**Министерство образования Республики Беларусь**

**Учреждение образования**

**«Белорусский государственный университет информатики   
и радиоэлектроники»**

Кафедра интеллектуальных

информационных технологий

**Отчёт**

по лабораторной работе № 5

по курсу «Графический интерфейс интеллектуальных систем»

Выполнили:

студенты группы 821701

Бозбей М.С.

Михневич Е.Д.

Минск 2011

**Цель:**

Изучить алгоритмы построения замкнутых областей. Разработать редактор, позволяющий разбиение плоскости на сегменты согласно алгоритму Вороного (диаграммы Вороного).

**Алгоритм:**

В лабораторной работе был применен алгоритм Форчуна. Он основан на применении заметающей прямой. Заметающая прямая — это вспомогательный объект, представляющий собой вертикальную прямую линию. На каждом шаге алгоритма диаграмма Вороного построена для множества, состоящего из заметающей прямой и точек слева от неё. При этом граница между областью Вороного прямой и областями точек состоит из отрезков парабол (так как геометрическое место точек, равноудалённых от заданной точки и прямой — это [парабола](http://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B1%D0%BE%D0%BB%D0%B0)). Прямая движется слева направо. Каждый раз, когда она проходит через очередную точку, эта точка добавляется к уже построенному участку диаграммы. Добавление точки к диаграмме при использовании [двоичного дерева поиска](http://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%BE%D0%B8%D1%87%D0%BD%D0%BE%D0%B5_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_%D0%BF%D0%BE%D0%B8%D1%81%D0%BA%D0%B0) имеет сложность *O*(log *n*), всего точек *n*, а сортировка точек по *x*-координате может быть выполнена за *O*(*n*log*n*), поэтому [вычислительная сложность](http://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D1%8C) алгоритма Форчуна равна *O*(*n*log*n*).

**draw: function() {**

**var ctx = this.canvas.getContext('2d');**

**this.drawBackground(ctx);**

**this.drawSites(ctx);**

**if (this.sweep < this.canvas.height) {**

**ctx.globalAlpha=0.9;**

**ctx.strokeStyle='#00f';**

**ctx.lineWidth=0.5;**

**ctx.beginPath();**

**ctx.moveTo(0,this.sweep);**

**ctx.lineTo(this.canvas.width,this.sweep);**

**ctx.stroke();**

**}**

**this.drawEdges(ctx);**

**if (!this.queueIsEmpty()) {**

**this.drawVertices(ctx);**

**}**

**if (this.sweep < this.canvas.height) {**

**this.drawBeachline(ctx);**

**}**

**},**

**drawBackground: function(ctx) {**

**ctx.globalAlpha = 1;**

**ctx.beginPath();**

**ctx.rect(0,0,this.canvas.width,this.canvas.height);**

**ctx.fillStyle = '#fff';**

**ctx.fill();**

**ctx.strokeStyle = '#888';**

**ctx.stroke();**

**},**

**drawSites: function(ctx) {**

**var queueIsEmpty = this.queueIsEmpty();**

**ctx.beginPath();**

**var nSites=this.sites.length;**

**for (var iSite=0; iSite<nSites; iSite++){**

**var site=this.sites[iSite];**

**if (queueIsEmpty) {**

**ctx.rect(site.x-0.25,site.y-0.25,1.5,1.5);**

**}**

**else {**

**ctx.rect(site.x-0.5,site.y-0.5,2,2);**

**}**

**}**

**ctx.globalAlpha = 1;**

**ctx.fillStyle = '#000';**

**ctx.fill();**

**},**

**drawCells: function() {**

**var colvalues = '0123456789ABCDEF';**

**var ctx = this.canvas.getContext('2d');**

**var cells = this.getCells();**

**if (!cells) {return;}**

**var halfedges, nHalfedges, iHalfedge;**

**var v;**

**for (var cellid in cells) {**

**halfedges = cells[cellid].halfedges;**

**nHalfedges = halfedges.length;**

**v = halfedges[0].getStartpoint();**

**ctx.beginPath();**

**ctx.moveTo(v.x,v.y);**

**for (iHalfedge=0; iHalfedge<nHalfedges; iHalfedge++) {**

**v = halfedges[iHalfedge].getEndpoint();**

**ctx.lineTo(v.x,v.y);**

**}**

**ctx.fillStyle='#'+colvalues[(this.random()\*16)&15]+colvalues[(this.random()\*16)&15]+colvalues[(this.random()\*16)&15];**

**ctx.fill();**

**}**

**},**

**drawBeachline: function(ctx) {**

**var nArcs=this.arcs.length;**

**if (!nArcs) {return;}**

**var cw = this.canvas.width;**

**ctx.lineWidth = 1;**

**var directrix = this.sweep;**

**var arc = this.arcs[0];**

**var xl = 0;**

**var yl, xr, yr;**

**var focx = arc.site.x;**

**var focy = arc.site.y;**

**var p;**

**if (focy == directrix) {**

**xl = focx;**

**yl = 0;**

**}**

**else {**

**p = (focy-directrix)/2;**

**yl = (focx\*focx)/(4\*p)+focy-p;**

**}**

**var neighbour;**

**var ac\_x, ac\_y, bc\_x, bc\_y, gx, gy, n;**

**var pi\_by\_2 = this.PI\*2;**

**for (var iArc=0; iArc<nArcs; iArc++) {**

**arc = this.arcs[iArc];**

**focx=arc.site.x;**

**focy=arc.site.y;**

**if ( arc.isCollapsing() ) {**

**var circEvent = arc.circleEvent;**

**ctx.save();**

**ctx.globalAlpha=0.25;**

**ctx.fillStyle='#800';**

**ctx.fillRect(circEvent.center.x-0.5,circEvent.center.y-0.5,2,2);**

**ctx.beginPath();**

**ctx.arc(circEvent.center.x,circEvent.center.y,circEvent.y-circEvent.center.y,0,pi\_by\_2,true);**

**ctx.strokeStyle='#aaa';**

**ctx.stroke();**

**ctx.fillStyle='#aaa';**

**ctx.beginPath();**

**ctx.fillRect(circEvent.x-0.5,circEvent.y-0.5,2,2);**

**ctx.restore();**

**}**

**if (focy == directrix) {**

**xr = focx;**

**neighbour = iArc>0 ? this.arcs[iArc-1] : null;**

**if (!neighbour || neighbour.site.y == directrix) {**

**neighbour = iArc < this.arcs.length-1 ? this.arcs[iArc+1] : null;**

**}**

**if (!neighbour || neighbour.site.y == directrix) {**

**yr = 0;**

**}**

**else {**

**p = (neighbour.site.y-directrix)/2;**

**yr = this.pow(focx-neighbour.site.x,2)/(4\*p)+neighbour.site.y-p;**

**}**

**ctx.strokeStyle = '#080';**

**ctx.beginPath();**

**ctx.moveTo(focx,focy);**

**ctx.lineTo(focx,yr);**

**ctx.stroke();**

**xl=xr;**

**yl=yr;**

**continue;**

**}**

**xr = this.min(this.rightBreakPoint(iArc,directrix),cw);**

**p = (focy-directrix)/2;**

**yr = this.pow(xr-focx,2)/(4\*p)+focy-p;**

**if (xr >= 0 && xl < cw && xr > xl) {**

**ctx.strokeStyle = arc.isCollapsing() ? '#800' : '#080';**

**ac\_x = focx-xl;**

**ac\_y = focy-directrix;**

**bc\_x = focx-xr;**

**bc\_y = focy-directrix;**

**gx = (xr+focx)/2;**

**gy = (directrix+focy)/2;**

**n = ((gx-(xl+focx)/2)\*ac\_x+(gy-(directrix+focy)/2)\*ac\_y)/(bc\_y\*ac\_x-bc\_x\*ac\_y);**

**ctx.beginPath();**

**ctx.moveTo(xl,yl);**

**ctx.quadraticCurveTo(gx-bc\_y\*n,gy+bc\_x\*n,xr,yr);**

**ctx.stroke();**

**}**

**xl=xr;**

**yl=yr;**

**}**

**},**

**drawVertices: function(ctx) {**

**ctx.beginPath();**

**ctx.globalAlpha=1;**

**var nEdges=this.edges.length;**

**var edge;**

**var va, vb;**

**for (var iEdge=0; iEdge<nEdges; iEdge++) {**

**edge=this.edges[iEdge];**

**va = edge.va;**

**if (va !== undefined) {**

**ctx.rect(va.x-0.75,va.y-0.75,2.5,2.5);**

**}**

**vb = edge.vb;**

**if (vb !== undefined) {**

**ctx.rect(vb.x-0.75,vb.y-0.75,2.5,2.5);**

**}**

**}**

**ctx.fillStyle='#07f';**

**ctx.fill();**

**},**

**drawEdges: function(ctx) {**

**ctx.beginPath();**

**ctx.lineWidth=0.5;**

**ctx.globalAlpha=1;**

**var nEdges=this.edges.length;**

**var edge;**

**var va, vb;**

**for (var iEdge=0; iEdge<nEdges; iEdge++) {**

**edge=this.edges[iEdge];**

**if (edge.va === undefined || edge.vb === undefined) {continue;}**

**va = edge.va;**

**vb = edge.vb;**

**ctx.moveTo(va.x,va.y);**

**ctx.lineTo(vb.x,vb.y);**

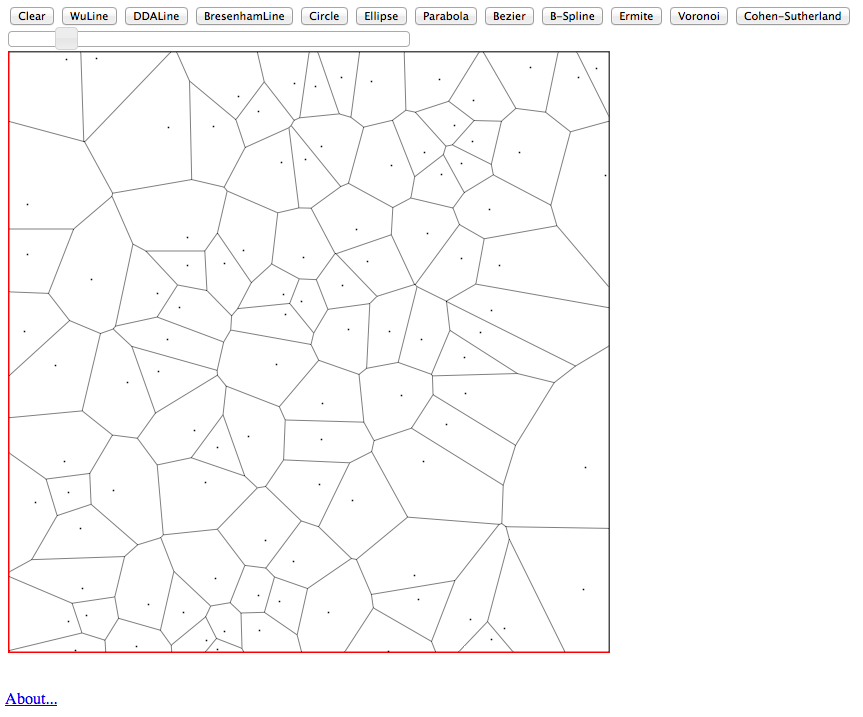
**}**

**ctx.strokeStyle='#000';**

**ctx.stroke();**

**},**

**Пример работы программы:**



**Выводы:**

В ходе лабораторной работы были изучен алгоритм Вороного, а точнее его вариант реализации - алгоритм Форчуна. Был разработан графический редактор, который позволяет в разных режимах увидеть разбиение области на сектора. Предусмотрен режим пошаговой отладки алгоритма, когда мы можем перемещаться к следующему пикселю заметающей прямой или к следующей точке на плоскости. Также разработан интерфейс для расстановки точек на плоскости и их удаления.