Министерство образования и науки Российской Федерации Федеральное агентство по образованию Федеральное государственное бюджетное образовательное учреждение высшего образования «Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Лабораторная работа №1

по курсу «Компьютерная графика»

**«Реализация алгоритмов растровой развертки»**

Выполнил студент группы ИВТ-21\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Птахова А.М/

Проверил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Коржавина А.С./

Киров 2021

**1 Постановка задачи**

Написать программу, реализующую алгоритмы построения прямой: простой пошаговый алгоритм и алгоритмы Брезенхема для четырех- и восьмисвязной развертки. Проверить правильность работы программы, нарисовав, например, каждым алгоритмом семейство радиальных прямых, выходящих из одной точки с шагом 15 градусов. Построить окружности

**2 Словесное описание алгоритмов**

Простой пошаговый алгоритм: Изначально для реализации алгоритма имеются следующие данные: m – угловой коэффициент (-1 < m < 1), (x1; x2) и (y1; y2) – координаты концов отрезка, цвет для подкрашивания точек отрезка, процедура для подкрашивания некоторым цветом пиксела с координатами (x, y) и функция округления x до ближайшего целого. Если увеличивать с определенным шагом координату Х, а затем находить координату Y используя уравнение прямой Y = mX + b и подкрашивать пиксель с координатами (Х, ROUND(Y)), то потребуется много времени (на нецелочисленные операции). Если шаг по Х принять равным единице, то m = dY/dX сводится к m = dY, т.е. изменение Х на единицу приведет к изменению углового коэффициента на m. Таким образом, если Х(I + 1) = X(I) + 1,то Y(I + 1)=Y(I) + m. Алгоритм корректно работает только для отрезков в первом и восьмом октандах. В остальных случаях требует модификации, что предлагается сделать студентам самостоятельно в ходе выполнения данной лабораторной работы. При модификации следует учесть, что при m > 1, единичный шаг по Х приведёт к такому увеличению Y, при котором две соседние точки на прямой расположатся далеко друг от друга. Поэтому X и Y следует поменять, чтобы увеличивать на единицу Y, а Х - на dX = dY/m = 1/m.

Общий алгоритм Брезенхема: Суть алгоритма в следующем: в процессе работы одна из координат либо х, либо y (в зависимости от углового коэффициента) изменяется на единицу. Изменение другой координаты (на 0 или 1) зависит от расстояния – е между действительным положением отрезка и ближайшими координатами растра (е назовем управляющей переменной). Алгоритм построен так, что на каждом шаге проверяется лишь знак е и корректируется ее значение после каждого изменения очередной координаты. Значение исходной управляющей переменной: e = 2\*(y2 - y1) - (x2 – x1), где x1, y1, x2, y2 - координаты начальной и конечной точек отрезка. В каждом шаге при e ≥ 0 значение y от предыдущего увеличивается на единицу, а е уменьшается на 2\*(x2 - x1), в противном случае – y не меняется, а значение e увеличивается на 2\*(y2 - y1). В обоих случаях координата х следующего пиксела увеличивается на единицу от предыдущего значения.

**3 Вывод**

Для каждого алгоритма (простой, 8-связный алгоритм Брезенхема и 4- связный алгоритм Брезенхема) были написаны процедуры, реализующие эти алгоритмы и программа, позволяющая работать с этими процедурами. Наиболее естественным из приведённых алгоритмов – 8-связный алгоритм Брезенхема, т.к. при рисовании с помощью него получаются наименее угловатые линии, также он более оптимизирован, чем простой алгоритм. Соответственно, наиболее плохим вариантом является простой алгоритм, т.к. при углах, близких к 90° и 270° линия распадается на отдельные точки. Схемы алгоритмов, листинг процедур, реализующих их, и экранные формы программы приведены в приложениях А, Б и В.

Приложение А

Схемы алгоритмов

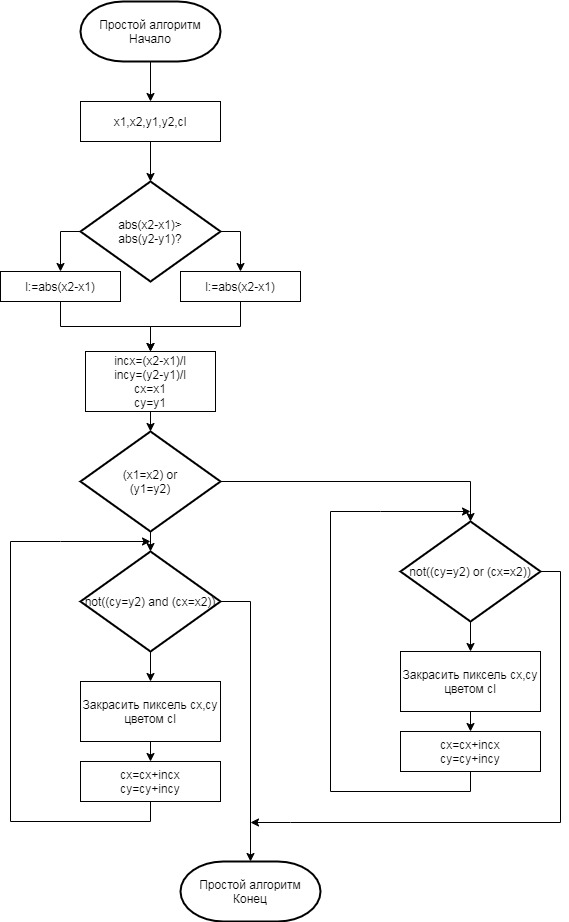
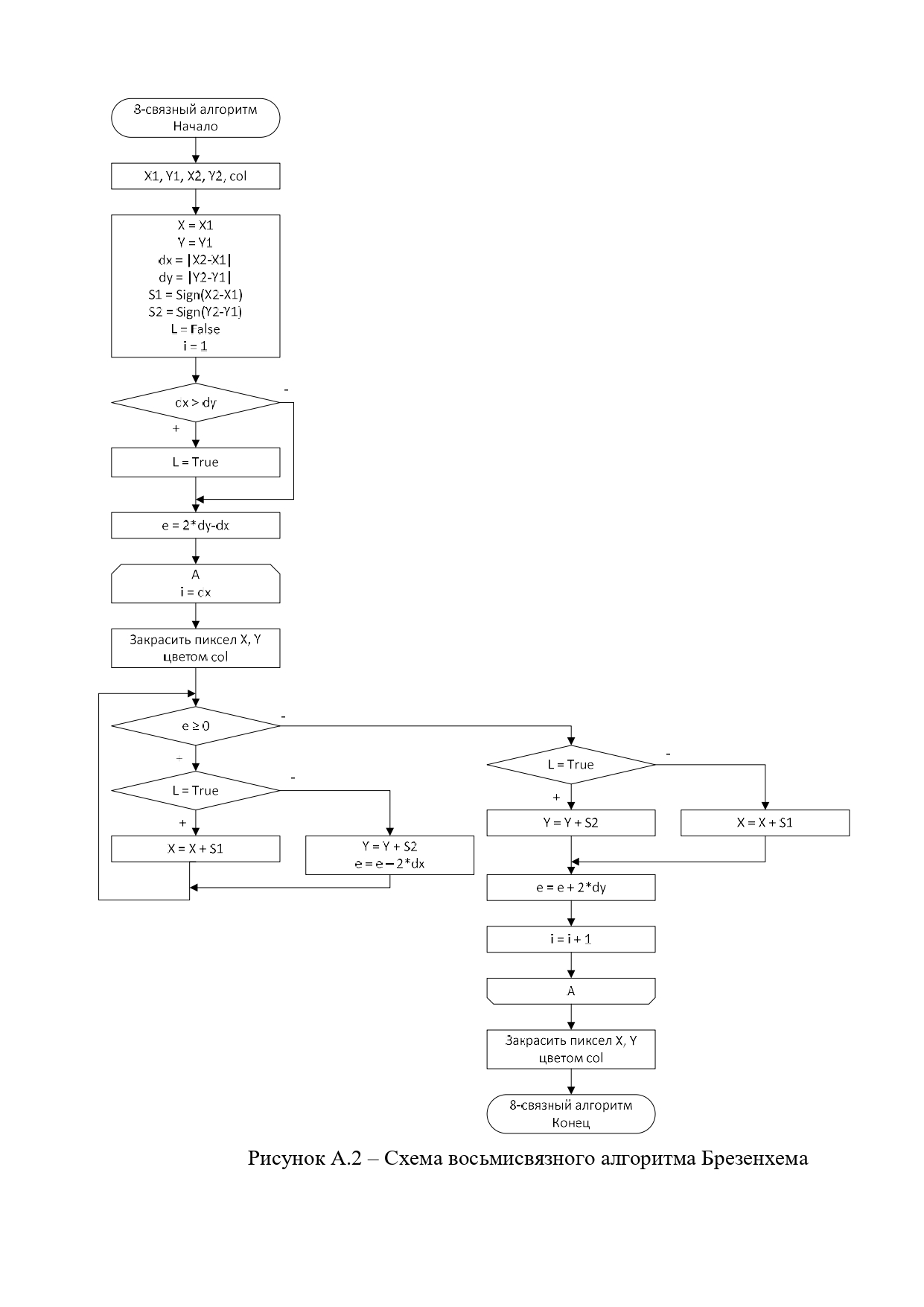
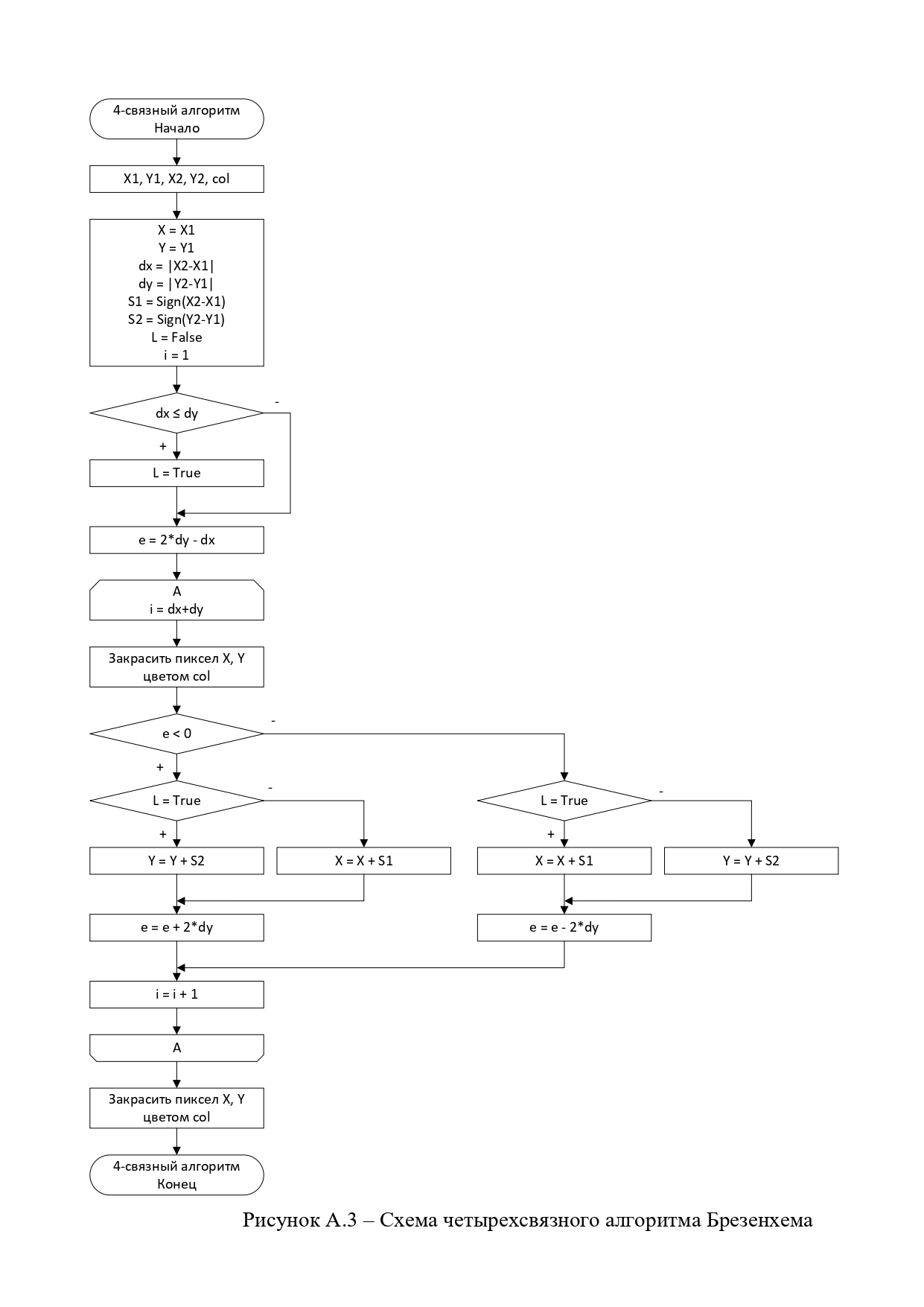


Рисунок А.1 – Схема простого алгоритма





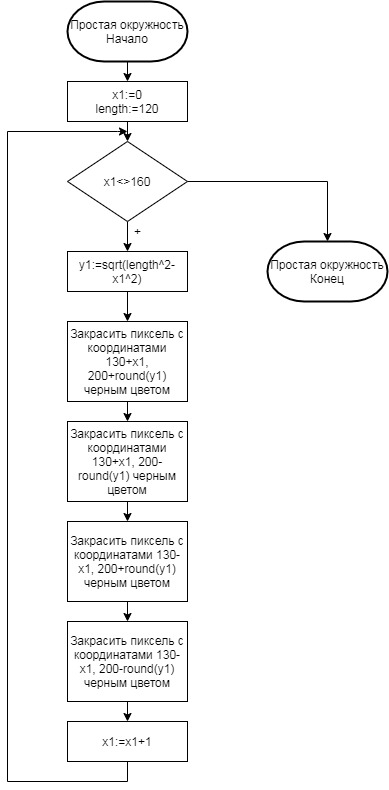


Рисунок А.4 – Схема алгоритма простой окружности

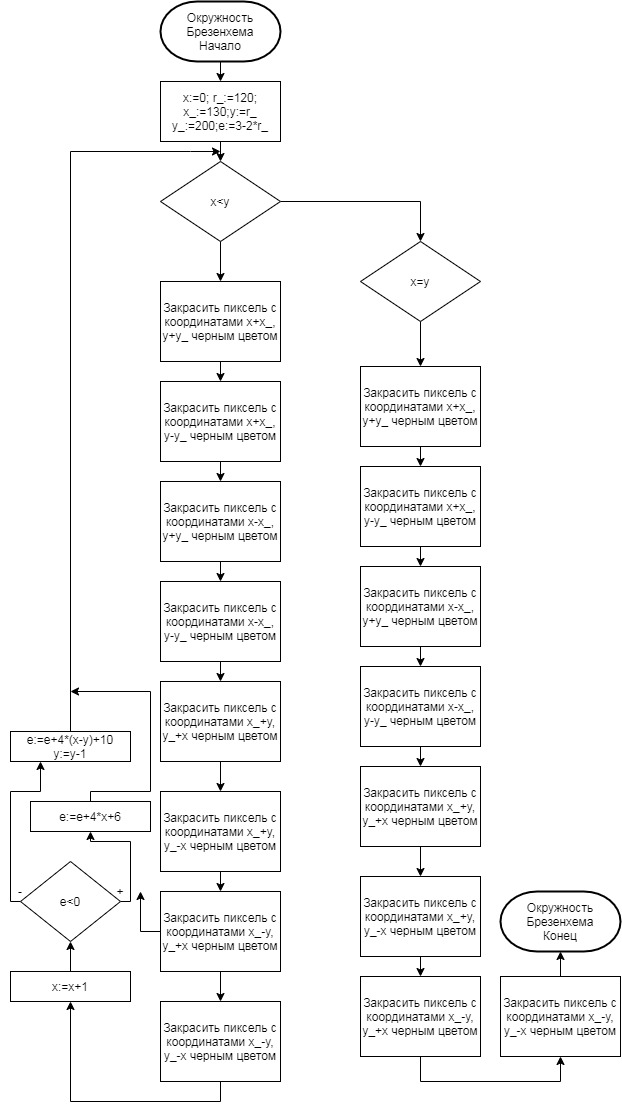


Рисунок А.5 – схема алгоритма окружности Брезенхема

Приложение Б

Листинг процедур

Procedure DrawSimLine(x1,y1,x2,y2:integer;cl:Tcolor);

var l:integer;

incx,incy,cx,cy:real;

begin

if abs(x2-x1) > abs(y2-y1) then l:=abs(x2-x1)

else l:= abs(y2-y1);

incx:=(x2-x1)/l;

incy:=(y2-y1)/l;

cx:=x1;

cy:=y1;

if (x1=x2) or (y1=y2) then

while not((cy=y2) and (cx=x2)) do begin

Form1.Board.canvas.Pixels[round(cx),round(cy)]:=cl;

cx:=cx+incx;

cy:=cy+incy;

end

else while not((cy=y2) or (cx=x2)) do begin

Form1.Board.canvas.Pixels[round(cx),round(cy)]:=cl;

cx:=cx+incx;

cy:=cy+incy;

end;

end;

procedure DrawBrez4(x1,y1,x2,y2:integer;cl:TColor);

var dx,dy,c,e:integer;

x,y,s2,s1,i:integer;

l:boolean;

begin

dx:=abs(x2-x1);

dy:=abs(y2-y1);

x:=x1; y:=y1;

if (x2-x1>=0) then s1:=1

else s1:=-1;

if (y2-y1>=0) then s2:=1

else s2:=-1;

// s2:=sign(y2-y1);

if dx>dy then l:=false

else begin

c:=dx;

dx:=dy;

dy:=c;

l:=true;

end;

e:=2\*dy-dx;

for i:=1 to (dx+dy) do begin

Form1.Board.Canvas.Pixels[x,y]:=cl;

if e<0 then begin

if l=true then y:=y+s2

else x:=x+s1;

e:=e+2\*dy;

end

else begin

if l=true then x:=x+s1

else y:=y+s2;

e:=e-2\*dx;

end;

end;

Form1.Board.Canvas.Pixels[x,y]:=10;

end;

procedure DrawBrez8(x1,y1,x2,y2:integer;cl:TColor);

var dx,dy,c,e:integer;

x,y,s2,s1,i:integer;

l:boolean;

begin

dx:=abs(x2-x1);

dy:=abs(y2-y1);

x:=x1; y:=y1;

if (x2-x1>=0) then s1:=1

else s1:=-1;

if (y2-y1>=0) then s2:=1

else s2:=-1;

// s2:=sign(y2-y1);

if dx<dy then begin

c:=dx;

dx:=dy;

dy:=c;

l:=true;

end

else l:=false;

e:=2\*dy-dx;

for i:=1 to dx do begin

Form1.Board.Canvas.Pixels[x,y]:=10;

while e>=0 do begin

if l=true then x:=x+s1

else y:=y+s2;

e:=e-2\*dx;

end;

if l=true then y:=y+s2

else x:=x+s1;

e:=e+2\*dy;

end;

Form1.Board.Canvas.Pixels[x,y]:=cl;

end;

**procedure** TForm1.BrezCircClick(Sender: TObject);

**var** x,y,x\_,r\_,y\_,e:integer;

**begin**

x:=0;

r\_:=120;

x\_:=130;

y\_:=200;

y:=r\_;

e:=3-2\*r\_;

**while** ( x<y ) **do begin**

Form1.Board.Canvas.Pixels[x\_+x,y\_+y]:=10;

Form1.Board.Canvas.Pixels[x\_+x,y\_-y]:=10;

Form1.board.Canvas.Pixels[x\_-x,y\_+y]:=10;

Form1.Board.Canvas.Pixels[x\_-x,y\_-y]:=10;

Form1.Board.Canvas.Pixels[x\_+y,y\_+x]:=10;

Form1.Board.Canvas.Pixels[x\_+y,y\_-x]:=10;

Form1.Board.Canvas.Pixels[x\_-y,y\_+x]:=10;

Form1.Board.Canvas.Pixels[x\_-y,y\_-x]:=10;

x:=x+1;

**if** e<0 **then** e:=e+4\*x+6

**else begin**

e:=e+4\*(x-y)+10;

y:=y-1;

**end**;

**end**;

**if** x=y **then begin**

Form1.Board.Canvas.Pixels[x\_+x,y\_+y]:=10;

Form1.Board.Canvas.Pixels[x\_+x,y\_-y]:=10;

Form1.board.Canvas.Pixels[x\_-x,y\_+y]:=10;

Form1.Board.Canvas.Pixels[x\_-x,y\_-y]:=10;

Form1.Board.Canvas.Pixels[x\_+y,y\_+x]:=10;

Form1.Board.Canvas.Pixels[x\_+y,y\_-x]:=10;

Form1.Board.Canvas.Pixels[x\_-y,y\_+x]:=10;

Form1.Board.Canvas.Pixels[x\_-y,y\_-x]:=10;

**end**;

**end**;

**procedure** TForm1.SimpCircClick(Sender: TObject);

**var** y1:real;

length,x1:integer;

**begin**

x1:=0;

length:=120;

**while** x1<>160 **do begin**

y1:=sqrt(length\*length-x1\*x1);

Board.Canvas.Pixels[130+x1,200+round(y1)]:=10;

Board.Canvas.Pixels[130+x1,200-round(y1)]:=10;

Board.Canvas.Pixels[130-x1,200+round(y1)]:=10;

Board.Canvas.Pixels[130-x1,200-round(y1)]:=10;

x1:=x1+1;

**end**;

**end**;

Приложение В

Экранные формы

