УДК 004.4 + 516.9

КАРТОГРАФИЧЕСКИЙ АЛГОРИТМ ЭФФЕКТИВНОЙ ОТБРАКОВКИ ВЕКТОРНЫХ ОБЪЕКТОВ

Алексейчук И.В., Вейцблит А.И.
Херсонский государственный университет

Представленный в статье алгоритм демонстрирует эффективную модель обработки, в частности визуализации, векторных объектов в ГИС.
Ключевые слова: алгоритм, эффективность, визуализация, векторный, ГИС

В последнее время пристальное внимание исследователей обращено на, казалось бы, давным-давно изученные и отточенные временем науки о географических объектах, геодезических и картографических методах. Новый виток в развитии этих дисциплин связан с естественной межотраслевой интеграцией наук и прогрессом информационных технологий: традиционно бумажная картография уступает свои позиции цифровым технологиям. Естественным является и то, что на данном этапе развития географических наук традиционные методы не отбрасываются, а усовершенствуются технологически.

Постановка задачи

В работе предлагается алгоритм для решения традиционной [1] по сути задачи об эффективной обработке, в частности, визуализации, векторных данных, имеющей важные практические приложения. Актуальность представленной задачи подтверждается огромной популярностью таких гео – информационных систем (ГИС), как *Google Maps*[2], *Meta.ua* [3], проект от *Yandex.ru* [4] и множества других. Потребность в алгоритмах такого рода существует также и в системах управления базами данных (СУБД). Подтверждением тому есть включение функций обработки пространственных данных и географических типов данных в ряд наиболее популярных пакетов, например, в MS SQL Server 2008 и PostgreSQL [5].

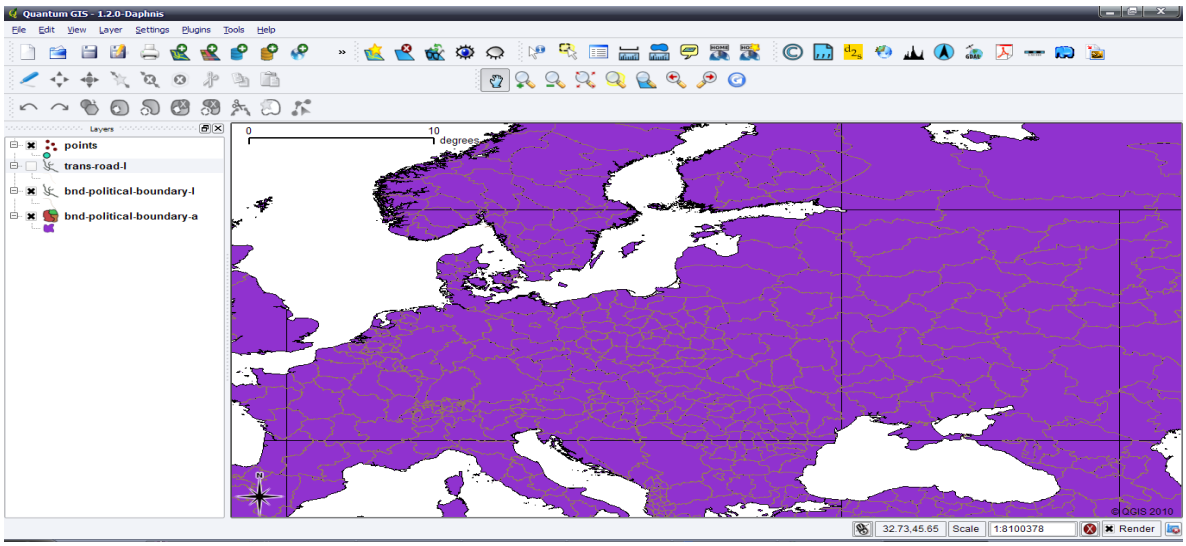

Рис.1. Quantum GIS отображает данные PostgreSQL сервера

Эта задача совершенно естественно появилась у одного из авторов: требовалось выбрать формат хранения и эффективный способ обработки карты для игрового приложения. Как известно, хранение карт в растровом виде имеет ряд недостатков, в том числе сложность сглаженного масштабирования и привязки точечных объектов в пространстве карты, а так как эти функции имели первоочередное значение, то от растра пришлось отказаться.

Альтернативный способ хранения геоданных – в векторной форме. Основным его недостатком является ограничение палитры объектов. Достоинствами же являются: малый объем, поддержка аффинных преобразований, аппаратная независимость и множество других. Необходимость самостоятельного определения формата хранения данных, а, как следствие, и алгоритма их обработки потребовало эффективных методов реализации программы.

Итак, требуется визуализировать информацию о геообъектах, представленных в векторной форме:

- *реке* (незамкнутая ломаная без самопересечений);
- *озере*/*море* (произвольный замкнутый полигон без самопересечения элементов контура);
- *границах государств и их территориальных единиц* (произвольный замкнутый полигон без самопересечения элементов контура).

На Рис.2 схематически представлена векторная карта:

- пространство представлено большой прямоугольной областью;
- линии – составные части контуров объектов;
- пунктирный прямоугольник – область, подлежащая визуализации;
- жирные линии – контуры объектов, пересекающих прямоугольник и тем самым подлежащих визуализации.

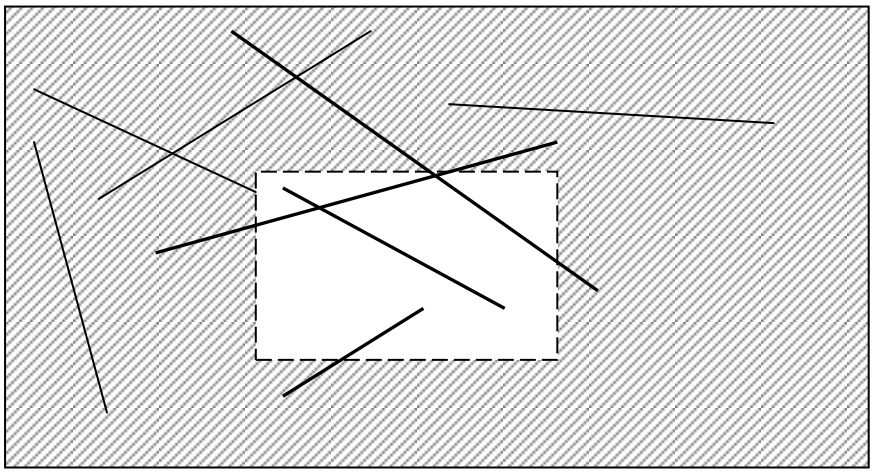

Рис.2. Макет векторной карты с областью просмотра

Должна быть обеспечена возможность отображения как всей карты, так и любой её части в любой момент в режиме реального времени. Метод решения такой задачи должен быть очень эффективным [6].

Обычно решение этой задачи основывается на алгоритме пространственной индексации R-деревьями [7]. R-дерево – древовидная структура данных, предложенная в 1984 году Антонином Гуттманом. Она используется для организации доступа к пространственным данным, то есть для индексации многомерной информации, такой, например, как географические данные с двумерными координатами (широтой и долготой). Типичным запросом с использованием R-деревьев мог бы быть такой: «Найти все музеи в пределах 2 километров от моего текущего местоположения» [8]. Схожей функциональностью пользуются физические движки для моделирования системы столкновений [9].

Эта структура данных разбивает пространство на множество иерархически вложенных и, возможно, пересекающихся, прямоугольников (для двумерного пространства). В случае трехмерного или многомерного пространства это будут прямоугольные параллелепипеды (кубоиды) или параллелотопы (рис. 3).

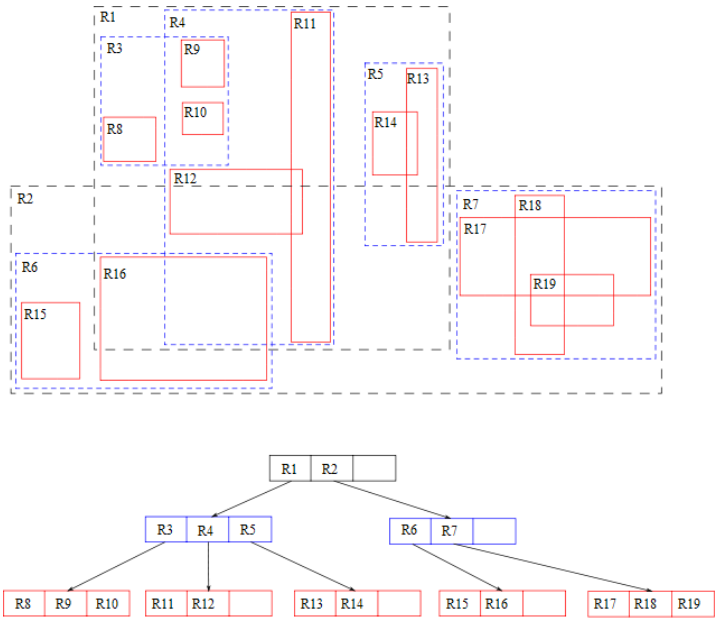

Рис. 3. Объекты в пространстве и соответствующее R-дерево

Мы предлагаем другой алгоритм, основанный на понятии дискретных координат пространственных переменных, который для некоторых конкретных задач оказывается более эффективным.

Дискретные координаты на плоскости.

Пусть $A_1(x_1; y_1)$, $A_2(x_2; y_1)$, $A_3(x_1; y_2)$, $A_4(x_2; y_2)$ (где $x_1 < x_2$, $y_1 < y_2$) – это вершины некоторого заданного прямоугольника П на координатной плоскости xOy со сторонами, параллельными осям координат: A_1A_2 , A_3A_4 || Ox, A_1A_3 , A_2A_4 || Oy.

Определение. Дискретная координата C_x точки C(x; y) на плоскости xOy по оси абсцисс равна $C_x = \begin{cases} 1, & если\ \ x > x_2 \\ 0, & если\ \ x_1 \le x \le x_2 \\ -1, & если\ \ x < x_1 \end{cases}$. Аналогично $C_y = \begin{cases} 1, & если\ \ y > y_2 \\ 0, & если\ \ y_1 \le y \le y_2 \\ -1, & если\ \ y < y_1 \end{cases}$, где C_y – это дискретная координата точки C(x; y) по оси ординат.

Дискретные координаты имеют простую геометрическую интерпретацию. А именно, 4 прямые $x = x_1$, $x = x_2$, $y = y_1$, $y = y_2$ делят координатную плоскость на 9 областей и в каждой из областей любая точка имеет дискретные координаты (C_x ; C_y), указанные далее.

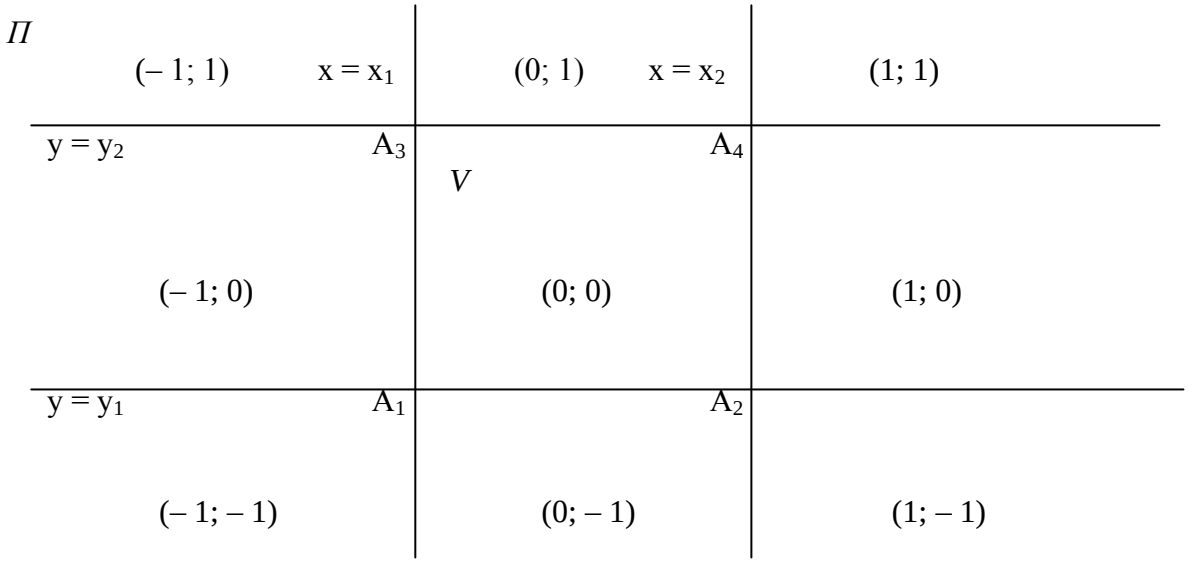

Рис. 4. Дискретные координаты на плоскости

Теорема. Пусть точки $B_1(\bar{x}_1;\bar{y}_1)$ и $B_2(\bar{x}_2;\bar{y}_2)$ – это концы заданного отрезка прямой. Тогда

1. Если дискретные координаты одной из этих точек равны (0; 0), то отрезок B_1B_2 имеет общие точки с прямоугольником П.

2. Если для этих точек совпадают обе дискретные координаты: $B_{1x} = B_{2x}$, $B_{1y} = B_{2y}$ и они отличны от (0; 0), то отрезок B_1B_2 не пересекает прямоугольник П.

3. Если для этих точек совпадает только одна дискретная координата и она отлична от 0, то отрезок B_1B_2 не пересекает прямоугольник.

4. Если для этих точек совпадает только одна дискретная координата и эта координата равна 0, то отрезок B_1B_2 пересекает прямоугольник П.

5. Пусть для точек B_1 и B_2 не совпадают обе дискретные координаты и для обеих точек они отличны от (0; 0). Положим $k = \dfrac{\bar{y}_2 - \bar{y}_1}{\bar{x}_2 - \bar{x}_1}$, $f(x,y) = y - \bar{y}_1 - k(x - \bar{x}_1)$. Рассмотрим значения $\operatorname{sign} f(x_i,y_j)$, где i, j = 1,2. Тогда

а) если все 4 значения совпадают, то отрезок и прямоугольник не пересекаются;

б) если имеются различные значения среди 4 – х, то отрезок пересекает П.

Доказательство. 1 очевидно, так как по условию один из концов отрезка принадлежит П. 2 следует из того, что обе крайние точки B_1 и B_2 по условию принадлежат выпуклой области, имеющей пустое пересечение с П.

3. Рассмотрим множество всех точек, одна из координат которой та же, что является общей для точек B_1 и B_2 . По условию это открытая полуплоскость, ограниченная одной из прямых, ограничивающих П, и имеющая с П пустое пересечение. Тогда и отрезок B_1B_2 содержится в этой полуплоскости.

4. По условию одна дискретная координата у B_1 и B_2 равна нулю, а вторые дискретные координаты отличны. В этом случае либо у одной из точек эта координата нуль и тогда дискретные координаты этой точки (0; 0), т.е. она принадлежит П, а B_1B_2 пересекает П. Либо же вторая дискретная координата равна +1 у одной точки и –1 у другой. Но тогда некоторая выпуклая комбинация B_1 и B_2 будет иметь вторую дискретную координату равной нулю, т. е. она будет принадлежать П.

5. Очевидно, прямая $f(x,y) = y - \bar{y}_1 - k(x - \bar{x}_1) = 0$ (где $k = \dfrac{\bar{y}_2 - \bar{y}_1}{\bar{x}_2 - \bar{x}_1}$) проходит через точки B_1 и B_2 . Для любой точки C(x;y) из одной из открытых полуплоскостей, ограниченных этой прямой, $f(x,y) > 0$; для любой точки C(x;y) из другой полуплоскости $f(x,y) < 0$. Поэтому если все значения $\operatorname{sign} f(x_i,y_j)$, где i,j = 1,2 совпадают, то это означает, что все 4 вершины прямоугольника П $A_1(x_1; y_1)$, $A_2(x_2; y_1)$, $A_3(x_1; y_2)$ и $A_4(x_2; y_2)$ лежат в одной открытой полуплоскости, а прямая $f(x,y) = 0$ не пересекает П. Если же значения $\operatorname{sign} f(x_i,y_j)$ различны, то прямая разделяет вершины прямоугольника П, т.е. пересекает П. Остаётся показать, что в этом случае и отрезок B_1B_2 пересекает П. Действительно, по условию каждая из точек B_1 , B_2 имеет ненулевую дискретную координату. Например, если дискретная абсцисса у B_1 равна –1, то $\bar{x}_1 < x_1$. Но тогда по условию дискретная абсцисса у B_2 отлична от –1, т. е. $\bar{x}_2 \geq x_1$. Это означает, что значения абсциссы на прямой $f(x,y) = 0$ возрастают в направлении от B_1 к B_2 и убывают в противоположном направлении. Но тогда значения абсциссы во всех точках луча из B_1 в направлении, противоположном B_2, меньше, чем x_1 и следовательно все эти точки не принадлежат П. Аналогично проводится рассуждение в случае, если дискретная абсцисса у B_1 равна +1 , а также и в случае, если дискретная ордината у B_1 равна +1 или –1: во всех этих случаях приходим к выводу, что луч из B_1 в направлении, противоположном B_2, не пересекает прямоугольник П. То же и так же справедливо и для точки B_2: луч из B_2 в направлении, противоположном B_1, не пересекает прямоугольник П. Но тогда, если прямая B_1B_2 пересекает прямоугольник П, то его пересекает и отрезок B_1B_2 , что и требуется.

На этой теореме непосредственно основан алгоритм выделения из данного множества отрезков на плоскости подмножества пересекающих данный прямоугольник П. Далее на Рис.5 приведена принципиальная блок-схема этого алгоритма. Порядок пунктов в теореме соответствует порядку условий, проверяемых алгоритмом, и подобран так, чтобы достичь максимально возможной его эффективности.

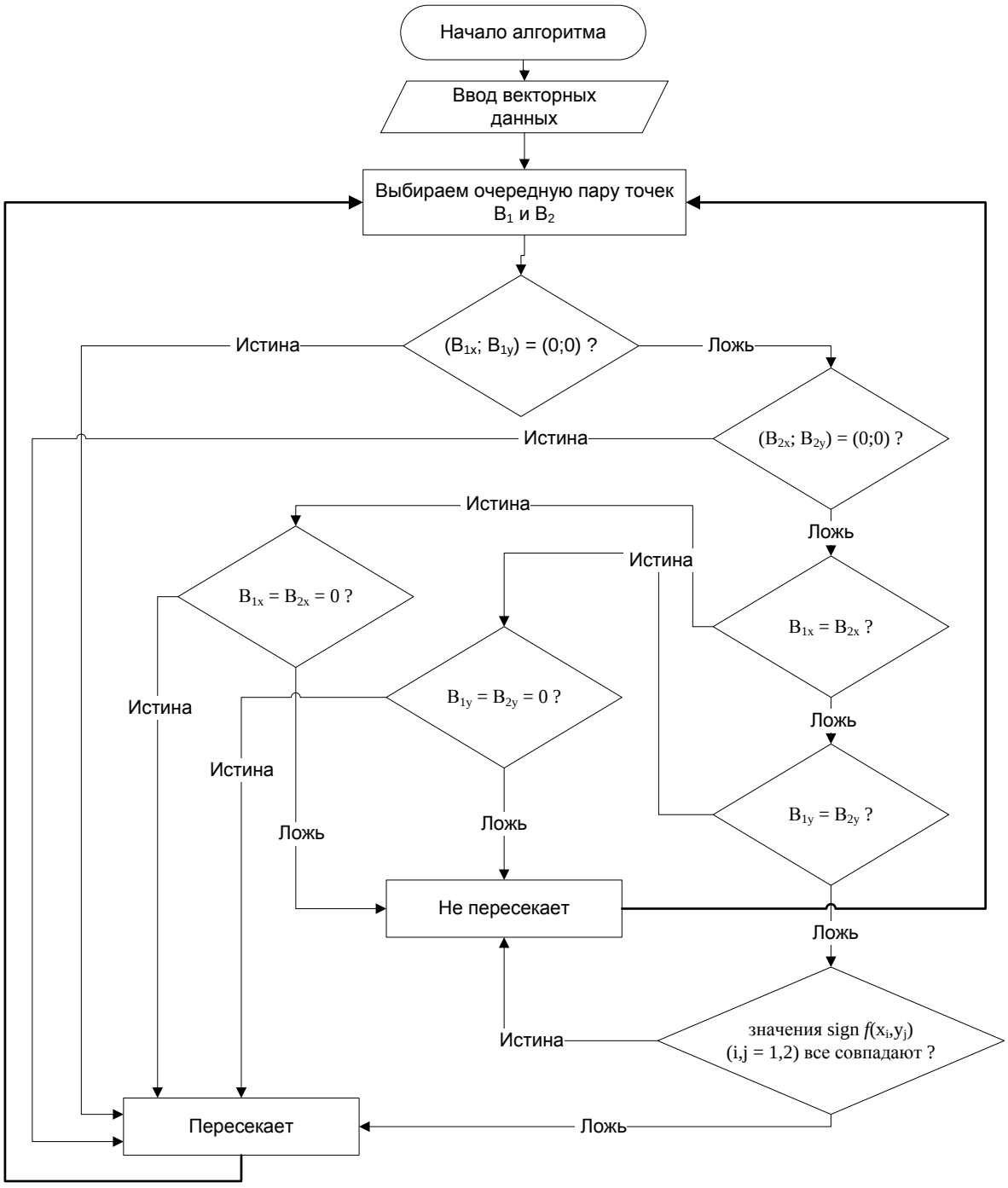

Рис. 5. Блок-схема алгоритма

Программная реализация алгоритма

Алгоритм реализован согласно парадигме объектно-ориентированного подхода на языке программирования *C++* в интегрированной среде *Microsoft Visual Studio 2008*.

Основные сущности, реализованные для работы программы:

- *Point* – пара действительных чисел, представляющая точку на плоскости;

```
class Point
{
	public:
		Point(float x, float y);
		inline float& getX();
		inline float& getY();
	protected:
		float m X;
		float m Y;
};
```

- *Edge* – пара точек, представляющих отрезок;

```
class Edge
{
	public:
		Edge(Point*, Point*);
		void setA(Point&);
		void setB(Point&);
		Point& getA();
		Point& getB();
	protected:
		Point m_A;
		Point m_B;
};
typedef std::vector<Edge*> EdgeList;
```

- *Surface* – двумерная плоскость заданного размера

```
class Surface
	{
	public:
		Surface(float, float);
		void assignVP(Viewport&);
		void generateEdges(int);
		void serialize(char*);
		void deserialize(char*);
		EdgeList getEdges() const;
	protected:
		bool belongToVP(Edge*);
		EdgeList m_Edges;
		float m_Width;
		float m Height;
		Viewport* vp;
};
```

- *Viewport* – прямоугольная область, которая расположена над Surface и может перемещаться по горизонтали и вертикали.

```
class Viewport
	{
	public:
		Viewport(float, float, float, float);
		void move(float, float, float, float);
		EdgeList getVpVisible();
		void inject(Edge&);
		float getTop() const;
		float getLeft() const;
		float getRight() const;
		float getBottom() const;
	protected:
		float m Top;
		float m Bottom;
		float m_Left;
		float m_Right;
		EdgeList m VisibleEdges;
};
```

Оценка эффективности алгоритма

Пусть дана область Ω с заданным на ней множеством отрезков Δ, $|\Delta| = N$. Пусть далее дана прямоугольная область $\Pi \subset \Omega$. Задача состоит в выборе тех отрезков из Δ, которые полностью или частично попадают в Π.

Согласно доказанной выше теореме, существует 4 типа отрезков:

- полностью лежит в Π

- одна из точек находится в П
- ребро пересекает грани области П
- ребро не пересекает П.

Основной шаг алгоритма:

```
bool Surface::belongToVP(geo::Edge* edge)
{
        float vpXl = vp->getLeft();
        float vpYt = vp->getTop();
        float vpXr = vp->getRight();
        float vpYb = vp->getBottom();

        float edXa = edge->getA().getX();
        float edYa = edge->getA().getY();
        float edXb = edge->getB().getX();
        float edYb = edge->getB().getY();

        bool aInXrange = edXa>vpXl && edXa<vpXr;
        bool aInYrange = edYa>vpYt && edYa<vpYb;

        bool bInXrange = edXb>vpXl && edXb<vpXr;
        bool bInYrange = edYb>vpYt && edYb<vpYb;

        if (!((aInXrange || bInXrange) && (aInYrange || bInYrange)))
                return false;

        if (aInXrange && aInYrange)
                return true;

        if (bInXrange && bInYrange)
                return true;

        if (aInXrange && bInXrange)
                return true;

        if (aInYrange && bInYrange)
                return true;

        return (0>((vpXl-edXa)*(edYb-edYa)-(vpYb-edYa)*(edXb-edXa))*
                        ((vpXl-edXa)*(edYb-edYa)-(vpYt-edYa)*(edXb-edXa))*
                        ((vpXr-edXa)*(edYb-edYa)-(vpYt-edYa)*(edXb-edXa))*
                        ((vpXr-edXa)*(edYb-edYa)-(vpYb-edYa)*(edXb-edXa)));

}
```

Для определения принадлежности отрезка области П нам достаточно проверить одну или же обе его вершины. Число операций C просто пропорционально числу отрезков $|\Delta| = N$, $C = k\,N$: алгоритм имеет линейную сложность. Как видно из текста программы, предельно мал коэффициент пропорциональности k. Следовательно, если тройка (Ω ; П; Δ) в условии задачи фиксирована, то для такой задачи полученный алгоритм максимально эффективен.

Заметим, что теорема этой статьи имеет очевидное n – мерное обобщение. В этом случае вместо прямоугольника задан n – мерный параллелепипед, ограниченный гиперплоскостями, параллельными координатным гиперплоскостям. Формулировка и доказательство фактически воспроизводят приведённую выше для двумерного случая. В трёхмерном случае соответствующий алгоритм решает задачу о выборе маршрутов самолётов, проходящих через данную область воздушного пространства [10]. В n – мерном случае – задачу о выборе частиц, координаты которых могут войти в данную область фазового пространства [11].

Выводы. Полученный алгоритм для рассматриваемой в этой работе задачи об эффективном отображении эффективен настолько, насколько это вообще возможно, что для такой задачи имеет решающее значение. Поэтому можно надеяться, что применение его в этой богатой приложениями тематике окажется успешным. Вместе с тем он может быть использован для решения задачи о выборе маршрутов самолётов, проходящих через данную область воздушного пространства, задачи о выборе частиц, координаты которых могут войти

в данную область фазового пространства и других задач, связанных с эффективным отбором пространственных объектов.

Перспективы дальнейших исследований

Интересно сравнить эффективность этого алгоритма с эффективностью алгоритма, основанного на пространственной индексации R-деревьями. Для этого алгоритма число операций C для рассмотренной выше задачи равно $C = a \cdot N + b \cdot \ln N$, где заведомо $a > b$. Однако, если рассмотреть важную с точки зрения приложений задачу с переменным П (скажем, затребована возможность отображения как всей карты, так и любой её части в любой момент в режиме реального времени), то соотношение меняется радикально. Если R-дерево уже сформировано, то при выборе последующих прямоугольников П может оказаться, что $a = 0$, $C = b \cdot \ln N$: для такой задачи этот алгоритм эффективнее алгоритма, основанного на дискретных координатах. Уже на приведённом здесь примере можно заметить то, к чему и приводит подробный анализ: в действительности эти алгоритмы не альтернативны, они выполняют различные функции и могут быть совмещены. По всей видимости, такое скрещивание должно привести к построению ещё более эффективных программ, использующих преимущества обоих методов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. http://forum.ixbt.com – «Быстрое отображение векторной графики. Алгоритмы»
2. http://maps.google
3. http://meta.ua
4. http://maps.yandex.ru
5. http://gis-lab.info – «GIS-Lab - Географические информационные системы и дистанционное зондирование». 2002-2010
6. И. С. Березин, Н. П. Жидков Методы вычислений т. 1. / – М.: “Наука”, 1966. – 632 с.
7. http://noregret.org – «Известные алгоритмы определения столкновений и реакции на них во Flash»
8. http://ru.wikipedia.org/wiki/R-дерево – «R-дерево»
9. http://www.superliminal.com – «Free Source Code»
10. Н. С. Бахвалов Численные методы т. 1. / – М.: “Наука”, 1973. – 631 с.
11. М. Рид, Б. Саймон Методы современной математической физики т.1. / М: Мир, 1977. 358 с.

Рецензент: Кравцов Г.М.