LAMPIRAN A

PROGRAM ANIMASI KARTUN

Program Animasi Iklan Kartun

```
Program Animasi_Iklan_Kartun;
USES
Crt, vgacad,rgraph,DOS;
var
    Back                                : Pointer;
    Size                                : Word;
    x1,x2,x3,x4,x5                      : Integer;
    y1,y2,y3,y4,y5                      : Integer;
    teta,ax5,ay5                        : integer;
    angle,angle_v,scale                 : Real;
    n                                   : Integer;

process : array [1..5] of pointer;
pict    : array [1..5] of pointer;



Procedure Initialize;
var i : integer;
  Begin
    getmem(Back,64000);
    for i:= 1 to 4 do  getmem(process[i],1);
    getmem(process[5],64000);
  End;



Procedure Hapus_All_mem;
begin
  freemem(Back,64000);
  for i:= 1 to 4 do  freemem(process[i],1);
  freemem(process[5],64000);
  freeMem(Pict[1],ImageSize(1,1,20,75));
  freeMem(Pict[2],ImageSize(301,1,320,75));
  freeMem(Pict[3],ImageSize(1,164,59,198));
  freeMem(Pict[4],ImageSize(110,1,210,39));
  freeMem(Pict[5],ImageSize(300,120,320,198));
end;
```

```
Procedure Generate_Picture;
var i : integer;
Begin
    Load_Picture('backb.Pic',Back^);
    for i:=1 to 5 do
    begin
         Load_picture('benda.Pic',Process[i]^);
         case i of
{tinta}    1:begin
             GetMem(Pict[i],ImageSize(1,1,20,75));
             GetImage(Process[i]^,1,1,20,75,Pict[i]^);
             end;
{pensil}   2:begin
             GetMem(Pict[i],ImageSize(301,1,320,75));
             GetImage(Process[i]^,301,1,320,75,Pict[i]^);
             end;
{jangka}   3:begin
             GetMem(Pict[i],ImageSize(300,120,320,198));
             GetImage(Process[i]^,300,120,320,198,Pict[i]^;
             end;
{penghapus}4:begin
             GetMem(Pict[i],ImageSize(1,164,59,198));
             GetImage(Process[i]^,1,164,59,198,Pict[i]^);
             end;
{logo}     5:begin
             GetMem(Pict[i],ImageSize(110,1,210,39));
             GetImage(Process[i]^,110,1,210,39,Pict[i]^);
             end;
         end;
    end;
End;


Procedure Main;
  var     i:integer;
  Begin
        y1:=95;
        for x1:=1 to 100 do
        begin
        MoveTo(Back^,Process[1]^);
        PutImage(Process[1]^,x1,y1,Pict[1]^);
        MoveTo(Process[1]^,Screen^);
        delay(20);
        end;
        PutImage(back^,x1,y1,Pict[1]^);
```

```
x2:=150;
for y2:=120 downto 95 do
begin
MoveTo(back^,Process[2]^);
PutImage(Process[2]^,x2,y2,Pict[2]^);
MoveTo(Process[2]^,Screen^);
delay(20);
end;
PutImage(Back^,x2,y2,Pict[2]^);

y3:=90;
for x3:=303 downto 200 do
begin
MoveTo(back^,Process[3]^);
PutImage(Process[3]^,x3,y3,Pict[3]^);
MoveTo(Process[3]^,Screen^);
delay(20);
end;

PutImage(back^,x3,y3,Pict[3]^);
x4:= 130;
for y4:= 1 to 60 do
begin
MoveTo(Back^,Process[4]^);
PutImage(Process[4]^,x4,y4,Pict[4]^);
MoveTo(Process[4]^,Screen^);
delay(20);
end;

PutImage(Back^,x4,y4,Pict[4]^);
y5:=160;
for x5:=310 downto 100 do
begin
MoveTo(back^,Process[5]^);
PutImage(Process[5]^,x5,y5,Pict[5]^);
MoveTo(Process[5]^,Screen^);
delay(20);
end;

i:=0;ay5:=38;
Repeat
for ax5:= 100 to 140 do
begin
MoveTo(Back^,Process[5]^);
Put3d(Process[5]^,x5,y5,ax5,ay5,Pict[5]^)
```

```
        MoveTo(Process[5]^,Screen^);
        delay(20);
        end;

        for ax5:= 140 downto 100 do
        begin
        MoveTo(Back^,Process[4]^);
        Put3d(Process[5]^,x5,y5,ax5,ay5,Pict[5]^);
        MoveTo(Process[5]^,Screen^);
        delay(20);
        end;
        inc(i);
        Until i=4;
  End;


Begin
ClrScr;
GetMem(texture, 64000);
GetMem(composite, 64000);
openvga256;
loadimage('logo.bmp');
clearcomposite;

angle:= PI/360;
angle_v:=-PI/180;
scale:=0.50;
i:=1;
while i < 360 do
Begin
     i:=i+1;
     FastRotate(scale,angle);
     copycomposite;
     angle:=angle+angle_v;
End;
FreeMem(composite,64000);
FreeMem(texture,64000);
 Install_Palette('backb.Pal');
  Initialize;
  Generate_Picture;
  Main;
  Active_To_Palette(Default_Pal,50);
  Set_All_Palette(User_Pal,255,0,0);
  Active_To_Palette(User_Pal,50);
  Hapus_all_Mem;
closevga256;
End
```

Program Animasi Kartun Mobil

```
Program Animasi_Kartun_mobil;

USES
Crt, vgacad,DOS;
var
    Back                                : Pointer;
    Size                                : Word;
    x1,x2,x3,x4,x5                      : Integer;
    y1,y2,y3,y4,y5                      : Integer;

process : array [1..5] of pointer;
pict    : array [1..5] of pointer;


Procedure Initialize;
var i : integer;
  Begin
    getmem(Back,64000);
    for i:= 1 to 4 do  getmem(process[i],1);
    getmem(process[5],64000);
  End;



Procedure Hapus_All_mem;
var i : integer;
begin
  freemem(Back,64000);
  for i:= 1 to 4 do  freemem(process[i],1);
  freemem(process[5],64000);
  freeMem(Pict[1],ImageSize(1,155,30,200));
  freeMem(Pict[2],ImageSize(1,1,30,30));
  freeMem(Pict[3],ImageSize(290,1,320,50));
  freeMem(Pict[4],ImageSize(290,170,319,199));
end;


Procedure Generate_Picture;
var i : integer;
Begin
    Load_Picture('sirl.Pic',Back^);
    for i:=1 to 4 do
```

```
   begin
         Load_picture('molek.Pic',Process[i]^);
         case i of
{mob1}   1:begin
           GetMem(Pict[i],ImageSize(1,155,30,200));
           GetImage(Process[i]^,1,157,30,199,Pict[i]^);
           end;
{mob2}   2:begin
           GetMem(Pict[i],ImageSize(1,1,30,30));
           GetImage(Process[i]^,1,1,30,30,Pict[i]^);
           end;
{mob3}   3:begin
           GetMem(Pict[i],ImageSize(290,1,320,50));
           GetImage(Process[i]^,290,1,320,50,Pict[i]^);
           end;
{mob4}   4:begin
           GetMem(Pict[i],ImageSize(290,170,319,199));
           GetImage(Process[i]^,290,170,319,199,Pict[i]^);
           end;
         end;
    end;
End;


Procedure Main;
  var     i:integer;
  Begin
        i:=0;
        repeat
        x1:=32;
        for y1:=140 downto 1 do
        begin
        MoveTo(back^,Process[1]^);
        PutImage(Process[1]^,x1,y1,Pict[1]^);
        MoveTo(Process[1]^,Screen^);
        delay(10);
        end;

        y2:=13;
        for x2:=20 to 258 do
        begin
        MoveTo(Back^,Process[2]^);
        PutImage(Process[2]^,x2,y2,Pict[2]^);
        MoveTo(Process[2]^,Screen^);
        delay(10);
        end;
```

```
        x3:= 259;
        for y3:= 1 to 160 do
        begin
        MoveTo(Back^,Process[3]^);
        PutImage(Process[3]^,x3,y3,Pict[3]^);
        MoveTo(Process[3]^,Screen^);
        delay(10);
        end;

        y4:=158;
        for x4:=259 downto 32 do
        begin
        MoveTo(back^,Process[4]^);
        PutImage(Process[4]^,x4,y4,Pict[4]^);
        MoveTo(Process[4]^,Screen^);
        delay(10);
        end;
        inc(i);
        Until i=2;
  End;


Begin
ClrScr;
openvga256;
install_palette('sirl.pal');
  Initialize;
  Generate_Picture;
  Main;
  Active_To_Palette(Default_Pal,50);
  Set_All_Palette(User_Pal,255,0,0);
  Active_To_Palette(User_Pal,50);
Hapus_all_Mem;
closevga256;
End.
```

LAMPIRAN B

UNIT YANG DIGUNAKAN

Unit VGACAD

```
{-------------------------------------------------}
{ Program : VGACAD.PAS                            }
{ Coder   : Agustinus Nalwan                      }
{ Fungsi  : Unit Animasi Dan Game Profesional     }
{           Versi yang telah dimodifikasi         }
{           Versi 2.00                            }
{-------------------------------------------------}

{$G+}
Unit VGACAD;

INTERFACE

Uses Crt,Dos,VgaFont;

Const Screen      : Pointer = Ptr ($A000,0);

Type Palette_Cell = Record
       R , G , B : Byte;
     End;

     Palette_Type = Array[0..255] of Palette_Cell;
     Palette_Addr = ^Palette_Type;

     Scroll_Page  = Array[1..800] of Pointer;

     Parallax_Data = Record
       Page_X, Page_Y : Integer;
       Last_X, Last_Y : Integer;
       Length, Width  : Integer;
     End;

Var
     A,B             :  Integer;
     Reg             :  Registers;
     Total_Page      :  Byte;
     File_P          :  File;
     UserPalette     :  Palette_Type;
     DefaultPalette:  Palette_Type;
     File_Pal        :  File;
     User_Pal        :  Palette_Addr;
     Default_Pal     :  Palette_Addr;
     Active_Pal      :  Palette_Addr;
     Page_Scroll     :  Scroll_Page;
     Parallax_Dat    :  Parallax_Data;
     Page_Width      :  Integer;
     X,Y             :  Word;
     Px2,Py2         :  Word;
     Px3,Py3         :  Word;
     Already         :  Boolean;
```

```
{ Procedure Inisialisasi Graphic }

Procedure OpenVga256;
Procedure CloseVga256;

{ Procedure Manipulasi Layar }


Procedure Define_Page   (Var PicBuff : Pointer);
Procedure UnDefine_Page (Var PicBuff : Pointer);
Procedure Put3D(Var PicBuff;X,Y : Integer;XL,YL:Word; Var Buffer);
Procedure MoveTo        (Var Source, Destination);


{ Procedure Manipulasi Sprite }

Function  ImageSize     (X1,Y1,X2,Y2 : Integer) : Word;
Procedure GetImage      (Var PicBuff; X1,Y1,X2,Y2 : Integer; Var
Buffer);
Procedure Load_Sprite(File_Name:String;Var Ptr:Pointer;Var Size:Word);
Procedure Load_Picture  (File_Name:String; Var Buffer);


Procedure PutImage      (Var PicBuff;X1,Y1:Integer;Var Buffer);



{ Procedure Manipulasi Palette }

Procedure Set_All_Palette  (Pal_Buff:Palette_Addr; R,G,B : Byte);
Procedure Active_To_Palette(Pal_Dest:Palette_Addr; Del:Byte);
Procedure Install_Palette(Name:String);



Implementation


Procedure GetImage (Var PicBuff; X1,Y1,X2,Y2 : Integer; Var Buffer);
ASSEMBLER;
  Asm
    Push ds          { Push Stack                 }
    Lds si,PicBuff   { Arahkan DS:SI ke PICBUFF   }
    Les di,Buffer    { Arahkan ES:DI ke BUFFER    }
    Mov dx,Y1        { Dx = Y awal                }
    Mov ax,dx        { Ax = Dx                    }
    Shl dx,1
    Shl dx,1         { Dx = Dx * 4                }
    Add dx,ax        { Dx = Dx + Ax               }
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1         { Dx = Dx * 32               }
```

```
    Add dx,X1          { Dx = Dx + X awal           }
    Add si,dx          { Offset = Offset + Dx       }
    Mov bx,X2          { Bx = X akhir               }
    Sub bx,X1          { Bx = Bx - X awal           }
    Inc bx             { Bx = Bx + 1 = Panjang Gambar }
    Mov [es:di],bx     { Tulis Panjang Gambar       }
    Inc di             { Offset = Offset + 1        }
    Inc di             { Offset = Offset + 1        }
    Mov cx,Y2          { Cx = Y akhir               }
    Sub cx,Y1          { Cx = Cx - Y awal           }
    Inc cx             { Cx = Cx + 1 = Lebar Gambar }
    Mov [es:di],cx     { Tulis Lebar Gambar         }
    Inc di             { Offset = Offset + 1        }
    Inc di             { Offset = Offset + 1        }
    Cld                { Counter Cx Mundur          }
    @1:
      Mov dx,cx        { Dx = Sisa lebar gambar     }
      Mov cx,bx        { Cx = Panjang gambar        }
      Mov ax,si        { Ax = Si                    }
      Rep Movsb        { Baca titik sejumlah Cx     }
      Mov si,ax        { Kembalikan Si ke Si awal   }
      Add si,320       { Si ke koordinat Y berikutnya }
      Mov cx,dx        { Cx = Dx                    }
      Loop @1          { Cx-1, Ulang hingga Cx = 0  }
    Pop ds             { Pop Stack                  }
  End;


Procedure OpenVga256;
  Begin
    Reg.Ax:=$13;
    Intr($10,Reg);
  End;


Procedure MoveW(Var Source; Var Dest; Size:Word); ASSEMBLER;
  Asm
    Push ds            { Push Stack                 }
    Les di,Dest        { Arahkan ES:DI ke Dest      }
    Lds si,Source      { Arahkan DS:SI ke Source    }
    Mov Cx,Size        { Counter Cx sebesar Size    }
    Rep MovsW          { Pindahkan DS:SI ke ES:DI   }
    Pop ds             { Pop Stack                  }
  End;



Procedure Load_Sprite(File_Name:String;Var Ptr:Pointer;Var Size:Word);
  Begin
    Assign(File_P,File_Name);
    Reset(File_P,1);
    GetMem(Ptr,FileSize(File_P));
    BlockRead(File_P,Ptr^,FileSize(File_P));
    Size:=FileSize(File_P);
    Close(File_P);
  End;
```

```
Procedure Define_Page(Var PicBuff : Pointer);
  Begin
    GetMem(PicBuff,64000);
  End;


Procedure UnDefine_Page(Var PicBuff : Pointer);
  Begin
    FreeMem(PicBuff,64000);
  End;


Procedure Fill_Page (Var PicBuff; Col : Byte);
  Begin
    FillWord(PicBuff,32000,Col);
  End;


Procedure MoveTo (Var Source, Destination);
  Begin
    MoveW(Source,Destination,32000);
  End;


Procedure Load_Picture (File_Name:String;Var Buffer);
  Begin
    Assign(File_P,File_Name);
    Reset(File_P,1);
    BlockRead(File_P,Buffer,64000);
    Close(File_P);
  End;


Procedure Save_Picture (File_Name:String;Var Buffer);
  Begin
    Assign(File_P,File_Name);
    Rewrite(File_P,1);
    BlockWrite(File_P,Buffer,64000);
    Close(File_P);
  End;

Procedure SetVisualPalette (No,R,G,B:Byte);
  Begin
    Port[$3c8]:=No;       { Port nomor Palette }
    Port[$3c9]:=R;        { Isi Port Merah     }
    Port[$3c9]:=G;        { Isi Port Hijau     }
    Port[$3c9]:=B;        { Isi Port Biru      }
  End;
```

```
Procedure SetPalette    (Var Arr_Pal:Palette_Addr; No,Rr,Gg,Bb:Byte);
  Begin
    With Arr_Pal^[No] do
      Begin
        R:=Rr;
        G:=Gg;
        B:=Bb;
      End;
  End;


Procedure Load_Palette(Pal_Buff:Pointer; File_Name:String);
Var X         : Byte;
    R,G,B     : Char;
    RR,GG,BB  : Byte;
  Begin
    Assign(File_Pal,File_Name);
    Reset(File_Pal,1);
    BlockRead(File_Pal,Pal_Buff^,768);
    Close(File_Pal);
  End;


Procedure Refresh_Palette;
  Begin
    For A:=0 to 255 do
      With Active_Pal^[A] do
        Begin
          SetVisualPalette(A,R,G,B);
        End;
  End;


Procedure Active_Palette(Ac_Pal:Palette_Addr);
  Begin
    MoveW(Ac_Pal^,Active_Pal^,384);
    Refresh_Palette;
  End;


Procedure Install_Palette(Name:String);
  Begin
    GetMem(Default_Pal,768);
    GetMem(User_Pal,768);
    GetMem(Active_Pal,768);
    Load_Palette(Default_Pal,Name);
    Active_Palette(Default_Pal);
  End;

Procedure Set_All_Palette   (Pal_Buff:Palette_Addr; R,G,B : Byte);
  Begin
    For A:=0 to 255 do
      SetPalette(Pal_Buff,A,R,G,B);
  End;
```

```
Procedure Active_To_Palette(Pal_Dest:Palette_Addr; Del:Byte);
Var Done     : Boolean;
    A        : Integer;
  Begin
    Repeat
      Done:=True;
      Delay(Del);
      For A:=0 to 255 do
        With Pal_Dest^[A] do
          Begin
            If Active_Pal^[A].R<R then Inc(Active_Pal^[A].R);
            If Active_Pal^[A].R>R then Dec(Active_Pal^[A].R);
            If Active_Pal^[A].G<G then Inc(Active_Pal^[A].G);
            If Active_Pal^[A].G>G then Dec(Active_Pal^[A].G);
            If Active_Pal^[A].B<B then Inc(Active_Pal^[A].B);
            If Active_Pal^[A].B>B then Dec(Active_Pal^[A].B);
            If Active_Pal^[A].R<>R then Done:=False;
            If Active_Pal^[A].G<>G then Done:=False;
            If Active_Pal^[A].B<>B then Done:=False;
          End;

      Refresh_Palette;
    Until Done;
  End;


Procedure PutImage(Var PicBuff;X1,Y1:Integer;Var Buffer);Assembler;
Var XWStart:Word;
    XRStart:Word;
    YWStart:Word;
    YRStart:Word;
    Length :Word;
    Width  :Word;
    XRead  :Word;  { CX }
    YRead  :Word;
  Asm
    Push ds
    Lds si,Buffer
    Les di,PicBuff
    Mov Ax,[ds:si]
    Mov Length,Ax
    Mov XRead,Ax
    Add si,2
    Mov Ax,[ds:si]
    Mov Width,Ax
    Mov YRead,Ax
    Mov YRStart,0
    Mov Ax,Y1
    Mov YWStart,Ax
    Add si,2
    Mov XRStart,0
    Mov Ax,X1
    Mov XWStart,Ax
    Cmp Ax,0
    Jge @X1_Not_Negative
      Neg Ax
```

```
  Mov XRStart,Ax
  Mov XWStart,0
  Sub XRead,Ax
@X1_Not_Negative:
  Mov Ax,X1
  Add Ax,Length
  Dec Ax
  Cmp Ax,319
Jle @X1_No_Right_Crop
  Sub Ax,319
  Sub XRead,Ax
@X1_No_Right_Crop:
  Mov Ax,Y1
  Cmp Ax,0
Jge @No_Top_Crop
  Neg Ax
  Mov YWStart,0
  Mov YRStart,Ax
  Sub YRead,Ax
@No_Top_Crop:
  Mov Ax,Y1
  Add Ax,Width
  Dec Ax
  Cmp Ax,199
Jle @No_Bot_Crop
  Sub Ax,199
  Sub YRead,Ax
@No_Bot_Crop:
  Add di,XWStart
  Mov Ax,YWStart
  Shl Ax,2
  Add Ax,YWStart
  Shl Ax,6
  Add di,Ax

  Mov Ax,YRStart
  Xor Ah,Ah
  Mul Length
  Add si,Ax
  Add si,XRStart

  Mov Cx,YRead
@Loop1:
  Push Cx
  Push di
  Push si
  Mov Cx,XRead
  @Loop2:
     Lodsb
     or al,al
     Jz @No_Write
     Stosb
     dec di
   @No_Write:
     inc di
  Loop @Loop2
```

```
      Pop si
      Pop di
      Add si,Length
      Add di,320
      Pop Cx
      Loop @Loop1
    Pop ds
  End;


Procedure Put3D(Var PicBuff;X,Y : Integer;XL,YL:Word; Var Buffer);
Var XWStart:Word;
    YWStart:Word;
    Length :Word;
    Width  :Word;
    XRead  :Word;  { CX }
    YRead  :Word;

    XFixed :Word;
    YFixed :Word;
    XFStart:Word;
    YFStart:Word;
    RX     :Word;
    RY     :Word;
  Begin
  Asm
    Push ds
    Lds si,Buffer
    Les di,PicBuff
    Mov Ax,[ds:si]
    Mov Length,Ax
    Mov Ah,Al
    Xor Al,Al
    Mov Bx,Ax
    Xor Dx,Dx
    Div XL
    Mov Rx,Ax
    Mov Ax,Bx
    Xor Dx,Dx
    Div Rx
    Mov XRead,Ax
    Add si,2
    Mov Ax,[ds:si]
    Mov Width,Ax
    Mov Ah,Al
    Xor Al,Al
    Mov Bx,Ax
    Xor Dx,Dx
    Div YL
    Mov Ry,Ax
    Mov Ax,Bx
    Xor Dx,Dx
    Div Ry
    Mov YRead,Ax
    Mov XFixed,0
```

```
Mov YFixed,0
Mov XFStart,0
Mov YFStart,0
Mov Ax,Y
Mov YWStart,Ax
Add si,2
Mov Ax,X
Mov XWStart,Ax
Cmp Ax,0
Jge @X1_Not_Negative
  Neg Ax
  Mov XWStart,0
  Sub XRead,Ax
  Mul Rx
  Mov XFStart,Ax
@X1_Not_Negative:
  Mov Ax,XWStart
  Add Ax,XRead
  Dec Ax
  Cmp Ax,319
Jle @X1_No_Right_Crop
  Sub Ax,319
  Sub XRead,Ax
@X1_No_Right_Crop:
  Mov Ax,Y
  Cmp Ax,0
Jge @No_Top_Crop
  Neg Ax
  Mov YWStart,0
  Sub YRead,Ax
  Mul Ry
  Mov YFStart,Ax
@No_Top_Crop:
  Mov Ax,YWStart
  Add Ax,YRead
  Cmp Ax,200
Jle @No_Bot_Crop
  Sub Ax,200
  Sub YRead,Ax
@No_Bot_Crop:
  Add di,XWStart
  Mov Ax,YWStart
  Shl Ax,2
  Add Ax,YWStart
  Shl Ax,6
  Add di,Ax

  Mov Cx,YRead
  Mov Ax,YFStart
  Mov YFixed,Ax
@Loop1:
  Push Cx
  Push di
  Push si
  Mov Ax,YFixed
  Mov Al,Ah
```

```
        Xor Ah,Ah
        Mul Length
        Add si,Ax

        Mov Cx,XRead
        Mov Ax,XFStart
        Mov XFixed,Ax
        Mov Bx,Rx
        @Loop2:
           Mov Ax,XFixed
           Mov Al,Ah
           Xor Ah,Ah
           Push si
           Add si,Ax

           Lodsb
           Pop si
           Add XFixed,Bx
           or al,al
           Jz @No_Write
           Stosb
           dec di
         @No_Write:
           inc di
        Loop @Loop2
        Pop si
        Pop di
        Mov Bx,Ry
        Add YFixed,Bx
        Add di,320
        Pop Cx
        Loop @Loop1
      Pop ds
    End;
    End;
        Jnz @Loop2
        Mov bl,al
        Lodsb
        Mov cl,al
        Mov al,bl
        @Loop2:
          Stosb
        Loop @Loop2
        Pop Cx
      Loop @Loop1
      Pop ds
    End;
    End;

  Procedure Rll_Compress(Var Source;Var Destination;Var Size:Word);
  Var SegS,OffS:Word;
      SegD,OffD:Word;
      Finish:Boolean;
      SrcCtr:Byte;
      FByte :Byte;
      Ctr   :Word;
```

```
  OldOFF:Word;
  Length,Width:Word;
Begin
  Ctr:=1;
  GetAddr(Source,SegS,OffS);
  GetAddr(Destination,SegD,OffD);
  Move(Mem[SegS:OffS],Length,2);
  Move(Mem[SegS:OffS+2],Width,2);
  Size:=Length*Width+4;
  OldOFF:=OffD;
  Finish:=False;
  FByte:=Mem[SegS:OffS];
  Repeat
    SrcCtr:=1;
    While (FByte=Mem[SegS:OffS+SrcCtr]) and (SrcCtr<255) and
    (Ctr<=Size) do
      Begin
        Inc(SrcCtr);
        Inc(Ctr);
      End;
    If SrcCtr>1 then
      Begin
        Mem[SegD:OffD]:=SrcCtr;
        Inc(OffD);
        Mem[SegD:OffD]:=FByte;
        Inc(OffD);
        Inc(OffS,SrcCtr);
        FByte:=Mem[SegS:OffS];
        Inc(Ctr);
      End Else
      Begin
        Mem[SegD:OffD]:=0;
        SrcCtr:=0;
        While(Mem[SegS:OffS+SrcCtr]<>Mem[SegS:OffS+SrcCtr+1])
        and
        (SrcCtr<255) and (Ctr<=Size) do
          Begin
            Inc(Ctr);
            Mem[SegD:OffD+SrcCtr+2]:=Mem[SegS:OffS+SrcCtr];
            Inc(SrcCtr);
          End;
        Mem[SegD:OffD+1]:=SrcCtr;
        Inc(OffD,SrcCtr+2);
        Inc(OffS,SrcCtr);
        FByte:=Mem[SegS:OffS];
      End;
    If Ctr>Size then Finish:=True;
  Until Finish;
  Mem[SegD:OffD]:=0;
  Mem[SegD:OffD+1]:=0;
  Size:=OffD+2-OldOFF;
End;
```

```
Procedure PutStr (Var PicBuff;X,Y:Integer;Col1,Col2:Byte;Str:String);
Var A,Start,Stop:Integer;
  Begin
    Start:=0;
    For A:=1 to Length(Str) do
      Begin
        If (A-1)*6+X<0 then Start:=A;
      End;
    If Start<>0 then
      For A:=1 to Start do
        Str[A]:=' ';
    Stop:=0;
    For A:=Length(Str) downto 1 do
      If ((A-1)*6)+X>313 then Stop:=A;
    If Stop<>0 then
      For A:=Length(Str) downto Stop do
        Str[A]:=' ';
    If (Y<192) and (Y>=0) then
      PutString(PicBuff,X,Y,Col1,Col2,Str);
  End;


Procedure RLL_UnCompress (Var Source,Destination); ASSEMBLER;
asm
  Mov dx,ds              { save ds }
{  Les di,Destination
  Lds si,Source
  Xor bx,bx
  Cld

@1:
  Xor cx,cx
  Lodsb
  Or  al,al
  Jz  @2
  Mov cl,al
  Add bx,cx
  Lodsb
  Rep Stosb
  Loop @1

@2:
  Lodsb
  Or  al,al
  Jz  @3
  Mov cl,al
  Add bx,cx
  Rep Movsb
  Loop @1

@3:
  Mov ds,dx
end; { RLLxUnCompress }
Function ImageSize (X1,Y1,X2,Y2 : INTEGER) : Word ;
  Begin
    ImageSize := (X2-X1+1) * (Y2-Y1+1) + 4 ;
```

```
  End;
Procedure CloseVga256;
  Begin
    TextMode(80);
  End;
Begin
  Randomize;
  Already:=False;
End.
```

Unit Rgraph

```
Unit rgraph;
interface

{$R-}
{$X+} { use FPU }
{$M 16384,196608,196608} {probably too much... but tired of crashes
while experimenting }

USES
   Crt, DOS;

const
      {screen constants}
      WIDTH = 320;
      HEIGHT = 200;
      SCREENSIZE = WIDTH*HEIGHT;
      PALETTESIZE = 256*3;
      VGA : Word = $a000;
      FAILURE = 0;
      SUCCESS = 1;
      PI = 3.14159;

type
      pScreen = ^pScreenType;
      pScreenType = array[0..SCREENSIZE] of byte;

VAR
      texture : pScreen;
      composite :pScreen;
      y320 : array[0..HEIGHT] of word;
        i: integer;

    Back                              : Pointer;
    Size                              : Word;
    x1,x2,x3,x4,x5                    : Integer;
    y1,y2,y3,y4,y5                    : Integer;
    teta,ax4,ay4                      : integer;

process : array [1..10] of pointer;
pict    : array [1..10] of pointer;


{Procedure SetGraphicsMode;
Procedure SetTextMode;}
Function LoadImage(filename : string) : byte;
procedure copycomposite;
procedure clearcomposite;
Procedure FastRotate(scale, ang : Real);

implementation
{ switch to 320x200 256 straight VGA }
{Procedure SetGraphicsMode; assembler;
Asm
      mov ax,13h
```

```
      int 10h
      mov dx,3c2mov al,0e3h
      out dx,al
End;


{ back to 80x25 text mode }
{Procedure SetTextMode; assembler;
Asm
      mov ax,03h
      int 10h
End;    }


{ load the 320x200 grayscale bitmap into memory }
Function LoadImage(filename : string) : byte;
type
      pPalette = ^PaletteType;
      PaletteType = array[0..PALETTESIZE] of byte;

Var
      pFileMem : pScreen;
      FileHandle : file;
      thrash : pScreen;
      palette : pPalette;
      result : word;

Begin
      Assign(FileHandle, filename);
      Reset(FileHandle, 1);
      GetMem(thrash, 320*200);
      GetMem(palette, 1024);
      Seek(FileHandle, 54);
      BlockRead(FileHandle, palette^, 1024, result);
      BlockRead(FileHandle, thrash^, 320*200, result);
      Close(FileHandle);
      port[$3c6]:=$0ff;
      port[$3c8]:=0;
      Asm
            les si,palette
            mov cx,256
            cld
            mov dx,$3c9
            @ploop:
            mov al,es:[si+2]
            shr al,1
            out dx,al
            mov al,es:[si+1]
            shr al,1
            out dx,al
            mov al,es:[si]
            shr al,1
```

```
                out dx,al
                add si,4
                dec cx
                jne @ploop
        End;


        { Thrash the dumb MS format... The image is stored uncompressed
        UPSIDE-DOWN, each line being on a 32-bit DWORD boundry. Hmm. }
        Begin
              Asm
                push ds
                cld
                mov cx, HEIGHT
                les di, texture
                lds si, thrash
                add di, (HEIGHT-1)*WIDTH
                @tloop:
                push cx
                push si
                push di
                mov cx, WIDTH/2
                rep movsw
                pop di
                pop si
                pop cx
                add si, WIDTH
                sub di, WIDTH
                dec cx
                cmp cx,0
                jne @tloop
                pop ds
              End;
        End;
        FreeMem(palette, 1024);
        FreeMem(thrash, 320*200);
        LoadImage:=SUCCESS;
End;

{ copy the composition texture to the VGA screen memory. }
Procedure Copycomposite; assembler;
Asm
        push ds
        mov di, VGA
        mov es,di
        sub di,di
        lds   si,composite
        mov cx,320*200/2
        cld
        rep movsw
        pop ds
End;
```

```
{ clear the composition texture }
Procedure Clearcomposite; assembler;
Asm
      les di, composite
      mov cx,320*200/2
      sub ax,ax
      cld
      rep stosw
End;

{ This is it. This rotates and scales the image into a 200x200 window.
}
Procedure FastRotate(scale, ang : Real);
VAR
      xscale,
      yscale,
      xc,
      yc : Longint;
      scanline : word;
      x, y : Integer;
      tempx, tempy : word;
      xlong, ylong : Longint;
      n: integer;
      tseg, toff : word;
      hseg, hoff : word;
      texel : byte;

Begin
        for n:=0 to 199 do y320[n]:=n*320;
      xscale := round ( (sin(ang)*65536.0)*scale);
      yscale := round ( (cos(ang)*65536.0)*scale);
      xc := 160*65536 - (100*(yscale+xscale));
      yc := 100*65536 - (100*(yscale-xscale));
      scanline:=0;

      tseg:=seg(texture^);
      toff:=ofs(texture^);
      hseg:=seg(composite^);
      hoff:=ofs(composite^);

      for y:=0 to 199 do
      Begin
            xlong:=xc;
            ylong:=yc; { init x/ylong to topleft of square }
            for x:=60 to 60+200 do
            Begin { normally from 0 to 319 }
                  tempx:=xlong SHR 16;
                  tempy:=ylong SHR 16;

                  if (tempx<0) OR (tempx>=WIDTH) OR (tempy<0) OR
(tempy>=HEIGHT) then
                  Begin
                        Mem[hseg:hoff+scanline+x]:=0;
                  End
                  else
                  Begin
```

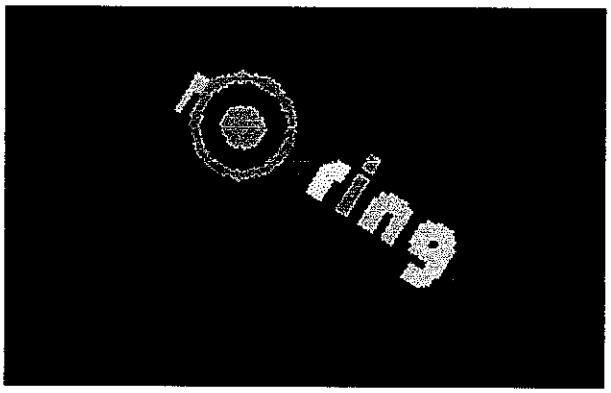
```
                        texel:=Mem[tseg:toff+y320[tempy]+tempx];
                        Mem[hseg:hoff+scanline+x]:=texel;
                    End;

                    inc(xlong,yscale);
                    dec(ylong,xscale);
                End;
                inc(scanline,WIDTH);
                inc(xc,xscale);
                inc(yc,yscale);
            End;
End;

end.
```

LAMPIRAN C

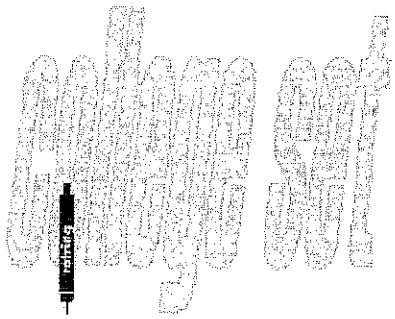
HASIL PROGRAM

Hasil Program Animasi Iklan Kartun

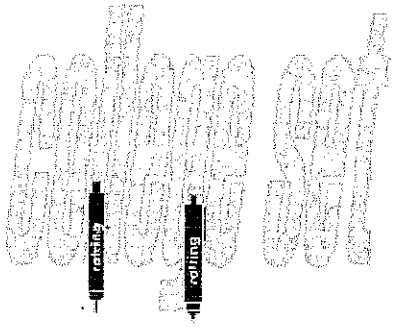
1. Perputaran tulisan rotring



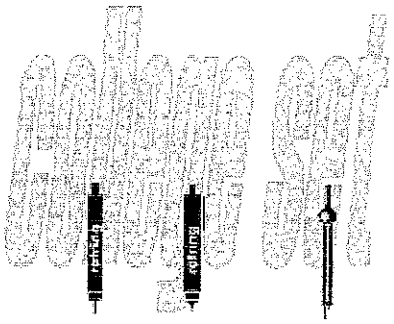
2. Pergerakan tinta dari kiri



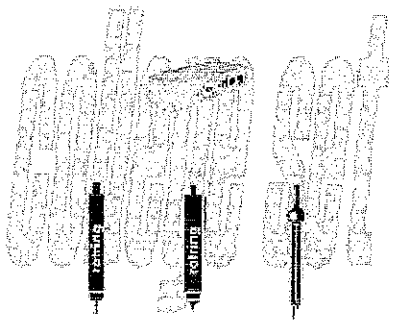
3. Pergerakan pensil dari sebelah bawah



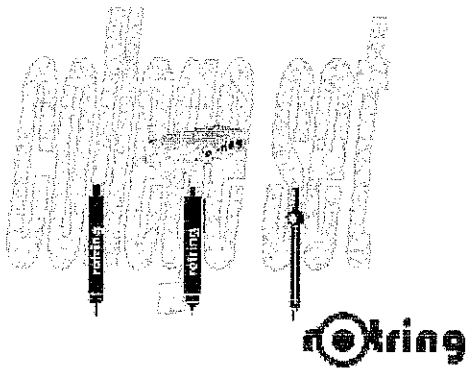
4. Pergerakan jangka dari sebelah kanan



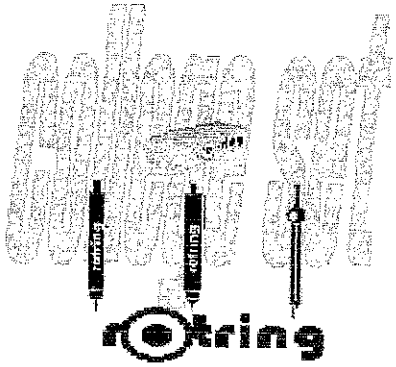
5. Pergerakan penghapus dari atas



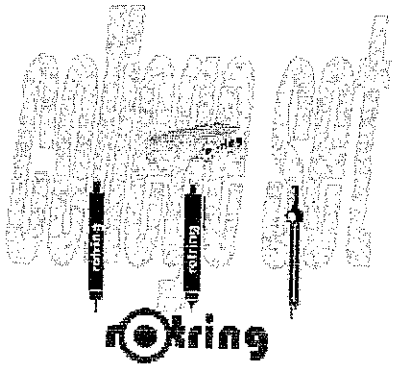
6. Pergerakan tulisan rotring dari sebelah kanan



7. Perbesaran yang terjadi pada tulisan rotring

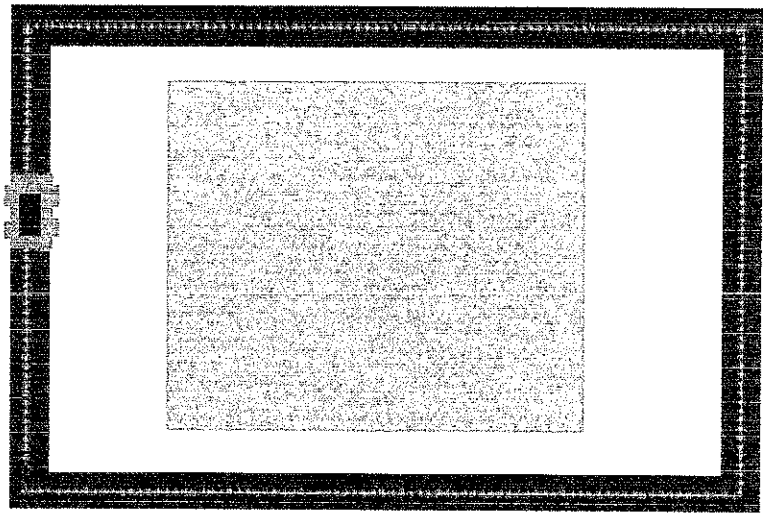


8. Semua benda telah berada pada posisi yang telah ditentukan

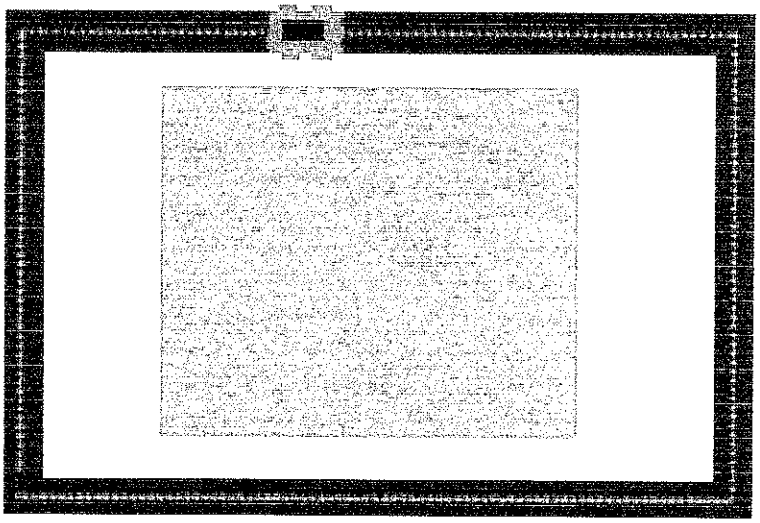


Hasil Program Animasi Kartun Mobil

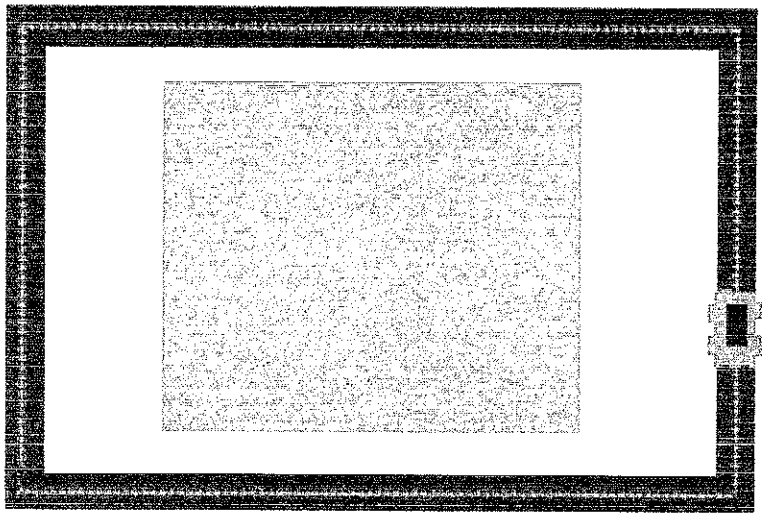
1. Mobil bergerak dari arah bawah ke atas



2. Mobil bergerak dari kiri ke kanan



3. Mobil bergerak dari arah atas ke bawah



4. Mobil bergerak dari arah kanan ke kiri