

Институт автоматики и электрометрии СО РАН,
630090, г. Новосибирск, просп. Академика Коптюга, 1
E-mail: sivser@mail.ru

Рассмотрены проблема синтеза изображений высокого качества в реальном времени, способ задания свободных форм без аппроксимации их полигонами или патчами, вопросы использования функций возмущения для анимации поверхностей трёхмерных объектов в реальном времени. Предложены метод визуализации функционально заданных объектов, адаптированный для графических ускорителей; реализация методов преобразования описывающей функции для геометрических операций: деформации, смещения и метаморфозиса, или морфинга, в том числе и негомеоморфных объектов. Показаны преимущества метода задания поверхностей перед существующими (алгебраическими и Безье и методами их визуализации).

Ключевые слова: геометрические объекты, функции возмущения, геометрические операции, бинарное деление объектного пространства.

Введение. На современном этапе развития компьютерной графики за счёт совершенствования элементной базы и технологии изготовления видеокарт, увеличения количества процессорных блоков и их тактовой частоты достигнута графическая производительность, достаточная для большинства применений. В то же время основные подходы к созданию фотореалистичных изображений остаются неизменными и не удовлетворяют требованиям многих областей применения трёхмерной графики, что приводит к противоречию, которое нельзя решить без разработки нового теоретического базиса.

При формировании трёхмерных сцен наиболее часто используется полигональное задание моделей объектов, которое имеет ряд ограничений. Каркасные модели трёхмерных объектов являются приближёнными. Повышение реалистичности графических сцен предусматривает увеличение уровня детализации для корректной аппроксимации поверхностей объектов реального мира, причём темпы роста геометрической сложности трёхмерных изображений превышают темпы роста производительности графических средств. Для достижения фотореализма необходимо свыше миллиона полигонов в сцене, при этом наблюдается тенденция к дальнейшему увеличению детализации. Уже сегодня во многих приложениях количество треугольников сцены сопоставимо или превышает число пикселов разрешения экрана, что нивелирует преимущества полигонального подхода. При возрастании сложности сцены эффективность полигонального метода экспоненциально падает. Структура полигональных сеток линейна и они не обеспечивают поддержки многомасштабности, поэтому работа с большими сетками затруднена и необходимы вычислительно-сложные методы упрощения. Динамическое управление детализацией потребляет значительные вычислительные ресурсы, требует непрерывного пересчёта не только координат вершин треугольников, но и параметров освещённости. При частом переключении между уровнями наблюдается эффект «волнистости» поверхности, что не свойственно реальным объектам. В полигональные модели сложно внести атрибутивную информацию, а алгоритмы визуализации выполнения топологических операций достаточно трудоёмки. При аппроксимации

треугольниками существуют проблемы высокой глубинной сложности, отбраковки невидимых граней, определения и смены уровней детальности, клипирования треугольников пирамидой видимости и другие. Объёмы данных при визуализации трёхмерных объектов со сложной поверхностью приближаются к воксельным моделям. Полигональная модель принципиально не позволяет получить многие визуальные эффекты, необходимые для реалистичного отображения сцены. С помощью скелетной анимации нельзя сделать качественную анимацию гибких материалов, а также выполнить трёхмерный метаморфозис, или морфинг, объектов.

От этих недостатков можно избавиться, применяя аналитическое задание объектов и растеризацию их с помощью алгоритмов трассировки лучей. Аналитическое задание геометрических объектов не требует большого объёма памяти. Существуют работы по визуализации функционально заданных поверхностей, таких как F-rep [1], поверхностей свёртки [2–4], неявно заданных поверхностей, известных в компьютерной графике как капельные модели [5, 6], метасфер [7], мягких объектов [8] и т. д. Однако их применение ограничено довольно узким классом моделируемых поверхностей и медленной визуализацией. Используемые алгоритмы трудно оптимизировать для визуализации в реальном времени, что также накладывает ограничения на их практическое применение.

Один из главных недостатков известных методов визуализации — сложность вычисления точек поверхности. Так, метод маршировки по лучу не гарантирует обнаружения поверхности, кроме того, он — медленный [9, 10]. Метод определения пересечения луча с поверхностью, заданной в неявном виде, является сложным для вычисления L- и G-параметров [11]. В методе трейсинга нахождение наибольшего радиуса, когда ни одна точка объёма не лежит внутри сферы, — нетривиальная задача [12]. При трассировке луча с анализом интервала для сложных функций необходимы индивидуальные вычисления для каждого луча и каждого интервала вдоль этого луча [13]. При быстрой трассировке поиск лучей, пересекающих поверхность, требует много вычислений и недостаточно эффективен, так как способы кластеризации этого метода не решают данную проблему полностью [14].

В работе [15] описан метод отслеживания лучей для визуализации поверхностей, заданных алгебраически полиномами высокой степени. Однако моделировать реальные объекты с помощью полиномов непросто. Также не гарантируется точность приближения начальной функции к кривой Безье. Ещё одним недостатком этого метода является то, что перевод объекта в другую систему координат — сложная задача. Поэтому создание динамических сцен проблематично.

Известен ещё один метод визуализации аналитически заданных объектов с применением графических ускорителей (GPU) [16], основанный на обычном пошаговом отслеживании лучей. Особенностью является то, что размер шага не постоянен, а выбирается на каждой итерации, где определяется радиус сферы с центром в текущей точке на луче. Недостаток метода заключается в том, что нахождение подходящего радиуса — сложная задача. Для статичных сцен авторы алгоритма предварительно обрабатывали данные. Поэтому, как и в предыдущем методе, визуализация объектов, которые изменяют свою форму и положение во времени, требует значительных вычислительных затрат.

Цель данной работы — создание метода визуализации функционально заданных объектов на основе функций возмущения с применением графических акселераторов.

Геометрические объекты. Для описания сложных геометрических объектов используются функции отклонения (второго порядка) от основной квадрики [17]. Свободные формы строятся с помощью квадрик и представляются композицией базовой квадрики и возмущений:

$$F'(x, y, z) = F(x, y, z) + \sum_{i=1}^{N} f_i R_i(x, y, z), \tag{1}$$

где f_i — формфактор; $R(x, y, z)$ — возмущение:

$$R_i(x, y, z) = \begin{cases} Q_i^3(x, y, z), & \text{если } Q_i(x, y, z) \geq 0, \\ 0, & \text{если } Q_i(x, y, z) < 0, \end{cases} \tag{2}$$

где $Q(x, y, z)$ — возмущающая квадрика.

Для того чтобы поверхность была гладкой, степень должна быть больше двух (2). Это условие гарантирует непрерывность функции и её производной.

Геометрические операции. Геометрическая модель создаёт условия для конструирования объектов и их композиций различной сложности, используя множество геометрических операций Φ, определяемое математически следующим образом [1]:

$$\Phi_j\colon M^1 + M^2 + \ldots + M^n \to M. \tag{3}$$

Для формирования моделей сложных объектов на базе функций возмущения проводятся такие операции, как проекции, смещение, метаморфозис, кручение и заметание движущимся твёрдым телом [18], осуществляемые с применением булевых операций объединения и пересечения. Бинарная операция ($n = 2$) (3) объектов G_1 и G_2 означает операцию $G_3 = \Phi_j(G_1, G_2)$ с определением

$$f_3 = \psi(f_1(\mathbf{X}), f_2(\mathbf{X})) \geq 0, \tag{4}$$

где ψ — непрерывная вещественная функция двух переменных. С помощью операции смещения можно создавать увеличенную или уменьшенную копию исходного объекта, т. е. делать положительное или отрицательное смещение соответственно. При метаморфозисе, или морфинге, происходит плавный переход одного объекта в другой. При задании объектов с применением функций возмущения осуществляются трёхмерный морфинг негомеоморфных объектов (например, тора и сферы) и морфинг с ограничениями без разрыва поверхности и последующего её «склеивания». Два тела гомеоморфны, если из исходного тела путём взаимно однозначного непрерывного преобразования описывающей его функции (с помощью морфинга) получается второе (т. е. для негомеоморфных объектов мы не можем простым приближением квадрик вычислить неразрывный морфинг). В процессе морфинга могут участвовать не два исходных объекта, а четыре. В таком случае схема интерполяции будет билинейной, а геометрическая операция — кватернарной ($n = 4$) (3). Кручение — это деформация тела, являющаяся частным случаем биективного отображения, которое служит для выявления деформаций исходных объектов. Заметание есть проекция движущегося тела из $4D(x, y, z, t)$- в $3D(x, y, z)$-пространство. Эти операции не изменяют степени функций заданных моделей. Одним из примеров отношений служит определение столкновений между объектами [19]. Бинарное отношение является множеством $M^2 = M \times M$. Оно выражается как

$$S_j\colon M \times M \to I. \tag{5}$$

С помощью особого теста на пересечение и бинарного поиска можно за постоянное число шагов (с заданной точностью) найти точку столкновения объектов, если таковое происходит. В целях расчёта времени обнаружения столкновений тестировались объекты, различавшиеся как по степени сложности (форме), так и по виду столкновения (различными сторонами и частями объектов).

Визуализация. Главным на этом этапе является эффективное нахождение первого пересечения луча с поверхностью. Такая задача сходна с визуализацией в объёмной томографии, где функция плотности задаётся в виде дискретных данных. А в нашем случае используется аналитически определённая функция плотности, что позволяет более эффективно осуществлять поиск точек поверхности.

Вычисление пересечения лучей с поверхностями трёхмерных объектов предлагается проводить методом, в котором отсутствуют вышеперечисленные недостатки, характерные для рассмотренных ранее известных методов аналитического задания поверхностей.

Для простоты понимания будем считать, что сцена находится в единичном трёхмерном кубе. Перспектива не анализируется ввиду того, что она сводится к переходу в другую систему координат. Поэтому опустим начальные преобразования и уделим больше внимания основной части метода. Примем, что наблюдатель смотрит вдоль оси Z. Необходимо получить проекцию сцены на плоскость XY. Проекция должна представлять собой конечный набор значений. Через плоскость XY куба проходят лучи, и каждому из них соответствует пиксел на изображении. Лучи ограничены передней и задней гранями куба. В процессе поиска точки пересечения луча и поверхности каждый луч делится вдоль оси Z, образуя набор вокселей. Таким образом, получим функцию плотности вдоль луча, которая зависит от одной переменной. Задача будет состоять в нахождении первой точки, в которой функция обращается в нуль. Определив такую точку для каждого луча, можно вычислить координату Z. Далее в каждом пикселе задаётся нормаль. При наличии всех координат и нормалей в каждом пикселе используется модель локального освещения. В итоге получится изображение гладкого объекта с учётом освещения. На рис. 1 и 2 показаны алгоритмы бинарного деления квадрик и луча.

Уменьшение времени на визуализацию достигается за счёт эффективного использования вычислительных ресурсов графического акселератора с архитектурой CUDA (Compute Unified Device Architecture) от компании "NVIDIA" (США). Система CUDA — это модель параллельного программирования, которая позволяет реализовывать программы на языке C для исполнения на стандартном графическом акселераторе. Результат выполнения программы на любом из семейства акселераторов будет один и тот же даже несмотря на то, что у них может быть различное число потоковых мультипроцессоров. Большое количество вычислительных процессоров позволяет параллельно проверять пересечение с объектом одновременно нескольких лучей. В большей части графических

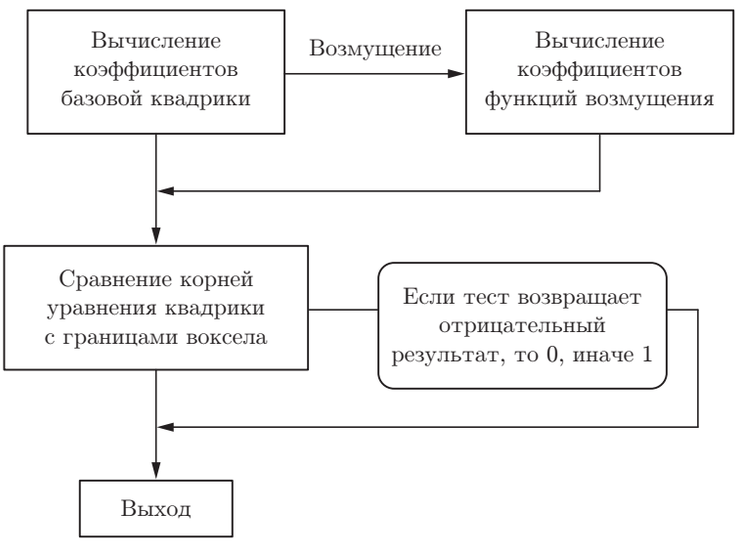

Рис. 1

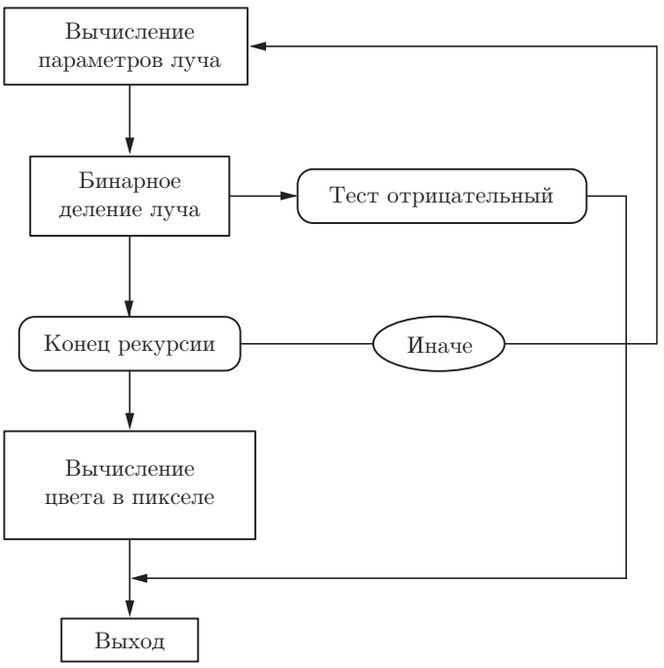

Рис. 2

акселераторов, поддерживающих CUDA, не менее 128 скалярных вычислительных ядер. Следовательно, будет параллельно вычисляться достаточно большая часть куба. Архитектура графических акселераторов основана на множестве потоковых мультипроцессоров, имеющих доступ к общей памяти для записи и чтения. Каждый из потоковых процессоров включает в себя восемь скалярных вычислительных ядер и набор внутрикристальной памяти четырёх типов. Количество регистров в зависимости от вычислительных возможностей акселератора может быть 8192 или 16384. Совместно используемая память составляет 16 Кбайт для каждого мультипроцессора. Кэш для константной памяти (доступно 8 Кбайт для каждого мультипроцессора) и кэш для текстурной памяти (доступно от 6 до 8 Кбайт для каждого мультипроцессора) применялись только для чтения. При реализации предлагаемого метода учитывалось влияние скорости работы процессоров с памятью

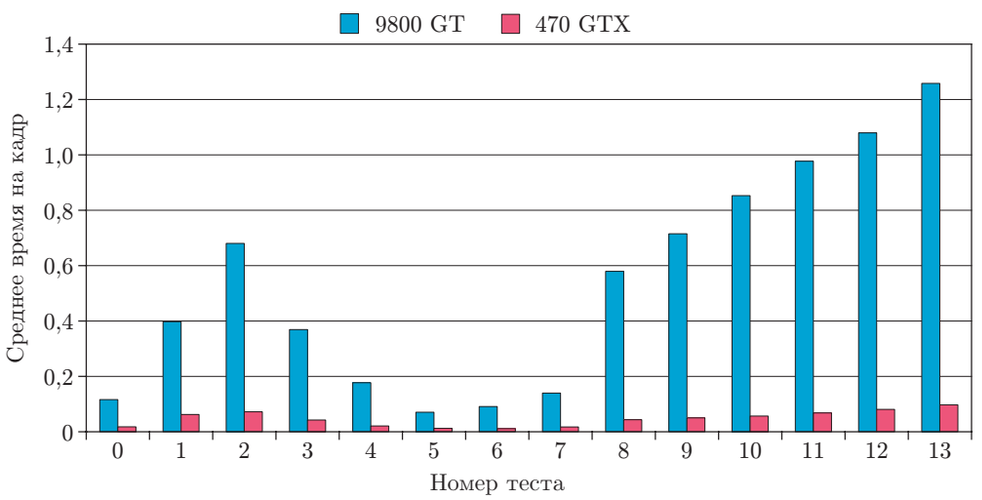

Рис. 3

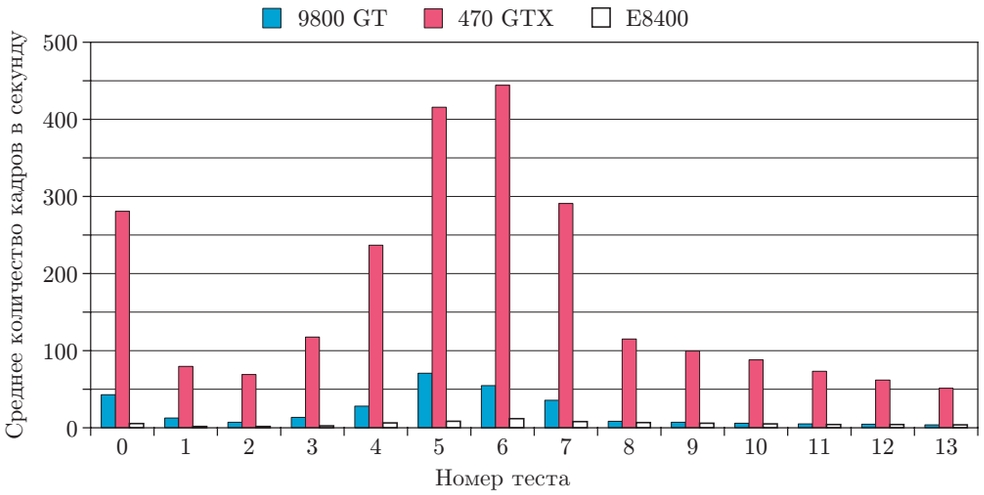

Рис. 4

на производительность. Максимально применялись регистры и совместно используемая память. Во всех остальных случаях задействовалась общая память графического акселератора.

В функцию графического акселератора входило вычисление координат точек поверхностей, нормалей и освещения. Геометрические преобразования выполнялись центральным процессором (CPU). Для визуализации использовался интерфейс прикладного программирования DirectX. Тестирование производилось на процессорах Intel Core2 CPU E8400 3.0 GHz, GPU 9800 GT и 470 GTX. На рис. 3, 4 показаны сравнительные результаты тестирования зависимости времени вычисления кадра от числа задаваемых функций возмущения для конкретного теста (см. таблицу).

Распределение числа задаваемых возмущений для объектов кадра в различных тестах

№ теста	Кол-во объектов в кадре	Типы возмущений*																		
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	1	4	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
1	22	4	4	4	1	1	4	1	1	3	1	1	3	1	1	3	1	1	3	1
2	22	4	4	4	1	1	6	1	1	3	1	1	3	1	1	3	1	1	3	1
3	3	1	1	3	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
4	3	1	1	3	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
5	10	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
6	1	2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
7	1	4	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
8	1	5	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
9	1	6	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
10	1	7	–	–	–	–	–	–	–	–	–	–		–	–	–	–	–	–	–
11	1	8	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
12	1	9	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
13	1	10	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

∗ — тесты для равного количества объектов в кадре и с равным числом функций возмущения для каждого объекта отличаются типом функции возмущения.

Заключение. Предложенные в данной работе способ задания трёхмерных объектов и метод визуализации имеют преимущества перед известными подходами. К основным достоинствам следует отнести: простоту вычисления точек поверхности с быстрым поиском и отбраковкой областей, не занятых объектами сцены; уменьшение количества поверхностей для описания криволинейных объектов в 100 и более раз; простоту анимации и деформации поверхностей.

Функциональное задание объектов особенно актуально в ряде задач компьютерной графики: при моделировании мягких или органических объектов, трёхмерном морфинге, определении столкновений объектов и конструктивной блочной геометрии. Области применения функционально заданных объектов — молекулярная биология, интерактивные графические системы визуализации, CAD-системы, системы $3D$-моделирования, трёхмерная веб-визуализация, системы прототипирования и т. д. [20].

СПИСОК ЛИТЕРАТУРЫ

1. **Pasko A., Adzhiev V., Sourin A. et al.** Function representation in geometric modeling: concepts, implementation and applications // The Visual Comput. 1995. **11**, N 6. P. 429–446.
2. **Bloomenthal J., Shoemake K.** Convolution surfaces // Comput. Graph. 1991. **25**, N 4. P. 251–256.
3. **Sealy G., Wyvill G.** Smoothing of three dimensional models by convolution // Proc. of the 1996 Conf. on Computer Graphics International. Washington: IEEE Computer Society, 1996. P. 184–190.
4. **McCormack J., Sherstyuk A.** Creating and rendering convolution surfaces // Comput. Graph. Forum. 1998. **17**, N 2. P. 113–120.
5. **Blinn J. F.** A generation of algebraic surface drawing // ACM Trans. Graph. 1982. **1**, N 3. P. 235–256.
6. **Muraki S.** Volumetric shape description of range data using "blobby model" // Comput. Graph. 1991. **25**, N 4. P. 227–235.
7. **Nishimura H., Hirai M., Kawai T. et al.** Object modeling by distribution function and a method of image generation // Trans. Institute of Electronics and Communication Engineers of Japan. 1985. **J68-D**, N 4. P. 718–725.
8. **Wyvill G., McPheeters C., Wyvill B.** Data structure for soft objects // The Visual Comput. 1986. **2**, N 4. P. 227–234.
9. **Tuy H., Tuy L.** Direct 2-D display of 3-D objects // IEEE Comput. Graph. and Appl. 1984. **4**, N 10. P. 29–33.
10. **Perlin K., Hoffert E. M.** Hypertexture // Comput. Graph. 1989. **23**, Is. 3. P. 253–262.
11. **Karla D., Barr A. H.** Guaranteed ray intersections with implicit surfaces // Comput. Graph. 1989. **23**, N 3. P. 297–306.
12. **Hart J. C.** Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces // The Visual Comput. 1994. **12**, N 10. P. 527–545.
13. **Mitchel D.** Robust ray intersection with interval arithmetic // Proc. on Graphics Interface. Toronto: Canadian Information Processing Society, 1990. P. 68–74.
14. **Sherstuyk A.** Fast ray tracing of implicit surfaces // Comput. Graph. Forum. 1999. **18**, N 2. P. 139–147.
15. **Reimers M., Seland J.** Ray casting algebraic surfaces using the frustum form // Comput. Graph. Forum. 2008. **27**, N 2. P. 361–370.
16. **Liktor G.** Ray tracing implicit surfaces on the GPU // Comput. Graph. and Geometry. 2008. **10**, N 3. P. 36–53.

17. **Вяткин С. И.** Моделирование сложных поверхностей с применением функций возмущения // Автометрия. 2007. **43**, № 3. С. 40–47.
18. **Вяткин С. И., Долговесов Б. С., Валетов А. Т.** Геометрические операции для функционально заданных объектов с применением функций возмущения // Автометрия. 2004. **40**, № 1. С. 65–73.
19. **Вяткин С. И., Долговесов Б. С., Корсун А. С.** Определение столкновений функционально заданных объектов в задачах компьютерной графики // Автометрия. 2003. **39**, № 6. С. 119–126.
20. **Вяткин С. И., Романюк А. Н.** Эффективные области применения функционально заданных объектов // Proc. of the Eighth Intern. Sci.-Pract. Conf. "Internet-Education-Science"(IES-2012). Vinnytsia, Ukraine, 1–5 October, 2012. С. 207–208.

Поступила в редакцию 14 мая 2014 г.