**Rīgas Tehniskā universitāte**

**Datorzinātnes un informācijas tehnoloģijas fakultāte**

Datorvadības, automātikas un datortehnikas institūts



**Atskaite par V. un VI. praktisko darbu**

priekšmeta "Scēnu analīze"

**Izstrādāja: Iļja Ginters**

**Pārbaudīja: O. Krutikova**

**2012./2013.m.g.**

**Teorijas apraksts:**

*Kukaiņa algoritms*

Kontūru izsekošana atgadina kukaini, kurš apiet šķēršļus. Gadījumā, ja attēls ir binārs, tad kukainis sāk savu ceļu uz baltā lauka un kustās pa labi, ja nākamais pikselis ir melnā krāsā, tad kukainis pagriežas pa kreisi un kustās pie nākama pikseļa. Šīs darbības turpinās kamēr kukainis neatgriežas sākumpunktā.

*Kukaiņa algoritms ar atgriešanos (soļu saglabāšanu)*

Šajā algoritmā, ja kukainis taisa pārēju no baltā uz melnu pikseli, tas atgriežas uz sākumpunktu un taisa pagriezienu pa labi. Kukainis taisa pagriezienu pa labi, vienmēr, kad tas taisa pārēju no baltā uz balto pikseli. Tādēļ ka iepriekšējie soļi tiek saglabāti, šis algoritms taisa divreiz vairāk darbību (lēnāk strādā).

**Programmas pirmkods ar komentāriem**

// Virziena tips

type

Direction = record

direct :Integer; // uz kuru pusi pagriežas

x,y :Integer; // jaunie koordināti

end;

// Procedūras, kuras apstrādā kukaiņa pagriešanu

procedure TMain\_menu.TurnRight(var dir:Direction); // pa labi

begin

Case dir.direct of

1: begin // UP

dir.direct := 2;

dir.x := dir.x + 1;

end;

2: begin // RIGHT

dir.direct := 4;

dir.y := dir.y + 1;

end;

3: begin // LEFT

dir.direct := 1;

dir.y := dir.y - 1;

end;

4: begin // DOWN

dir.direct := 3;

dir.x := dir.x - 1;

end;

else label2.Caption:='Direction is Unknown!';

end;

end;

procedure TMain\_menu.TurnLeft(var dir:Direction); // pa kreisi

begin

Case dir.direct of

1: begin // UP

dir.direct := 3;

dir.x := dir.x - 1;

end;

2: begin // RIGHT

dir.direct := 1;

dir.y := dir.y - 1;

end;

3: begin // LEFT

dir.direct := 4;

dir.y := dir.y + 1;

end;

4: begin // DOWN

dir.direct := 2;

dir.x := dir.x + 1;

end;

else label2.Caption:='Direction is Unknown!';

end;

end;

procedure TMain\_menu.TurnBack(var dir:Direction); // atgriežas uz sākumpunktu

begin

Case dir.direct of

1: begin // UP

dir.direct := 4;

dir.y := dir.y + 1;

end;

2: begin // RIGHT

dir.direct := 3;

dir.x := dir.x - 1;

end;

3: begin // LEFT

dir.direct := 2;

dir.x := dir.x + 1;

end;

4: begin // DOWN

dir.direct := 1;

dir.y := dir.y - 1;

end;

else label2.Caption:='Direction is Unknown!';

end;

end;

// Procedūras, kuras apstrādā kukaiņa pagriešanu

procedure TMain\_menu.Kukainis();

Var X,Y :Integer; // objekta pirmie koordināti

i,j, ind, maxX, maxY, minX, minY, cc :Integer;

B :Byte; // apskatāma pikseļa intensitāte

dir :Direction; // virziens un tekošie koordināti

begin

Image1.Picture.Bitmap.Width:=Length(img);

Image1.Picture.Bitmap.Height:=Length(img[0]);

Image1.Picture.Bitmap.PixelFormat:=pf24bit;

minX := 9999; minY := 9999;

maxX := 0; maxY := 0;

cc := 0;

for j:=0 to High(img[0])-1 do // cikla sakums

for i:=0 to High(img)-1 do

begin

B := img[i,j].R;

if (((B = 0) and (img[i,j].segment=0)) // neapstrādam jau apskatītus pikseļus

and (((i > maxX) or (i < minX)) and ((j < minY) or (j > maxY))) ) then

begin // kukaiņa algoritma sakums

X := i;

Y := j;

img[X,Y].segment := -1;

dir.direct := 1;

dir.x := X;

dir.y := Y-1;

minX := 9999; minY := 9999;

maxX := 0; maxY := 0;

while ((dir.x <> X) or (dir.y <> Y)) do

begin

B := img[dir.x, dir.y].R;

if (B = 0) then

begin

if (dir.x > maxX) then maxX := dir.x;

if (dir.y > maxY) then maxY := dir.y;

if (dir.x < minX) then minX := dir.x;

if (dir.y < minY) then minY := dir.y;

img[dir.x, dir.y].segment := -1;

if (CheckBox1.Checked = true) then // vai pielietot algoritmu ar atgriešanos?

begin

TurnBack(dir);

TurnRight(dir);

end else TurnLeft(dir);

end else

begin

TurnRight(dir); // pagriezties pa labi

end;

end;

cc:= cc+1; // palielināt objektu skaits

end; //if

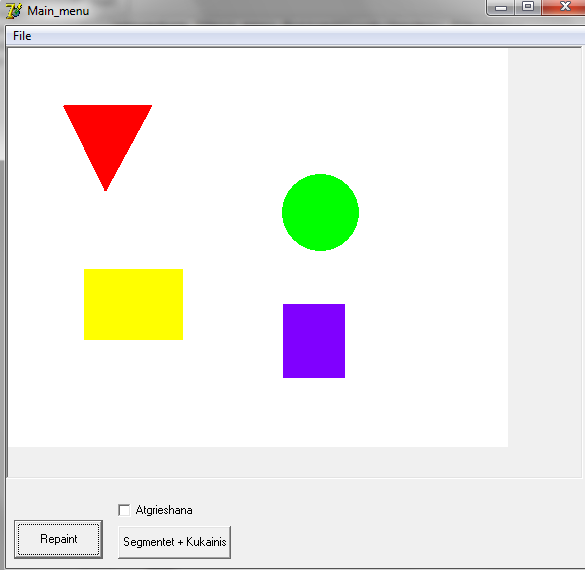
end; //for

Label2.Caption:='Objektu skaits: '+IntToStr(cc);

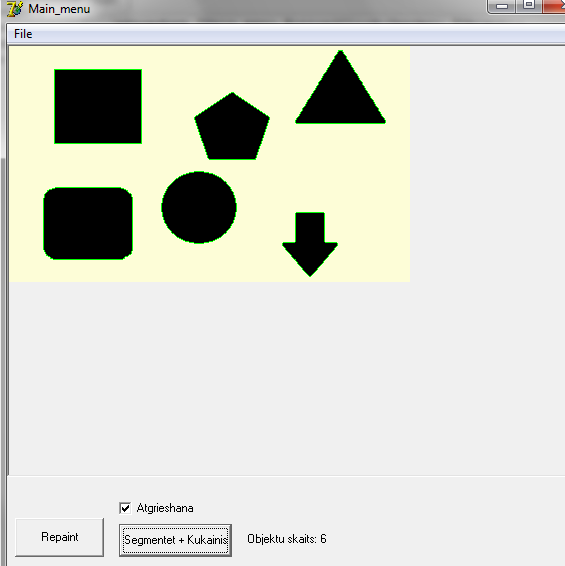
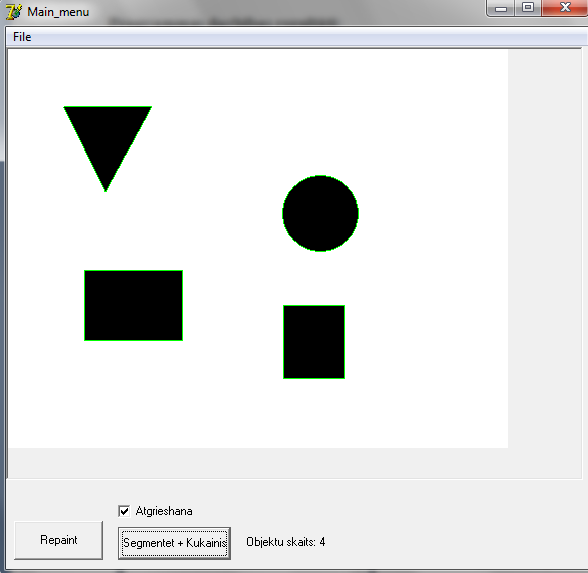
segmentToImage(); // attēlot rezultātu

end;

**Programmas darbības rezultāti:**

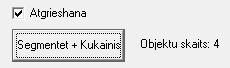


Attēls 1. Galvenās programmas logs



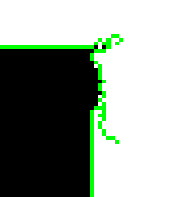
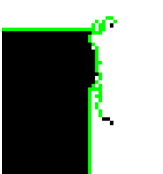
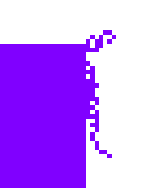
Attēls 2. un 3. Kukaiņa algoritma darbības rezultāti

Kukaiņa procedūra palaižas kad attēls tiek apstrādāts ar segmentācijas algoritmu (Krāsainie objekti tiek atrasti un atzīmēti ar melno krāsu). Varām redzēt ka algoritms sekmīgi iezīmēja kontūrus dažāda forma objektiem.



Attēls 4. Kukaiņa algoritma tipa izvēle

Kukaiņa algoritma tipu varam izvēlēties atzīmējot punktu "Atgrieshana". Ja nospiest pogu "Segmentet + Kukainis" ar atzīmēto "Atgrieshana", tad palaižas kukaiņa algoritms ar atgriešanu, pretēja gadījumā parastais kukaiņa algoritms.



Attēls 5. Algoritmu salīdzinājums

Uz attēliem tiek paradīta objekta daļa ar sarežģītu formu, un kā atšķiras darbības rezultāti parasta kukaiņa algoritma (attēls centrā) no algoritma ar atgriešanu (attēls pa labi). Ir redzams ka uzlabotais algoritms atzīmēja vairāk pikseļu veidojot kontūru.

**Secinājūmi:**

Tiek izstrādāts laboratorijas darbs, kura laika tika realizēti kukaiņa algoritms un tā uzlabota versija. Pēc darbības rezultāta salīdzinājuma ir skaidrs, ka algoritms ar atgriešanu dod labākus rezultātus, bet tas trūkums - darbības ātrums (darbību skaits ir aptuveni divreiz lielāks, nekā parastam algoritmam). Kādu algoritmu labāk pielietot ir atkarīgs no uzdevuma, tas laika un kvalitātes prasībām. Kopumā laboratorijas darbs tika izpildīts sekmīgi un algoritmi strādā pareizi.