**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

отчет

**по лабораторной работе №4**

**по дисциплине «Компьютерная графика»**

Тема: **Исследование алгоритмов отсечения отрезков и многоугольников окнами различного вида**

|  |  |  |
| --- | --- | --- |
| Студенты гр. 8362 |  | Ларионова Е.Е.  Матвеев Н.Д. |
| Преподаватель |  | Матвеева И. В. |

Санкт-Петербург

2021

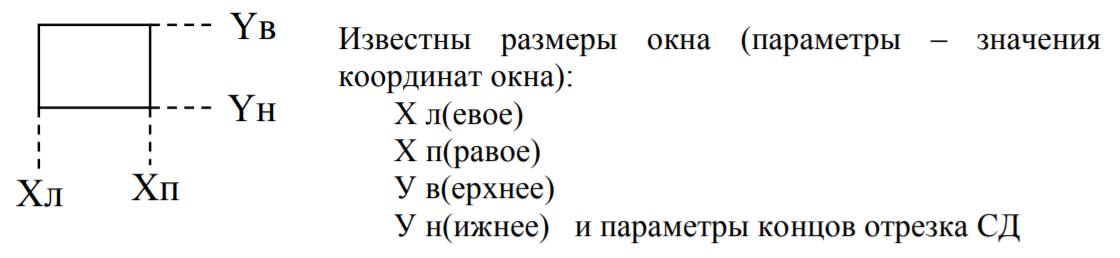
**ЗАДАНИЕ**

Обеспечить реализацию простого алгоритма отсечения массива произвольных отрезков заданным прямоугольным окном. Массив отрезков следует формировать генератором случайных чисел. Вначале следует вывести на экран сгенерированные отрезки полностью, а затем другим цветом или яркостью те, которые полностью или частично попадают в область окна.

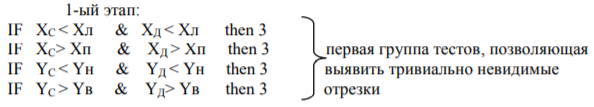
**ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

Отсечение необходимо, чтобы из обширной базы данных выделить отдельные элементы для вывода на дисплей или принтер. Из-за разнообразных окон (прямоугольных со сторонами параллельными осям координат, выпуклых произвольных и невыпуклых), а также графических объектов, подлежащих отсечению, разнообразной конфигурации в М.Гр. разработано многозначительное количество алгоритмов, учитывающее особенности как окон, так и объектов.

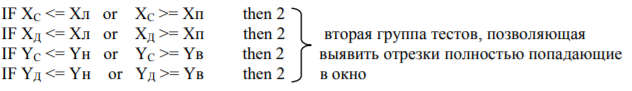
Простейший алгоритм отсечения прямоугольным окном



Простой алгоритм сводится на начальном этапе к сравнению границ окна с параметрами отрезка. Для этого сравнивают координаты начала и конца отрезка с границами окна:



Если хотя бы одна проверка не выполняется – то отрезок не виден, и надо переходить к анализу следующего отрезка, т.е. на конец анализа текущего отрезка и к возврату на начало анализа.



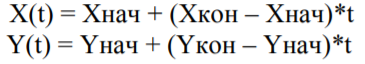
Если все эти очередные 4 теста не проходят – отрезок виден и его отображают в окне и переходят к анализу следующего отрезка, т.е. к возврату на начало алгоритма.

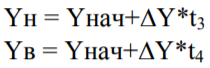
2. Определение нетривиально невидимого отрезка (СD - №5) или видимой части частично видимого отрезка на основе его параметрического описания Р(t) и параметров границ окна:

описание отрезка:

Р(t) = Pначальное + (Pконечное – Pначальное)\*t 0 ≤ t ≤ 1

или





и определяется соответствующее этим границам ti.

Далее, если ti оказывается в диапазоне задания отрезка: 0 ≤ t ≤ 1, осуществляется вычисление второй координаты точки пересечения с соответствующей границей окна и проверка на попадание этой точки в границы окна (например, Хл ≤ Хi ≤ Хп или Yн ≤ Yi ≤ Yв) в соответствии с диапазоном задания отрезка: 0 ≤ t ≤ 1.

Если такая точка не попадает в соответствующие границы окна, то это отрезок типа СД (5) , характеризующийся не тривиальной невидимостью.

Если таких точек оказывается две, то ti и tj определяют видимую часть отрезка.

Если таких точек одна, то необходимо определить параметры второго конца видимого отрезка, т.е. в окно попадает начало (t = 0) или конец (t = 1) анализируемого отрезка.

После этого можно вывести частично видимый отрезок и перейти к анализу следующего

**РЕАЛИЗАЦИЯ ПРОГРАММЫ**

При запуске программы открываются 2 окна «MainWindow» и «Form». В «MainWindow» задаются координаты точек. В «Form» служит для отрисовки системы координат, заданного квадрата и линий (Рисунок 1 и Рисунок 2).

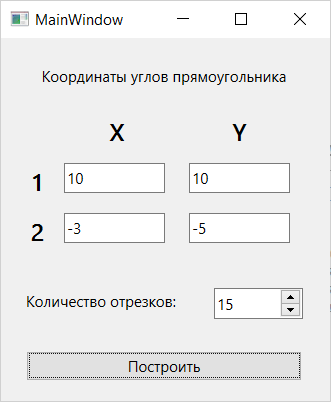


Рисунок 1 – Окно «MainWindow»

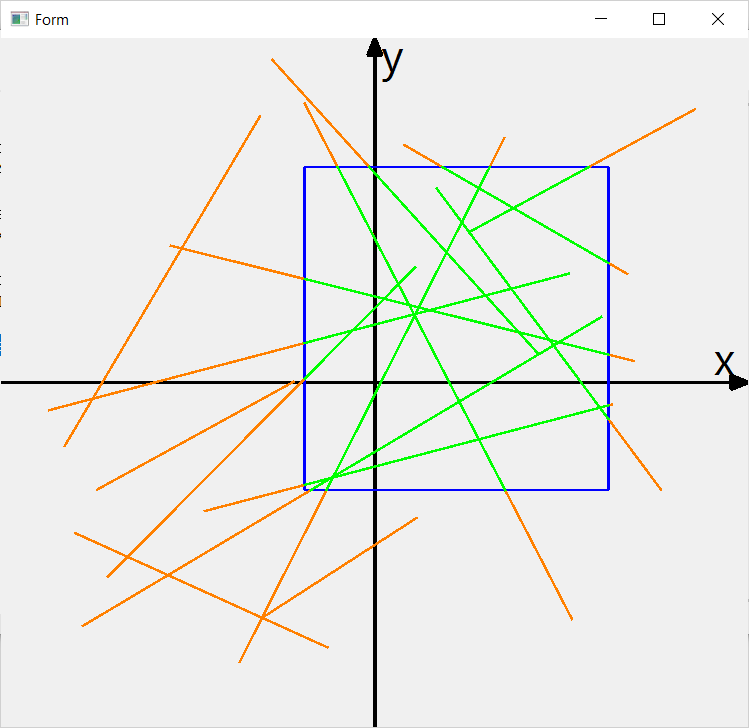


Рисунок 1 – Окно «Form»

**ПРИЛОЖЕНИЕ 1 – КОД ПРОГРАММЫ**

**Файл main.cpp**

#include <application.h>

int main(int argc, char \*argv[])

{

Application a(argc, argv);

return a.exec();

}

**Файл mainwindow.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#define DIF 0.01

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::on\_pushButton\_clicked()

{

ControlState \*c = new ControlState;

c->p1.setX(ui->lineEdit\_x\_1->text().toDouble());

c->p1.setY(ui->lineEdit\_y\_1->text().toDouble());

c->p2.setX(ui->lineEdit\_x\_2->text().toDouble());

c->p2.setY(ui->lineEdit\_y\_2->text().toDouble());

if (abs(c->p1.x()-c->p2.x())<DIF||abs(c->p1.y()-c->p2.y())<DIF)

{

QMessageBox::warning(nullptr,"Ошибка","Необходимо указать координаты противоположных углов прямоугольника");

delete c;

return;

}

c->sec\_num = ui->spinBox->value();

emit(send\_control(c));

}

**Файл drawwindow.cpp**

#include "drawwindow.h"

#include "ui\_drawwindow.h"

DrawWindow::DrawWindow(QWidget \*parent) :

QWidget(parent),

ui(new Ui::DrawWindow)

{

ui->setupUi(this);

d = nullptr;

}

DrawWindow::~DrawWindow()

{

delete ui;

}

void DrawWindow::recive\_draw(DrawState \*rds)

{

d=rds;

repaint();

}

void DrawWindow::paintEvent (QPaintEvent \*event)

{

Q\_UNUSED(event);

QPainter painter(this);

QFont font;

support\_state s;

s.cw = 0.5\*rect().width();

s.ch = 0.5\*rect().height();

int n;

if (d == nullptr)

{

n = 5;

}

else

{

n = d->lim;

}

s.ew = (s.cw) / (n);

s.eh = (s.ch) / (n);

qreal c = s.cw>s.ch?s.ch:s.cw;

qreal ca = 0.05\*c;

qreal caa = ca \* 0.4;

qreal cf = 0.1 \* c;

font.setPointSize(cf);

painter.setFont(font);

QPen mp;

mp.setColor(Qt::black);

mp.setWidth(3);

QPen sp;

sp.setColor(Qt::blue);

sp.setWidth(2);

QPen ip;

ip.setColor(Qt::green);

ip.setWidth(2);

QPen op;

op.setColor(QColor::fromRgb(255, 128, 0));

op.setWidth(2);

painter.setPen(mp);

painter.drawLine(QLineF(0,s.ch,2\*s.cw,s.ch));

painter.drawLine(QLineF(s.cw,0,s.cw,2\*s.ch));

QPointF arr1[3],arr2[3];

arr1[0] = QPointF(s.cw,0);

arr1[1] = QPointF(s.cw-caa,ca);

arr1[2] = QPointF(s.cw+caa,ca);

arr2[0] = QPointF(2\*s.cw,s.ch);

arr2[1] = QPointF(2\*s.cw-ca,s.ch+caa);

arr2[2] = QPointF(2\*s.cw-ca,s.ch-caa);

painter.setBrush(QBrush(Qt::black));

painter.drawPolygon(arr1,3);

painter.drawPolygon(arr2,3);

painter.drawText(arr1[2]+QPointF(0,0.5\*cf),"y");

painter.drawText(arr2[2]-QPointF(0.5\*cf,0),"x");

if (d != nullptr)

{

QPointF square[4];

for (size\_t i = 0; i<4; i++)

{

square[i] = transform(d->p[i],s);

}

painter.setBrush(QBrush(Qt::transparent));

painter.setPen(sp);

painter.drawPolygon(square,4);

for (size\_t i = 0; i < d->sec.size(); i++)

{

switch (d->sec[i].visible)

{

case 0:

{

painter.setPen(op);

painter.drawLine(transform(d->sec[i].line,s));

break;

}

case 1:

{

painter.setPen(ip);

painter.drawLine(transform(d->sec[i].line,s));

break;

}

case 2:

{

painter.setPen(op);

painter.drawLine(transform(d->sec[i].line,s));

painter.setPen(ip);

QLineF tmp;

tmp.setP1(QPointF(d->sec[i].line.p1()+(d->sec[i].line.p2()-d->sec[i].line.p1())\*d->sec[i].t[0]));

tmp.setP2(QPointF(d->sec[i].line.p1()+(d->sec[i].line.p2()-d->sec[i].line.p1())\*d->sec[i].t[1]));

painter.drawLine(transform(tmp,s));

break;

}

default:

{

break;

}

}

}

}

}

QPointF DrawWindow::transform(QPointF a, support\_state s)

{

QPointF tmp;

tmp.setX(s.cw + a.x()\*s.ew);

tmp.setY(s.ch - a.y()\*s.eh);

return tmp;

}

QLineF DrawWindow::transform(QLineF a, support\_state s)

{

QLineF tmp;

tmp.setP1(transform(a.p1(),s));

tmp.setP2(transform(a.p2(),s));

return tmp;

}

**Файл application.cpp**

#include "application.h"

#define SPREAD\_COEFFICIENT 1.5

Application::Application(int argc, char \*argv[])

: QApplication(argc,argv)

{

d = new DrawWindow;

m = new MainWindow;

d->show();

m->show();

connect(m,SIGNAL(send\_control(ControlState\*)),

this,SLOT(recive\_control(ControlState\*)));

connect(this,SIGNAL(send\_draw(DrawState\*)),

d,SLOT(recive\_draw(DrawState\*)));

}

void Application::recive\_control(ControlState\* c)

{

DrawState \*d = new DrawState;

d->p[0] = c->p1;

d->p[1] = QPointF(c->p1.x(),c->p2.y());

d->p[2] = c->p2;

d->p[3] = QPointF(c->p2.x(),c->p1.y());

qreal Xl = qMin(c->p1.x(),c->p2.x());

qreal Xr = qMax(c->p1.x(),c->p2.x());

qreal Yu = qMax(c->p1.y(),c->p2.y());

qreal Yd = qMin(c->p1.y(),c->p2.y());

int lim = static\_cast <int> (qMax(qMax(abs(Xl),abs(Xr)),qMax(abs(Yu),abs(Yd)))\*100\*SPREAD\_COEFFICIENT);

d->lim = (lim/100)+1;

for (size\_t i = 0; i<c->sec\_num; i++)

{

MyLineF tmp;

//блок генерации отрезка

srand(QTime::currentTime().msecsSinceStartOfDay()\*(i+1));

tmp.line.setP1(QPointF((rand()%(lim\*2)-lim)\*0.01,(rand()%(lim\*2)-lim)\*0.01));

tmp.line.setP2(QPointF((rand()%(lim\*2)-lim)\*0.01,(rand()%(lim\*2)-lim)\*0.01));

//блок определения видимости отрезков

unsigned short code1 = 0, code2 = 0;

//вычисления для p1

if (tmp.line.p1().x() < Xl) code1 += 8;

if (tmp.line.p1().x() > Xr) code1 += 4;

if (tmp.line.p1().y() < Yd) code1 += 2;

if (tmp.line.p1().y() > Yu) code1 += 1;

//вычисления для p2

if (tmp.line.p2().x() < Xl) code2 += 8;

if (tmp.line.p2().x() > Xr) code2 += 4;

if (tmp.line.p2().y() < Yd) code2 += 2;

if (tmp.line.p2().y() > Yu) code2 += 1;

if ((code1 == 0)&&(code2 == 0))

{

//отсечение полностью видимых отрезков

tmp.visible = 1;

}

else if((code1&code2) != 0)

{

//отсечение тривиально невидимых отрезков

tmp.visible = 0;

}

else

{

//блок обработки нетривиальных отрезков

qreal t1,t2,t3,t4;

if ((tmp.line.p2().x()-tmp.line.p1().x())!=0)

{

//условие пересечения левой границы

t1 = (Xl-tmp.line.p1().x())/(tmp.line.p2().x()-tmp.line.p1().x());

if ((t1 >= 0)&&(t1 <= 1))

{

if (((tmp.line.p1().y()+(tmp.line.p2().y()-tmp.line.p1().y())\*t1) >= Yd)&&((tmp.line.p1().y()+(tmp.line.p2().y()-tmp.line.p1().y())\*t1) <= Yu))

{

tmp.t.push\_back(t1);

}

}

//условие пересечения правой границы

t2 = (Xr-tmp.line.p1().x())/(tmp.line.p2().x()-tmp.line.p1().x());

if ((t2 >= 0)&&(t2 <= 1))

{

if (((tmp.line.p1().y()+(tmp.line.p2().y()-tmp.line.p1().y())\*t2) >= Yd)&&((tmp.line.p1().y()+(tmp.line.p2().y()-tmp.line.p1().y())\*t2) <= Yu))

{

tmp.t.push\_back(t2);

}

}

}

if ((tmp.line.p2().y()-tmp.line.p1().y())!=0)

{

//условие пересечения нижней границы

t3 = (Yd-tmp.line.p1().y())/(tmp.line.p2().y()-tmp.line.p1().y());

if ((t3 >= 0)&&(t3 <= 1))

{

if (((tmp.line.p1().x()+(tmp.line.p2().x()-tmp.line.p1().x())\*t3) >= Xl)&&((tmp.line.p1().x()+(tmp.line.p2().x()-tmp.line.p1().x())\*t3) <= Xr))

{

tmp.t.push\_back(t3);

}

}

//условие пересечения верхней границы

t4 = (Yu-tmp.line.p1().y())/(tmp.line.p2().y()-tmp.line.p1().y());

if ((t4 >= 0)&&(t4 <= 1))

{

if (((tmp.line.p1().x()+(tmp.line.p2().x()-tmp.line.p1().x())\*t4) >= Xl)&&((tmp.line.p1().x()+(tmp.line.p2().x()-tmp.line.p1().x())\*t4) <= Xr))

{

tmp.t.push\_back(t4);

}

}

}

if (tmp.t.size()==0)

{

tmp.visible = 0;

}

else

{

tmp.visible = 2;

if(tmp.t.size()==1)

{

if (code1 == 0)

{

tmp.t.push\_back(0);

}

else

{

tmp.t.push\_back(1);

}

}

}

}

d->sec.push\_back(tmp);

}

emit(send\_draw(d));

delete c;

}