Лабораторна робота № 5

Розробити програму для емуляції дисплейного модуля (розширені можливості).

Виконав студент

Групи кн21-1

Кончич Данііл

Варіант 14

1. Розширити можливості програми реалізованою в лабораторній роботі №2 (емулятор дисплейного модуля) шляхом реалізації команди draw_text тільки з використанням команд малювання ліній. Програма повинна зберегти сумісність з програмами, розробленими в лабораторних роботах №3 і №4.

2. Інтерфейс бібліотеки GraphicsLib не змінювати.

3. Протокол обміну командами не змінювати.

Хід роботи:

Лістинг програми(Client):

```
unit Maim;

interface

uses
  System.SysUtils, System.Types, System.UITypes, System.Classes, System.Variants,
  FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics, FMX.Dialogs,
  FMX.Controls.Presentation, FMX.StdCtrls, IdBaseComponent, IdComponent,
  IdUDPBase, IdUDPClient, FMX.Memo.Types, FMX.ScrollBox, FMX.Memo, System.DateUtils, idGlobal,
  FMX.Edit, FMX.ComboEdit, FMX.Objects;
```

```
type TPacket = packed record
  msLen:Byte;
  colorarray:array [1..40,1..40] of cardinal;
  w:integer;
  h:integer;
  msg:string[255];
end;


const commands: array [1..8] of string = (
    'drawline', 'drawellipse', 'drawtext', 'clear',
'drawimage','fillroundedrectangle','drawpixel','drawsymbol'
);


// Перечисление для типов команд
type TCommand=(DRAW_LINE, DRAW_ELLIPSE, DRAW_TEXT, CLEAR, DRAW_IMAGE, FILL_ROUNDED_RECTANGLE,
DRAW_PIXEL, DRAW_SYMBOL);


type
  TForm1 = class(TForm)
    IdUDPClient1: TIdUDPClient;
    Button1: TButton;
    Memo1: TMemo;
    ComboEdit1: TComboEdit;
    Label1: TLabel;
    Image1: TImage;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
    bmp:TBitmap;
    packet:TPacket;
    send_data:TIdBytes;
    sendcommand:TCommand;
  public
    { Public declarations }
    function DrawPixelEncode(const sendcommand, px1,py1,parcolor:string):string;
    function DrawLineEncode(const sendcommand,
parx1,pary1,parx2,pary2,parcolor:string):string;
    function DrawSymbolEncode(const sendcommand, symbol, x,y,parcolor:string):string;
    function DrawEllipseEncode(const sendcommand, elx1,ely1,elx2,ely2,parcolor:string):string;
```

```
    function DrawTextEncode(const sendcommand, tx1,ty1,tx2,ty2,text,parcolor:string):string;
    function ClearEncode(const sendcommand:string; const parcolor:string):string;
    function DrawImageEncode(const sendcommand:string; width,heigth:string):string;
    function FillRoundedRectangleEncode(const sendcommand:string;
px1,py1,px2,py2,radius,parcolor:string):string;
  end;

var
  Form1: TForm1;

implementation

{$R *.fmx}
{$R *.iPhone47in.fmx IOS}

procedure TForm1.Button1Click(Sender: TObject);
var spl:TArray<string>; s:string; i:integer; iw,jw:integer; b:TBitmapData;
begin

  packet.msLen:=Length(Memo1.Text);
  SetLength(packet.msg,packet.msLen);

  s:=Memo1.Text;
  spl:=s.Split([' ']);

  for i:=1 to 8 do
  begin
    if commands[i]=spl[0] then
    begin
      sendcommand:=TCommand(i-1);
      case sendcommand of
      TCommand.DRAW_LINE:
        packet.msg:=DrawLineEncode((i-1).ToString,spl[1],spl[2],spl[3],spl[4],spl[5]);
      TCommand.DRAW_ELLIPSE:
        packet.msg:=DrawEllipseEncode((i-1).ToString,spl[1],spl[2],spl[3],spl[4],spl[5]);
      TCommand.DRAW_TEXT:
        packet.msg:=DrawTextEncode((i-1).ToString,spl[1],spl[2],spl[3],spl[4],spl[5],spl[6]);
      TCommand.CLEAR:
```

```
        packet.msg:=ClearEncode((i-1).ToString,spl[1]);
      TCommand.DRAW_IMAGE:
      begin
        packet.msg:=DrawImageEncode((i-1).ToString,spl[1],spl[2]);
        bmp:=TBitmap.CreateFromFile(spl[3]);

        packet.w:=bmp.Width;
        packet.h:=bmp.Height;



        bmp.Map(TMapAccess.Read,b);

        for iw:=1 to Round(bmp.Width) do
        for jw:=1 to Round(bmp.Height) do
          packet.colorarray[iw,jw]:=b.GetPixel(iw,jw);

        bmp.Unmap(b);
        Image1.Bitmap.Assign(bmp);

      end;
      TCommand.FILL_ROUNDED_RECTANGLE:
      begin
        packet.msg:=FillRoundedRectangleEncode((i-
1).ToString,spl[1],spl[2],spl[3],spl[4],spl[5],spl[6]);
      end;
      TCommand.DRAW_PIXEL:
      begin
        packet.msg:=DrawPixelEncode((i-1).ToString,spl[1],spl[2],spl[3]);
      end;
      TCommand.DRAW_SYMBOL:
      begin
        packet.msg:=DrawSymbolEncode((i-1).ToString,spl[1],spl[2],spl[3],spl[4]);
      end;
      end;
    end;
  end;

  IdUDPClient1.Active:=true;
```

```
  IdUDPClient1.Port:=5000;
  IdUDPClient1.Host:=ComboEdit1.Text;
  IdUDPClient1.Connect;

  if IdUDPClient1.Connected then
  begin
    SetLength(send_data,sizeof(packet));
    Move(packet,send_data[0],sizeof(packet));
    IdUDPClient1.SendBuffer(send_data);
  end;

  IdUDPClient1.Active:=false;

end;

function TForm1.ClearEncode(const sendcommand:string; const parcolor: string): string;
var command:integer;
begin
try
    command:=Integer.Parse(sendcommand);
    Result:=command.ToString+' '+parcolor;
  except on EConvertError do
  begin
    ShowMessage('Цвет неверный!!!');
    Result:='3 '+'000000';
  end;
  end;
end;

function TForm1.DrawSymbolEncode(const sendcommand, symbol, x, y, parcolor: string): string;
var xx,yy: Double; command:integer;
begin
  try
    xx:=Double.Parse(x);
    yy:=Double.Parse(y);
    command:=Integer.Parse(sendcommand);
    Result:=command.ToString+' '+symbol+' '+xx.ToString+' '+yy.ToString+' '+parcolor;
  except on EConvertError do
```

```
  begin
    ShowMessage('Координаты буквы неверны!!!');
    Result:='7 0 0 0 0';
  end;
end;
end;

function TForm1.DrawEllipseEncode(const sendcommand, elx1, ely1, elx2, ely2,
  parcolor: string): string;
var x1,y1,x2,y2,command:integer;
begin
  try
    x1:=Integer.Parse(elx1);
    y1:=Integer.Parse(ely1);
    x2:=Integer.Parse(elx2);
    y2:=Integer.Parse(ely2);
    command:=Integer.Parse(sendcommand);
    Result:=command.ToString+' '+x1.ToString+' '+y1.ToString+' '+x2.ToString+' '+y2.ToString+'
'+parcolor;
  except on EConvertError do
  begin
    ShowMessage('Координаты эллипса неверны!!!');
    Result:='1 0 0 0 0 '+parcolor;
  end;
  end;
end;

function TForm1.DrawImageEncode(const sendcommand: string; width,
  heigth: string): string;
var w,h,command:integer;
begin
  try
    w:=Integer.Parse(width);
    h:=Integer.Parse(heigth);
    command:=Integer.Parse(sendcommand);
    Result:=command.ToString+' '+w.ToString+' '+h.ToString;
  except on EConvertError do
  begin
```

```
    ShowMessage('размеры неверны!!!');
    Result:='4 0 0';
  end;
  end;
end;

function TForm1.DrawLineEncode(const sendcommand, parx1, pary1, parx2, pary2,
  parcolor: string): string;
var x1,y1,x2,y2,command:integer;
begin
  try
    x1:=Integer.Parse(parx1);
    y1:=Integer.Parse(pary1);
    x2:=Integer.Parse(parx2);
    y2:=Integer.Parse(pary2);
    command:=Integer.Parse(sendcommand);
    Result:=command.ToString+' '+x1.ToString+' '+y1.ToString+' '+x2.ToString+' '
    +y2.ToString+' '+parcolor;
  except on EConvertError do
  begin
    ShowMessage('Координаты линии неверны!!!');
    Result:='0 0 0 0 0 '+parcolor;
  end;
  end;
end;

function TForm1.DrawPixelEncode(const sendcommand, px1, py1,
  parcolor: string): string;
var x1,y1,command:integer;
begin
  try
    x1:=Integer.Parse(px1);
    y1:=Integer.Parse(py1);
    command:=Integer.Parse(sendcommand);
    Result:=command.ToString+' '+x1.ToString+' '+y1.ToString+' '+parcolor;
  except on EConvertError do
  begin
    ShowMessage('Координаты линии неверны!!!');
```

```
    Result:='6 0 0 '+parcolor;
  end;
  end;
end;

function TForm1.DrawTextEncode(const sendcommand, tx1, ty1, tx2, ty2, text,
  parcolor: string): string;
var x1,y1,x2,y2,command:integer;
begin
  try
    x1:=Integer.Parse(tx1);
    y1:=Integer.Parse(ty1);
    x2:=Integer.Parse(tx2);
    y2:=Integer.Parse(ty2);
    command:=Integer.Parse(sendcommand);
    Result:=command.ToString+' '+x1.ToString+' '+y1.ToString+' '+x2.ToString+' '
    +y2.ToString+' '+text+' '+parcolor;
  except on EConvertError do
  begin
    ShowMessage('Координаты линии неверны!!!');
    Result:='2 0 0 0 0 '+text+' '+parcolor;
  end;
  end;
end;

function TForm1.FillRoundedRectangleEncode(const sendcommand: string; px1, py1,
  px2, py2, radius, parcolor: string): string;
var x1,y1,x2,y2,rad,command,color:integer;
begin
  try
    x1:=Integer.Parse(px1);
    y1:=Integer.Parse(py1);
    x2:=Integer.Parse(px2);
    y2:=Integer.Parse(py2);
    rad:=Integer.Parse(radius);
    command:=Integer.Parse(sendcommand);
    Result:=command.ToString+' '+x1.ToString+' '+y1.ToString+' '+
    x2.ToString+' '+y2.ToString+' '+rad.ToString+' '+parcolor;
```

```
  except on EConvertError do
  begin
    ShowMessage('Ошибка!!!');
    Result:='5 0 0 0 0 0 0';
  end;
  end;
end;

end.
```

Лістинг програми(Server):

```
unit Main;

interface

uses
  System.SysUtils, System.Types, System.UITypes, System.Classes, System.Variants,
  FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics, FMX.Dialogs,
  FMX.Controls.Presentation, FMX.StdCtrls, IdBaseComponent, IdComponent,
  IdUDPBase, IdUDPServer, IdGlobal, IdSocketHandle, FMX.Memo.Types,
  FMX.ScrollBox, FMX.Memo, System.DateUtils, FMX.Objects, MyCommands,
System.Generics.Collections;

const symbols: array [1..8] of string = (
    'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H'
);




// Запись для приема данных от клиента
type TPacket = packed record
  msLen:Byte;
  colorarray:array [1..40,1..40] of cardinal;
  w:integer;
  h:integer;
  msg:string[255];
end;
```

```
// Параметры картинки

type TPicData = class
  pic:TBitmap;
  x:Double;
  y:Double;
  constructor Create(var x,y:Double;var pic:TBitmap); overload;
end;

// Параметры надписи

type TTextData = class
  text:string;
  x1:Double;
  y1:Double;
  x2:Double;
  y2:Double;
  color:string;
  constructor Create(var text:string; var x1,y1,x2,y2:Double; color:string); overload;
end;


type TEllipseData = class
  x1:Double;
  y1:Double;
  x2:Double;
  y2:Double;
  color:string;
  constructor Create(var x1,y1,x2,y2:Double; color:string); overload;
end;


type TPixelData = class
  x1:Double;
  y1:Double;
  color:string;
  constructor Create(var x1,y1:Double; color:string); overload;
```

```
end;

type TSymbolData = class
  x:Double;
  y:Double;
  color:string;
  symbpos:integer;
  constructor Create(var x, y : Double; color : string;  symbpos : integer); overload;
end;



type TFillRoundedRectangleData = class
  x1:Integer;
  y1:Integer;
  x2:Integer;
  y2:Integer;
  radius:Integer;
  color:string;
  constructor Create(var x1,y1,x2,y2,radius:Integer;color:string); overload;
end;



// Параметры линии

type TLineData = class
  p1:TPointF;
  p2:TPointF;
  color:string;
  constructor Create(var p1,p2:TPointF; color:string); overload;
end;

// Перечисление для типов команд
type TCommand=(DRAW_LINE, DRAW_ELLIPSE, DRAW_TEXT, CLEAR, DRAW_IMAGE, FILL_ROUNDED_RECTANGLE,
DRAW_PIXEL, DRAW_SYMBOL);

type
  TForm1 = class(TForm)
    IdUDPServer1: TIdUDPServer;
```

```
    ToolBar1: TToolBar;
    Label2: TLabel;
    PaintBox1: TPaintBox;
    procedure FormCreate(Sender: TObject);
    procedure IdUDPServer1UDPRead(AThread: TIdUDPListenerThread;
      const AData: TIdBytes; ABinding: TIdSocketHandle);
    procedure PaintBox1Paint(Sender: TObject; Canvas: TCanvas);
  private
    { Private declarations }
    bmp:TBitmap;
    packet:TPacket;
    command:TCommand;
    drawcommand:integer;
    piclist:TList<TPicData>;
    textlist:TList<TTextData>;
    linelist:TList<TLineData>;
    ellipselist:TList<TEllipseData>;
    fillroundedrectanglelist:TList<TFillRoundedRectangleData>;
    pixellist:TList<TPixelData>;
    symbollist:TList<TSymbolData>;
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.fmx}

procedure TForm1.FormCreate(Sender: TObject);
begin
  IdUDPServer1.Active:=true;
  TMyCommands.linepath:=TPathData.Create;
  TMyCommands.ellipsepath:=TPathData.Create;
  TMyCommands.clearcolor:='000000';
  piclist:=TList<TPicData>.Create;
```

```
  textlist:=TList<TTextData>.Create;
  linelist:=TList<TLineData>.Create;
  ellipselist:=TList<TEllipseData>.Create;
  fillroundedrectanglelist:=TList<TFillRoundedRectangleData>.Create;
  pixellist:=TList<TPixelData>.Create;
  symbollist:=TList<TSymbolData>.Create;
end;

procedure TForm1.IdUDPServer1UDPRead(AThread: TIdUDPListenerThread;
  const AData: TIdBytes; ABinding: TIdSocketHandle);
var s:string; i:integer;     spl:TArray<string>; iw,jw:integer;
    b1:TBitmapData; picdata:TPicData; textdata:TTextData;
    linedata:TLineData; ellipsedata:TEllipseData;
    fillroundedrectangledata:TFillRoundedRectangleData;
    pixeldata:TPixelData; px,py:Double; mysymboldata:TSymbolData;
    symbolpos:integer; symbolx,symboly:Double;  symbolcolor:string;
begin

          Move(AData[0],packet,sizeof(packet));
          s:=packet.msg;
          spl:=s.Split([' ']);

          // Парсим полученную команду от клиента

          command:=TCommand(Integer.Parse(spl[0]));

        case command of
          TCommand.DRAW_LINE:
          begin
            drawcommand:=Integer.Parse(spl[0]);
            TMyCommands.PrepareLine(spl[1],spl[2],spl[3],spl[4],spl[5]);
            linedata:=TLineData.Create(TMyCommands.p1,TMyCommands.p2,TMyCommands.linecolor);
            linelist.Add(linedata);
            PaintBox1.Repaint;
          end;
          TCommand.DRAW_ELLIPSE:
          begin
            drawcommand:=Integer.Parse(spl[0]);
```

```
            TMyCommands.PrepareEllipse(spl[1],spl[2],spl[3],spl[4],spl[5]);
            ellipsedata:=TEllipseData.Create(TMyCommands.x1_ellipse,TMyCommands.y1_ellipse,
            TMyCommands.x2_ellipse,TMyCommands.y2_ellipse,TMyCommands.ellipsecolor);
            ellipselist.Add(ellipsedata);
            PaintBox1.Repaint;
          end;
          TCommand.DRAW_TEXT:
          begin
            drawcommand:=Integer.Parse(spl[0]);
            TMyCommands.PrepareText(spl[1],spl[2],spl[3],spl[4],spl[5],spl[6]);

textdata:=TTextData.Create(TMyCommands.textout,TMyCommands.x1_text,TMyCommands.y1_text,
            TMyCommands.x2_text,TMyCommands.y2_text,TMyCommands.textcolor);
            textlist.Add(textdata);
            PaintBox1.Repaint;
          end;
          TCommand.CLEAR:
          begin
            drawcommand:=Integer.Parse(spl[0]);
            TMyCommands.PrepareClear(spl[1]);
            piclist.Clear;
            textlist.Clear;
            linelist.Clear;
            pixellist.Clear;
            symbollist.Clear;
            ellipselist.Clear;
            fillroundedrectanglelist.Clear;
            Form1.Fill.Color:=StrToInt('$ff'+TMyCommands.clearcolor);
            PaintBox1.Repaint;
          end;
          TCommand.DRAW_IMAGE:
          begin
            drawcommand:=Integer.Parse(spl[0]);
            TMyCommands.PrepareDrawImage(spl[1],spl[2]);
            bmp:=TBitmap.Create();

            bmp.SetSize(packet.w,packet.h);
```

```
            bmp.Map(TMapAccess.Write,b1);

            for iw:=1 to Round(bmp.Width) do
            for jw:=1 to Round(bmp.Height) do
            begin
              b1.SetPixel(iw,jw,packet.colorarray[iw,jw]);
            end;
            bmp.Unmap(b1);

            picdata:=TPicData.Create(TMyCommands.ximage,TMyCommands.yimage,bmp);
            piclist.Add(picdata);

            PaintBox1.Repaint;
          end;
          TCommand.FILL_ROUNDED_RECTANGLE:
          begin
TMyCommands.PrepareFillRoundedRectangle(spl[1],spl[2],spl[3],spl[4],spl[5],spl[6]);
fillroundedrectangledata:=TFillRoundedRectangleData.Create(TMyCommands.x1,TMyCommands.y1,
TMyCommands.x2,TMyCommands.y2,TMyCommands.radius,TMyCommands.fillroundedrectanglecolor);
            fillroundedrectanglelist.Add(fillroundedrectangledata);
            PaintBox1.Repaint;
          end;
          TCommand.DRAW_PIXEL:
          begin
            TMyCommands.PreparePixel(spl[1],spl[2],spl[3]);
            px:=TMyCommands.ppoint.X;
            py:=TMyCommands.ppoint.Y;
            pixeldata:=TPixelData.Create(px, py, TMyCommands.pixelcolor);
            pixellist.Add(pixeldata);
            PaintBox1.Repaint;
          end;
          TCommand.DRAW_SYMBOL:
          begin
            TMyCommands.PrepareSymbol(spl[1],spl[2],spl[3],spl[4]);
            for symbolpos:=1 to 8 do
            begin
```

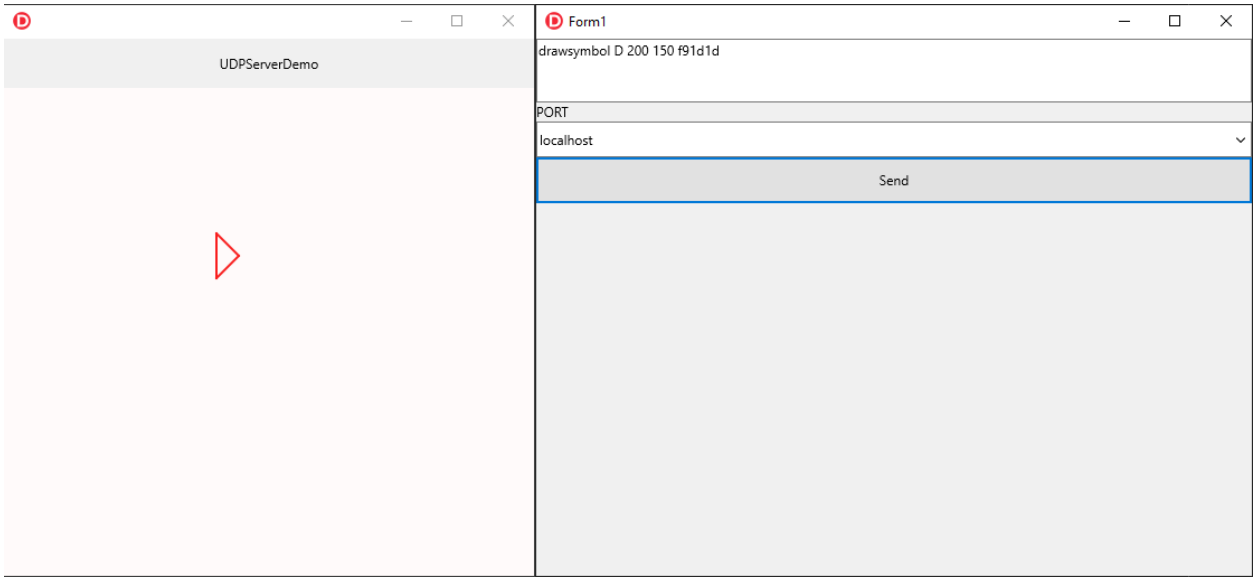
```
              if TMyCommands.symbol=symbols[symbolpos] then
              begin
                symbolx:=TMyCommands.sx;
                symboly:=TMyCommands.sy;
                symbolcolor:=TMyCommands.symbolcolor;
                mysymboldata:=TSymbolData.Create(symbolx, symboly, symbolcolor, (symbolpos-
1));
                symbollist.Add(mysymboldata);
              end;
            end;

            PaintBox1.Repaint;
          end;



        end;

end;



procedure TForm1.PaintBox1Paint(Sender: TObject; Canvas: TCanvas);
var i:integer; p:TPicData; t:TTextData; l:TLineData;  e:TEllipseData;
    frr:TFillRoundedRectangleData; pixel:TPixelData; a:TSymbolData;
begin
  PaintBox1.Canvas.BeginScene();

        for l in linelist do
          TMyCommands.DrawMyLine(l.p1,l.p2,Canvas,StrToInt('$ff'+l.color));

        for e in ellipselist do
          TMyCommands.DrawMyEllipse(e.x1,e.y1,e.x2,e.y2,Canvas,StrToInt('$ff'+e.color));

        for t in textlist do
          TMyCommands.DrawMyText(t.x1,t.y1,t.x2,t.y2,
             t.text, 30, Canvas, StrToInt('$ff'+t.color));

        for p in piclist do
          TMyCommands.DrawImage(p.x,p.y,p.pic,Canvas);
```

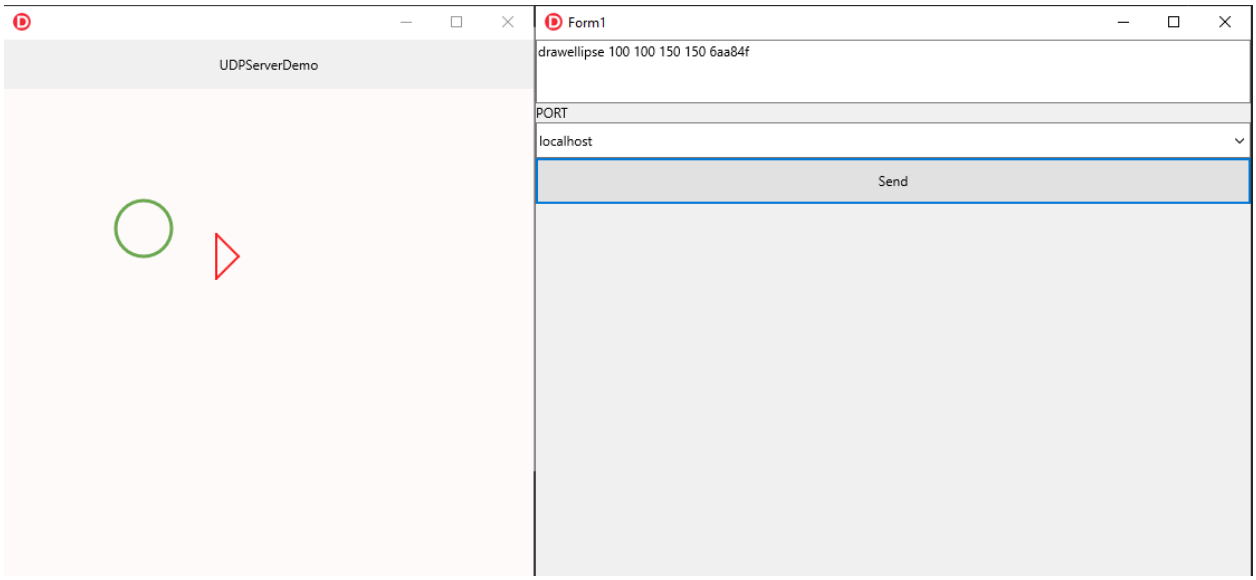
```
        for frr in fillroundedrectanglelist do
          TMyCommands.FillRoundedRectangle(frr.x1,frr.y1,frr.x2,frr.y2,frr.radius,
            Canvas,StrToInt('$ff'+frr.color));

        for pixel in pixellist do
        begin
          TMyCommands.DrawMyPixel(TPointF.Create(pixel.x1,pixel.y1),
            Canvas,StrToInt('$ff'+pixel.color));
        end;

        for a in symbollist do
        begin
TMyCommands.DrawSymbol(a.symbpos,TPointF.Create(a.x,a.y),Canvas,StrToInt('$ff'+a.color));
        end;

  PaintBox1.Canvas.EndScene;

end;




{ TPicData }

constructor TPicData.Create(var x, y: Double; var pic: TBitmap);
begin
  Self.x:=x;
  Self.y:=y;
  Self.pic:=pic;
end;

{ TTextData }

constructor TTextData.Create(var text:string; var x1,y1,x2,y2:Double; color:string);
begin
  Self.text:=text;
```

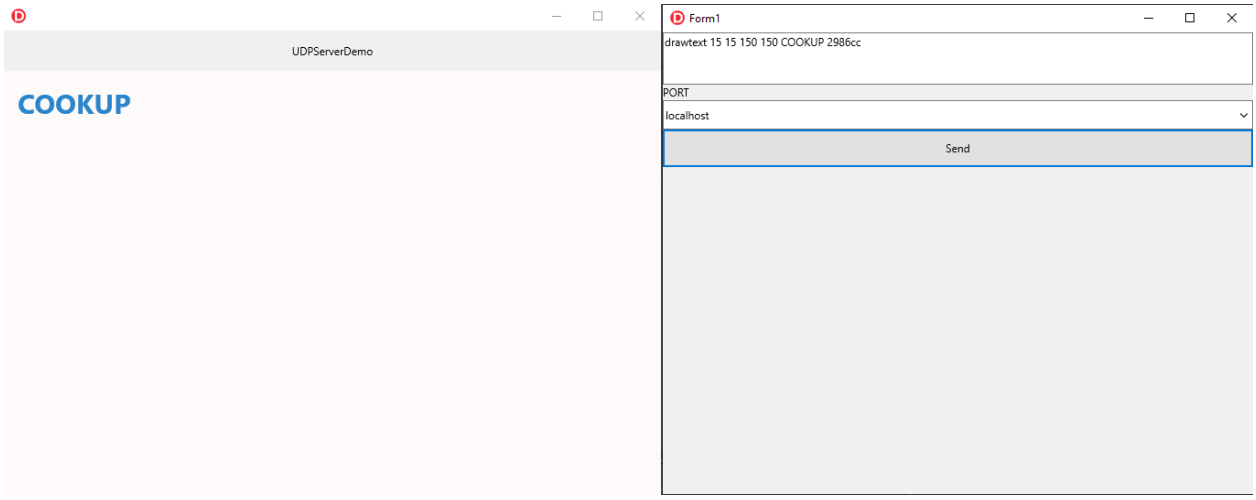
```
  Self.x1:=x1;
  Self.y1:=y1;
  Self.x2:=x2;
  Self.y2:=y2;
  Self.color:=color;
end;

{ TLineData }

constructor TLineData.Create(var p1,p2:TPointF; color:string);
begin
  Self.p1:=p1;
  Self.p2:=p2;
  Self.color:=color;
end;

{ TEllipseData }

constructor TEllipseData.Create(var x1, y1, x2, y2: Double; color: string);
begin
  Self.x1:=x1;
  Self.y1:=y1;
  Self.x2:=x2;
  Self.y2:=y2;
  Self.color:=color;
end;

{ TFillRoundedRectangleData }

constructor TFillRoundedRectangleData.Create(var x1, y1, x2, y2,
  radius: Integer; color: string);
begin
  Self.x1:=x1;
  Self.y1:=y1;
  Self.x2:=x2;
  Self.y2:=y2;
  Self.radius:=radius;
  Self.color:=color;
```

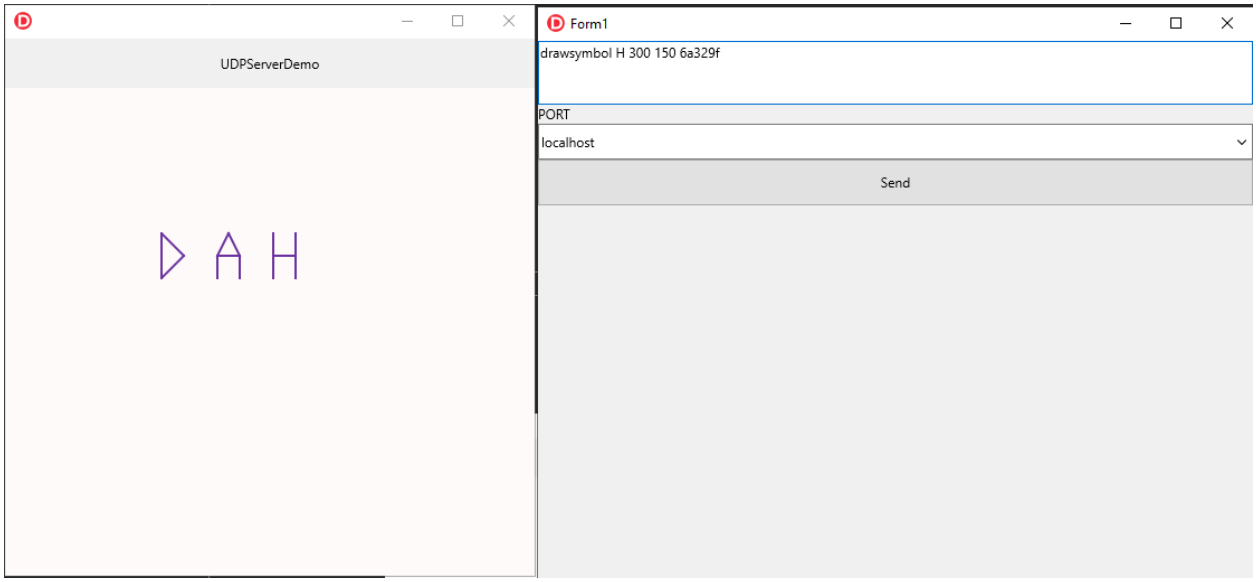
```
end;

{ TPixelData }

constructor TPixelData.Create(var x1, y1: Double; color: string);
begin
  Self.x1:=x1;
  Self.y1:=y1;
  Self.color:=color;
end;

{ TAData }

constructor TSymbolData.Create(var x, y: Double; color: string; symbpos : integer);
begin
  Self.symbpos:=symbpos;
  Self.x:=x;
  Self.y:=y;
  Self.color:=color;
end;

end.
```

Результати роботи:









Висновок: розробив програму для емуляції дисплейного модуля (розширені можливості).