# ПОЛНЫЙ НАБОР ИСХОДНЫХ ТЕКСТОВ КЛИЕНТА

## ОГЛАВЛЕНИЕ

[ПОЛНЫЙ НАБОР ИСХОДНЫХ ТЕКСТОВ КЛИЕНТА 1](#_Toc483044359)

[ОГЛАВЛЕНИЕ 1](#_Toc483044360)

[Main, образ экрана 1](#_Toc483044361)

[unit Main; 1](#_Toc483044362)

[unit Common1; 12](#_Toc483044363)

[unit AppDispatch1; 13](#_Toc483044364)

[UserApp01Main, образ экрана 16](#_Toc483044365)

[unit UserApp01Main; 16](#_Toc483044366)

[unit Scope01; 21](#_Toc483044367)

[unit Oscillograph01; 31](#_Toc483044368)

[unit Service01; 40](#_Toc483044369)

[unit ImgLoadSave01; 46](#_Toc483044370)

[UserApp02Main, образ экрана 55](#_Toc483044371)

[unit UserApp02Main; 56](#_Toc483044372)

[UserApp03Main, образ экрана 60](#_Toc483044373)

[unit UserApp03Main; 60](#_Toc483044374)

[unit UserApp03Data; 62](#_Toc483044375)

[unit Cargo01; 64](#_Toc483044376)

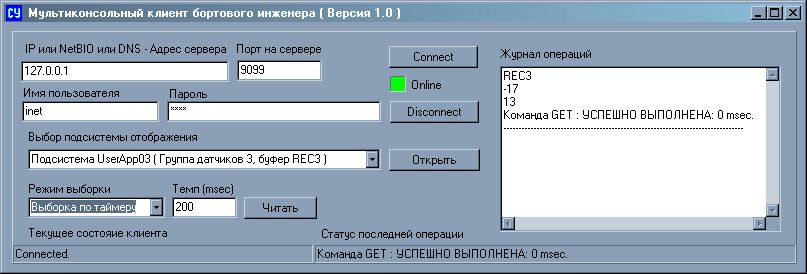
[unit Loading01; 66](#_Toc483044377)

[unit AirCraft01; 88](#_Toc483044378)

[unit MomentScope01; 90](#_Toc483044379)

[unit md5; 98](#_Toc483044380)

## Main, образ экрана



## unit Main;

unit Main;

// ==========================================================================

// КЛИЕНТ

// ==========================================================================

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, MMSystem, StrUtils, StdCtrls, ExtCtrls, ComCtrls,

//

IdBaseComponent, IdComponent, IdTCPConnection, IdTCPClient,

//

Common1, AppDispatch1,

UserApp01Main, UserApp02Main, UserApp03Main;

type

TClientForm = class(TForm)

TCPClient: TIdTCPClient;

Timer1: TTimer;

Timer2: TTimer;

Panel1: TPanel;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

Label5: TLabel;

Label6: TLabel;

Label7: TLabel;

Label8: TLabel;

Label9: TLabel;

edIP: TEdit;

edPort: TEdit;

Button1: TButton;

Button2: TButton;

stxtConnect: TStaticText;

edName: TEdit;

edPSW: TEdit;

StatusBar1: TStatusBar;

Button4: TButton;

cbBoxCMD: TComboBox;

Button3: TButton;

cboBoxReadTimer: TComboBox;

Label10: TLabel;

edTimerValue: TEdit;

Label11: TLabel;

Label1: TLabel;

Memo1: TMemo;

procedure TCPClientConnected(Sender: TObject);

procedure TCPClientDisconnected(Sender: TObject);

procedure TCPClientStatus(ASender: TObject;

const AStatus: TIdStatus; const AStatusText: String);

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure Timer1Timer(Sender: TObject);

procedure Button3Click(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure Button4Click(Sender: TObject);

procedure Timer2Timer(Sender: TObject);

procedure edTimerValueChange(Sender: TObject);

private

{ Private declarations }

// Попытка отправить сообщение на сервер

function TryWriteLn(Msg : string) : boolean;

// Попытка прочитать сообщение от сервера

function TryReadLn(var Msg : string) : boolean;

// Попытка прочитать буфер переданный сервером

function TryReadBuffer(var ABuffer;

const AByteCount: Integer;

var SatusStr : string) : boolean;

// Выполнить дисконнект клиента

function DoDisConnect(Msg : string) : boolean;

// Выполнить запрос клиента на обслуживание

procedure RunRequest(RqCmdInd : integer);

public

{ Public declarations }

end;

var

ClientForm: TClientForm;

// ==========================================================================

// ==========================================================================

implementation

{$R \*.dfm}

// ==========================================================================

// ==========================================================================

// ==========================================================================

// СЕГМЕНТ ОБЩЕСИСТЕМНЫХ КОНСТАНТ

// ==========================================================================

// ------------------------------

// Реплики и команды выдаваемые сервером

const cRepTimeOut = 20; // Тайаут для рандеву сервера и клиента

cFieldSep = ':'; // Разделитель полей в командах и квитанциях

cRepYES = 'YES'; // Код позитивной квитанции от сервера

cRepNOT = 'NOT'; // Код негативной квитанции от сервера

cCmdEND = 'END'; // Код команды отключения клиента

// -----------------------------

// Команды выдаваемые клиентом

const cCmdConnect = 'CNN'; // Код команды подключения к серверу

cCmdDisConnect = 'DCN'; // Код команды отключения от сервера

cCmdLink = 'LNK'; // Код команды запроса на подтверждение коннекта

cCmdService = 'SRV'; // Код команды к сервисам обслуживания клиента

// -----------------------------

// Запросы выдаваемые клиентом

const cCmdGET = 'GET'; // Запрос на блок данных

cCmdSET = 'SET'; // Запрос на конфигурацию сервера или сервисов

// -----------------------------

// Структура для оценок времени выполнения

// -----------------------------

// Отметки времени по мультимедийному таймеру (шаг 1 msec)

// Используются процедуры из unit MMSystem.pas

type TTimeStampsMM = record

MMErr : word; // Код ошибки при обращении к ММ

BTime : LongInt; // Стартовая отметка в msec

ETime : LongInt; // Финишная отметка в msec

end;

// ==========================================================================

// СЕГМЕНТ СЕРВИСНЫХ ОПЕРАЦИЙ

// ==========================================================================

// -------------------------------------------------------------------------

// Получить директорию старта приложения

function GetApplicationDirectory() : string;

begin

Result := Application.ExeName;

Result := ExtractFileDir(Result);

end;

// --------------------------------------------------------------------------

// Получить Login-имя текущего пользователя

function GetCurrentUserName() : string;

const cnMaxUserNameLen = 254;

var sUserName : string;

dwUserNameLen : DWord;

begin

dwUserNameLen := cnMaxUserNameLen-1;

SetLength( sUserName, cnMaxUserNameLen );

GetUserName(

PChar( sUserName ),

dwUserNameLen );

SetLength( sUserName, dwUserNameLen );

Result := sUserName;

end;

// --------------------------------------------------------------------------

// Преобразование RqEdit.Text в целое число со знаком

function TryEditToInt(RqEdit : TEdit; var Value : integer) : boolean;

begin

Result := False;

try

if Trim(RqEdit.Text) = ''

then Value := 0

else begin

Value := StrToInt(Trim(RqEdit.Text));

RqEdit.Color := clWindow;

end;

Result := True;

except

RqEdit.Color := RGB(255,240,240);

ShowMessage('Не могу преобразовать в целое число со знаком');

end;

end;

// --------------------------------------------------------------------------

// 24.11.2016

// Вырезать из текста команды очередное поле

// Пример 1

// На входе Cmd = 'field1:field2:field3'

// На выходе Result = 'field1', Cmd = 'field2:field3'

// Пример 2

// На входе Cmd = 'field3'

// На выходе Result = 'field3', Cmd = ''

function CutNextCmdField(var Cmd : string) : string;

var wPos : integer;

wStr : string;

begin

wStr := Trim(Cmd);

Result := wStr;

wPos := pos(cFieldSep, wStr);

if wPos > 0

then begin

Result := copy(wStr, 1, wPos - 1);

if (Length(wStr) > wPos)

then Cmd := copy(wStr, wPos + 1, Length(wStr))

end

else Cmd := '';

end;

// =========================================================================

// ОЦЕНКА ВРЕМЕННЫХ ИНТЕРВАЛОВ С ПОМОЩЬЮ МУЛЬТИМЕДИЙНОГО ТАЙМЕРА

// =========================================================================

// 10.03.2013

// Зафиксировать стартовую временную отметку

procedure StartMMTimeStamp (var RqStamp : TTimeStampsMM);

begin

try

RqStamp.BTime := timeGetTime;

RqStamp.MMErr := 0;

except

RqStamp.MMErr := 1; // Мультимедиа таймер недоступен

end;

end;

// -------------------------------------------------------------------------

// 10.03.2013

// Зафиксировать финишную временную отметку и отобразить интервал

function StopMMTimeStamp (var RqStamp : TTimeStampsMM) : string;

begin

if RqStamp.MMErr = 0

then begin

with RqStamp

do begin

try

ETime := timeGetTime;

if (ETime - BTime) >= 0

then Result := IntToStr(ETime - BTime);

except

MMErr := 1;

Result := '-1';

end;

end;

end

else Result := '-1';

end;

// ==========================================================================

// СЕГМЕНТ СЕРВИСНЫХ TRY-ОПЕРАЦИЙ ДЛЯ ВЗАИМОДЕЙСТВИЯ КЛИЕНТА И СЕРВЕРА

// ==========================================================================

// --------------------------------------------------------------------------

// 24.11.2016

// Попытка отправить сообщение на сервер

function TClientForm.TryWriteLn(Msg : string) : boolean;

begin

try

TCPClient.WriteLn(Msg);

Result := True;

except

Result := False;

end;

end;

// --------------------------------------------------------------------------

// 24.11.2016

// Попытка прочитать сообщение от сервера

function TClientForm.TryReadLn(var Msg : string) : boolean;

var RepeatCount : integer;

begin

RepeatCount := 10; // Число повторов

Msg := '';

try

repeat

Msg := TCPClient.ReadLn('', cRepTimeOut);

Dec(RepeatCount);

until ((Msg <> '') or (RepeatCount < 0));

// Msg <> '' // Сообщение сервера успешно прочитано

// Msg = '' // Выход по таймауту

Result := True;

except

// Выход по аварии в операции чтения

Result := False;

end;

end;

// --------------------------------------------------------------------------

// 06.12.2016

// Попытка прочитать буфер переданный сервером

function TClientForm.TryReadBuffer(var ABuffer;

const AByteCount: Integer;

var SatusStr : string) : boolean;

const TimeOut = 10000; // 10 секунд. Можно поднимать до 60 сек.

var TimeStampsMM : TTimeStampsMM; // Для оценки времени выполнения

begin

Result := False;

if (AByteCount > 0) and (@ABuffer <> nil)

then begin

StartMMTimeStamp(TimeStampsMM);

with TCPClient

do begin

// Попытка выполнить чтение

while (InputBuffer.Size < AByteCount)

do begin

try

try

// Читать порцию

ReadFromStack(True,TimeOut,True);

except

SatusStr := 'Команда GET : ОТКАЗ, TimeOut';

Exit;

end;

// Проверить разрыв коннекта

CheckForDisconnect(True, True);

except

SatusStr := 'Команда GET : ОТКАЗ, Disconnect';

Exit;

end;

end;

// Copy it to the callers buffer

Move(InputBuffer.Memory^, ABuffer, AByteCount);

// Remove used data from buffer

InputBuffer.Remove(AByteCount);

// ---------------------------

SatusStr := 'Команда GET : УСПЕШНО ВЫПОЛНЕНА: '

+ StopMMTimeStamp(TimeStampsMM) + ' msec.';

Result := True;

end;

end;

end;

// ==========================================================================

// СЕГМЕНТ ВЫПОЛНЕНИЯ ДИСКОННЕКТА

// ==========================================================================

// --------------------------------------------------------------------------

// 07.01.2017

// Выполнить дисконнект клиента

function TClientForm.DoDisConnect(Msg : string) : boolean;

begin

Result := False;

// Остановить таймер запуска проверки связи между сервером и клиентом

Timer1.Enabled := False;

// Выполнить Disconnect

if TCPClient.Connected

then begin

// Отправить серверу команду - "Дисконнект клиента"

TryWriteLn(cCmdDisConnect);

end;

// Безусловное отключение ( игнорируя TCPClient.Connected )

try

TCPClient.Disconnect();

StatusBar1.Panels[1].Text := Msg;

Result := True;

except

StatusBar1.Panels[1].Text := 'ERROR : Не могу выполнить'

+ 'дисконнект клиента';

end;

end;

// --------------------------------------------------------------------------

// 07.01.2017

// Обработчик события - "Выполненяется Disconnect"

procedure TClientForm.TCPClientDisconnected(Sender: TObject);

begin

// Погасить индикатор Online

stxtConnect.Color := clBtnFace;

// Отчет в журнал

Memo1.Clear;

Memo1.Lines.Add('СЕАНС ЗАКОНЧЕН');

end;

// --------------------------------------------------------------------------

// 07.01.2017

// Инициатор выполнения DisConnect со стороны клиента

procedure TClientForm.Button2Click(Sender: TObject);

begin

// Выполнить отключение

DoDisConnect('Клиент отключился от сервера');

end;

// ==========================================================================

// СЕГМЕНТ ВЫПОЛНЕНИЯ КОННЕКТА

// ==========================================================================

// --------------------------------------------------------------------------

// 07.01.2017

// Проверка связи между сервером и клиентом

procedure TClientForm.Timer1Timer(Sender: TObject);

var wCmd : string;

wRep : string;

wYesNot : string; // Тип квитанции

begin

// ---------------------------------------

// Проверка сообщений от сервера, отправленных по его инициативе

if TCPClient.Connected

then begin

if TryReadLn(wRep)

then begin

if wRep <> ''

then begin

// Получено сообщение от сервера, выделим командную часть

wCmd := CutNextCmdField(wRep);

if wCmd = cCmdEND

then begin

// Получен запрос на принудительный дисконнект

DoDisConnect('Сервер разорвал соединение командой : ' + cCmdEND);

Exit;

end;

end;

end

else begin

// Авария при попытке чтения сообщений от сервера

DoDisConnect('Не могу выполнить операцию чтения с сервера');

Exit;

end;

end;

// ---------------------------------------

// Проверка связи между клиентом и сервером

if TCPClient.Connected

then begin

// Выдать запрос на проверку связи

if not TryWriteLn(cCmdLink)

then begin

// Авария при попытке отправить запрос на проверку связи

DoDisConnect('Сервер не реагирует на запрос : ' + cCmdLink);

Exit;

end

else begin

// Прочитать ответ сервера на запрос - Проверка связи

if TryReadLn(wRep)

then begin

if wRep = ''

then begin

// Связь утрачена

DoDisConnect(cCmdLink

+ ' : Сервер не отвечает в течении таймаута');

Exit;

end

else begin

// Выделить тип квитанции

wYesNot := CutNextCmdField(wRep);

if wYesNot = cRepYES

then begin

// Связь успешно проверена

// В этой версии ничего не предпринимаем

Exit;

end;

end;

end

else begin

// Авария при попытке чтения сообщений от сервера

DoDisConnect(cCmdLink

+ ' : Не могу выполнить операцию чтения с сервера');

Exit;

end;

end;

end;

end;

// --------------------------------------------------------------------------

// 07.01.2017

// Обработчик события - "Выполненяется Connect"

procedure TClientForm.TCPClientConnected(Sender: TObject);

var wCmd : string;

wOp : string;

begin

// Рандеву авторизации

wCmd := cCmdConnect + cFieldSep + edName.Text + cFieldSep + edPSW.Text;

if not TryWriteLn(wCmd)

then begin

StatusBar1.Panels[1].Text := 'Сервер не реагирует на запрос : '

+ cCmdConnect;

Exit;

end;

// Прочитать и проанализировать квитанцию

if TryReadLn(wCmd)

then begin

// Анализ квитанции

wOp := CutNextCmdField(wCmd);

if wOp = ''

then begin

DoDisConnect('Сервер не отвечает');

Exit;

end;

if wOp = cRepYES

then begin

Timer1.Enabled := True;

StatusBar1.Panels[1].Text := 'Сервер выполнил подключение';

stxtConnect.Color := clLime;

Memo1.Clear;

Memo1.Lines.Add('НАЧАЛО СЕАНСА');

Exit;

end;

if wOp = cRepNOT

then begin

DoDisConnect('Сервер отказал в подключении');

Exit;

end;

end

else begin

DoDisConnect('ERROR : Не могу выполнить операцию чтения с сервера');

end;

end;

// --------------------------------------------------------------------------

// Инициатор выполнения - подключение к серверу

procedure TClientForm.Button1Click(Sender: TObject);

begin

if (edIP.Text = '') or

(edPort.Text = '') or

(edName.Text = '') or

(edPSW.Text = '')

then begin

StatusBar1.Panels[1].Text := 'Не все поля для подключения заполнены';

Exit;

end;

TCPClient.Host := edIP.Text;

TCPClient.Port := StrToInt(edPort.Text);

try

TCPClient.Connect();

except

StatusBar1.Panels[1].Text := 'Сервер не отвечает на запрос: Connect';

end;

end;

// --------------------------------------------------------------------------

// 07.01.2017

// Обработчик события - Изменился текущий статус клиента

procedure TClientForm.TCPClientStatus(ASender: TObject;

const AStatus: TIdStatus; const AStatusText: String);

begin

StatusBar1.Panels[0].Text := AStatusText;

end;

// ==========================================================================

// СЕГМЕНТ ВЫПОЛНЕНИЯ ЗАПРОСОВ КЛИЕКТА

// ==========================================================================

// --------------------------------------------------------------------------

// 07.01.2017

// Выполнить запрос клиента на обслуживание

procedure TClientForm.RunRequest(RqCmdInd : integer);

var wRequest : TRequest; // Блок управления запросом к серверу

wCmd : string; // Текст запроса к серверу

wRep : string; // Текст ответа сервера

begin

// ----------------------------

// Контроль подключения

if not TCPClient.Connected

then begin

StatusBar1.Panels[1].Text := 'Клиент еще не подключен к серверу';

Exit;

end;

// ----------------------------

// Подготовить блок управления запросом к серверу

if not SetRequest(RqCmdInd, wRequest)

then begin

StatusBar1.Panels[1].Text := 'Неизвестная пользовательская команда';

Exit;

end;

// ----------------------------

// Начальные установки

wCmd := '';

wRep := '';

StatusBar1.Panels[1].Text :='';

Memo1.Clear;

// ----------------------------

// Сформируем текст запроса к серверу

wCmd := cCmdService

+ cFieldSep + Trim(wRequest.RqCmd)

+ cFieldSep + Trim(wRequest.RqPrm);

// ----------------------------

// Отправим заявку на сервер

if not TryWriteLn(wCmd)

then begin

StatusBar1.Panels[1].Text := 'Сервер не реагирует на запрос : '

+ wCmd;

Exit;

end;

// ----------------------------

// Читаем квитанцию по заявке

if TryReadLn(wRep)

then begin

wRequest.Replay := wRep;

// Контроль превышения таймаута

if wRep = ''

then begin

DoDisConnect('Сервер не отвечает в течении таймаута');

Exit;

end;

// Выделим тип квитанции

wRep := CutNextCmdField(wRep);

// Анализ квитанции

if wRep = cRepNOT

then begin

StatusBar1.Panels[1].Text := 'Сервер отказал '

+ 'в обслуживании команды : '

+ wCmd;

Exit;

end;

// Если сервер ответил положительной квитанцией

if wRep = cRepYES

then begin

if (wRequest.RqCmd = 'GET') and

(wRequest.Buffer <> nil) and

(wRequest.BufSize > 0)

then begin

// ----------------------------

// Читать результаты обслуживания

if TryReadBuffer(wRequest.Buffer^,

wRequest.BufSize,

wRep)

then begin

ShowResult(wRequest, Memo1);

end

else begin

Memo1.Lines.Add('Проблема c ReadBuffer');

end;

end;

if (wRequest.RqCmd = 'SET')

then begin

// В данной версии команды SET не используются

end;

// Отчет о выполнении команды

StatusBar1.Panels[1].Text := wRep;

end;

end;

// Отчет о выполнении команды в журнал

Memo1.Lines.Add(wRep);

Memo1.Lines.Add (DupeString('-', 80));

end;

// ==========================================================================

// СЕГМЕНТ УПРАВЛЕНИЯ ЗАПРОСАМИ

// ==========================================================================

// --------------------------------------------------------------------------

// 07.01.2017

// Инициатор выполнения запросов клиента на обслуживание

procedure TClientForm.Button3Click(Sender: TObject);

begin

RunRequest(cbBoxCMD.ItemIndex);

end;

// --------------------------------------------------------------------------

// 16.01.2017

// Читать по таймеру

procedure TClientForm.Timer2Timer(Sender: TObject);

begin

if cboBoxReadTimer.ItemIndex = 1

then RunRequest(cbBoxCMD.ItemIndex);

end;

// --------------------------------------------------------------------------

// 16.01.2017

// Установка интервала повторения запросов

procedure TClientForm.edTimerValueChange(Sender: TObject);

var Value : integer;

begin

if TryEditToInt(edTimerValue, Value)

then begin

Timer2.Enabled := False;

if Value < 20

then begin

Timer2.Interval := 20;

edTimerValue.Text := '20';

end

else Timer2.Interval := Value;

Timer2.Enabled := True;

end;

end;

// ==========================================================================

// СЕГМЕНТ ИНИЦИИРОВАНИЯ / ТЕРМИНИРОВАНИЯ

// ==========================================================================

// --------------------------------------------------------------------------

// ИНИЦИАЛИЗАЦИЯ

procedure TClientForm.FormCreate(Sender: TObject);

begin

// Запомнить директорию старта проекта

ApplicationDirectory := GetApplicationDirectory();

// Получить логин текущего пользователя

edName.Text := GetCurrentUserName();

end;

// --------------------------------------------------------------------------

// Обслживание попыток терминировать активный клиент

procedure TClientForm.FormClose(Sender: TObject; var Action: TCloseAction);

begin

if TCPClient.Connected

then begin

Action := caNone;

MessageDlg('Вы не можете завершить приложение'

+ #13#10 + 'пока клиент активиен( Online )...',

mtInformation, [mbOk], 0);

end;

end;

// --------------------------------------------------------------------------

// Открыть выбранную форму (тоесть консоль подсистемы)

procedure TClientForm.Button4Click(Sender: TObject);

begin

case cbBoxCMD.ItemIndex of

0 : UserApp01Form.Show;

1 : UserApp02Form.Show;

2 : UserApp03Form.Show;

end;

end;

// ==========================================================================

// КОНЕЦ

// ==========================================================================

end.

## unit Common1;

interface

uses

Windows, {Messages, SysUtils, Variants, Classes, Graphics, Controls,} Forms;

// ==========================================================================

// БУФЕРА ДЛЯ ПРИКЛАДНОЙ ЧАСТИ

// ==========================================================================

// --------------------------------------------------------------------------

const REC1MaxIndx = 5;

// Структура запмси в буфере результатов выполнения команды 'GET:REC1'

type TREC1 = packed record

Id : string[8];

Arr : array[0..REC1MaxIndx] of extended;

end;

// --------------------------------------------------------------------------

const REC2MaxIndx = 3;

// Структура запмси в буфере результатов выполнения команды 'GET:REC2'

type TREC2 = packed record

Id : string[8];

Arr : array[0..REC2MaxIndx] of extended;

end;

// --------------------------------------------------------------------------

// Структура запмси в буфере результатов выполнения команды 'GET:REC3'

type TREC3 = packed record

Id : string[8];

Pith : extended;

Roll : extended;

end;

// ==========================================================================

// --------------------------------------------------------------------------

// Директория старта приложения

var ApplicationDirectory : string;

// ==========================================================================

// БУФЕРА РЕЗУЛЬТАТОВ ВЫПОЛНЕНИЯ ПРИКЛАДНЫХ СЕРВИСОВ

// ==========================================================================

var REC1 : TREC1; // Буфер результатов команды 'REC1'

REC2 : TREC2; // Буфер результатов команды 'REC2'

REC3 : TREC3; // Буфер результатов команды 'REC3'

// ==========================================================================

implementation

// ==========================================================================

// ==========================================================================

// КОНЕЦ

// ==========================================================================

end.

## unit AppDispatch1;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, ExtCtrls, ComCtrls,

Common1,

UserApp01Main, UserApp02Main, UserApp03Main;

// ==========================================================================

// СТРУКТУРА ЗАПРОСА К ПРИКЛАДНЫМ СЕРВИСАМ СЕРВЕРА

// ==========================================================================

type TRequest = record

RqCmd : string; // Комманда на обслуживание

RqPrm : string; // Параметры комманды на обслуживание

Replay : string; // Текст квитанции от сервера

// Буфер чтения для команд группы GET

Buffer : pointer; // Адрес буфера или nil

BufSize : integer; // Размер буфера в байтах

end;

// --------------------------------------------------------------------------

// Подготовить блок управления запросом к серверу

function SetRequest(RqCmdInd : integer; var RqRequest : TRequest) : boolean;

// Отобразить результаты выполнения запроса

procedure ShowResult(RqRequest : TRequest; RqReport : TMemo);

// ==========================================================================

// --------------------------------------------------------------------------

implementation

// --------------------------------------------------------------------------

// ==========================================================================

// --------------------------------------------------------------------------

// Подготовить блок управления запросом к серверу

function SetRequest(RqCmdInd : integer; var RqRequest : TRequest) : boolean;

begin

Result := False;

// Очистить блок управления запросом к серверу

FillChar(RqRequest, SizeOf(RqRequest), #0);

// Заполнить блок управления запросом к серверу

with RqRequest do

begin

case RqCmdInd of

0 : begin

RqCmd := 'GET';

RqPrm := 'REC1';

Buffer := Addr(REC1);

BufSize := SizeOF(REC1);

Result := True;

end;

1 : begin

RqCmd := 'GET';

RqPrm := 'REC2';

Buffer := Addr(REC2);

BufSize := SizeOF(REC2);

Result := True;

end;

2 : begin

RqCmd := 'GET';

RqPrm := 'REC3';

Buffer := Addr(REC3);

BufSize := SizeOF(REC3);

Result := True;

end;

end;

end;

end;

// --------------------------------------------------------------------------

// Выполнить команду 'REC1'

procedure RunREC1(RqReport : TMemo);

var Ind : integer;

begin

RqReport.Clear;

with RqReport.Lines

do begin

Add(REC1.Id);

for Ind := Low(REC1.Arr) to High(REC1.Arr)

do begin

Add(FloatToStr(REC1.Arr[Ind]));

end;

UserApp01Form.ShowSensors;

end;

end;

// --------------------------------------------------------------------------

// Выполнить команду 'REC2'

procedure RunREC2(RqReport : TMemo);

var Ind : integer;

begin

RqReport.Clear;

with RqReport.Lines

do begin

Add(REC2.Id);

for Ind := Low(REC2.Arr) to High(REC2.Arr)

do begin

Add(FloatToStr(REC2.Arr[Ind]));

end;

UserApp02Form.ShowSensors;

end;

end;

// --------------------------------------------------------------------------

// Выполнить команду 'REC3'

procedure RunREC3(RqReport : TMemo);

begin

RqReport.Clear;

with RqReport.Lines

do begin

Add(REC3.Id);

Add(FloatToStr(REC3.Pith));

Add(FloatToStr(REC3.Roll));

Loading.PitchAngle := REC3.Pith;

Loading.RollAngle := REC3.Roll;

end;

end;

// --------------------------------------------------------------------------

// Отобразить результаты выполнения запроса

procedure ShowResult(RqRequest : TRequest; RqReport : TMemo);

begin

//--------------------

// Команда 'GET:REC1'

if (UpperCase(RqRequest.RqCmd) = 'GET') and

(UpperCase(RqRequest.RqPrm) = 'REC1')

then begin

// Выполнение прикладного сервиса

RunREC1(RqReport);

Exit;

end;

//--------------------

// Команда 'GET:REC2'

if (UpperCase(RqRequest.RqCmd) = 'GET') and

(UpperCase(RqRequest.RqPrm) = 'REC2')

then begin

// Выполнение прикладного сервиса

RunREC2(RqReport);

Exit;

end;

//--------------------

// Команда 'GET:REC3'

if (UpperCase(RqRequest.RqCmd) = 'GET') and

(UpperCase(RqRequest.RqPrm) = 'REC3')

then begin

// Выполнение прикладного сервиса

RunREC3(RqReport);

Exit;

end;

end;

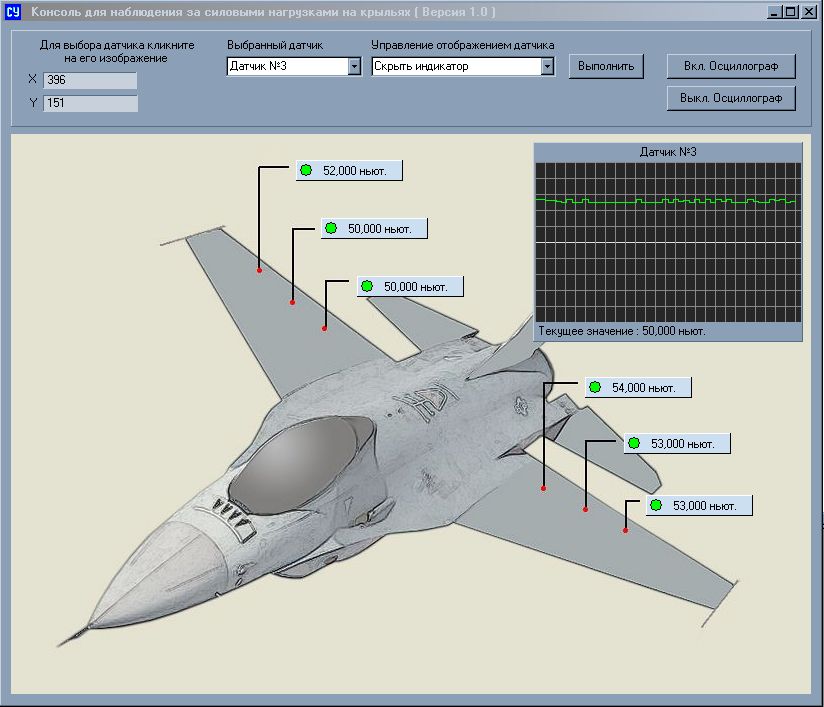
// ==========================================================================

// КОНЕЦ

// ==========================================================================

end.

## UserApp01Main, образ экрана



## unit UserApp01Main;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, ExtCtrls, StdCtrls,

Common1, Service01, ImgLoadSave01, Scope01, Oscillograph01;

type

TUserApp01Form = class(TForm)

Image1: TImage;

Panel1: TPanel;

Label4: TLabel;

Label5: TLabel;

Label6: TLabel;

Button2: TButton;

Button3: TButton;

StaticText3: TStaticText;

StaticText4: TStaticText;

ComboBox1: TComboBox;

ComboBox2: TComboBox;

Button4: TButton;

Label1: TLabel;

Label2: TLabel;

procedure Button2Click(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure FormDestroy(Sender: TObject);

procedure Image1MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure Button4Click(Sender: TObject);

procedure Button3Click(Sender: TObject);

procedure ComboBox2Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

// Отобразить новое значение на датчиках

procedure ShowSensors();

end;

var

UserApp01Form: TUserApp01Form;

// =========================================================================

// =========================================================================

implementation

{$R \*.dfm}

// =========================================================================

// =========================================================================

// =========================================================================

// ОПИСАНИЕ ДАННЫХ

// =========================================================================

const ArrIndMax = 5; // Конечный индекс массивов датчиков

// Описатель параметров датчика

type TDescScope = array[0..ArrIndMax] of record

Xb, YB : integer;

Compact : boolean;

Title : string;

Min : extended;

Max : extended;

end;

// Массив статических параметров датчика

const DescScope : TDescScope =

(

(Xb:285; YB:26; Compact:True; Title:'Датчик №1'; Min:-110; Max:110),

(Xb:310; YB:84; Compact:True; Title:'Датчик №2'; Min:-110; Max:110),

(Xb:346; YB:142; Compact:True; Title:'Датчик №3'; Min:-110; Max:110),

(Xb:574; YB:243; Compact:True; Title:'Датчик №4'; Min:-110; Max:110),

(Xb:613; YB:299; Compact:True; Title:'Датчик №5'; Min:-110; Max:110),

(Xb:635; YB:361; Compact:True; Title:'Датчик №6'; Min:-110; Max:110)

);

// =========================================================================

// РАБОЧИЕ ДАННЫЕ ДАННЫХ

// =========================================================================

// Индекс выбранного датчика

var SelectInd : integer;

// Рабочий массив датчиков

var ScopeArr : array[0..ArrIndMax] of TScope;

// Объект - Осциллограф

var Osc01 : TOscillograph;

// Объект - загрузки / сохраниния картинки

var ImgLoadSave1 : TLoadSaveImage; // Обект загрузки и сохранения Image

// =========================================================================

// РАБОЧИЕ ПРОЦЕДУРЫ И ФУНКЦИИ

// =========================================================================

// -------------------------------------------------------------------------

// Выбор датчика

procedure TUserApp01Form.Image1MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

var Ind : integer;

begin

// Координаты MouseDown

StaticText3.Caption := ' ' + IntToStr(X);

StaticText4.Caption := ' ' + IntToStr(Y);

// Выбор сенсора

if Shift \* [ssAlt] = []

then begin

for Ind := Low(ScopeArr) to High(ScopeArr)

do begin

if not Assigned(ScopeArr[Ind]) then Continue;

if IsXYInRect(ScopeArr[Ind].SRect, X, Y)

then begin

SelectInd := Ind;

ComboBox2.ItemIndex := SelectInd;

if Assigned(Osc01)

then Osc01.Title := DescScope[SelectInd].Title;

end;

end;

end;

// Переместить осциллограф

if (Shift \* [ssAlt] <> []) and Assigned(Osc01)

then begin

Osc01.Free;

Osc01 := TOscillograph.Create(UserApp01Form,

Rect(X + Image1.Left,

Y + Image1.Top,

X + Image1.Left + 270,

Y + Image1.Top + 200));

Osc01.Title := DescScope[SelectInd].Title;

end;

end;

// -------------------------------------------------------------------------

// Выбор датчика

procedure TUserApp01Form.ComboBox2Click(Sender: TObject);

var Ind : integer;

begin

Ind := ComboBox2.ItemIndex;

if (Ind >= Low(ScopeArr)) and (Ind <= High(ScopeArr))

then begin

SelectInd := Ind;

if Assigned(Osc01)

then Osc01.Title := DescScope[SelectInd].Title;

end;

end;

// -------------------------------------------------------------------------

// Операции с индикаторами (Выполнить)

procedure TUserApp01Form.Button2Click(Sender: TObject);

var Ind : integer;

begin

Ind := ComboBox2.ItemIndex;

if Assigned(ScopeArr[Ind])

then begin

case ComboBox1.ItemIndex of

// Скрыть индикатор

0 : ScopeArr[Ind].Visible := False;

//Показать индикатор

1 : ScopeArr[Ind].Visible := True;

// Сделать компактным

2 : ScopeArr[Ind].Compact := True;

// Сделать полным

3 : ScopeArr[Ind].Compact := False;

// Сделать прозрачным

4 : ScopeArr[Ind].Transparent := True;

//Сделать непрозрачным

5 : ScopeArr[Ind].Transparent := False;

end;

end;

end;

// -------------------------------------------------------------------------

// Отобразить новое значение на датчиках

procedure TUserApp01Form.ShowSensors();

var Ind : integer;

begin

for Ind := Low(ScopeArr) to High(ScopeArr)

do begin

if Assigned(ScopeArr[Ind])

then begin

ScopeArr[Ind].Value := REC1.Arr[Ind];

ScopeArr[Ind].DrawScope();

end;

end;

if Assigned(Osc01)

then begin

Osc01.AddNewData(REC1.Arr[SelectInd]);

Osc01.ShowBeam;

end;

end;

// -------------------------------------------------------------------------

// Включить осциллограф

procedure TUserApp01Form.Button3Click(Sender: TObject);

const XB = 530;

YB = 120;

begin

if not Assigned(Osc01)

then begin

Osc01 := TOscillograph.Create(UserApp01Form,

Rect(XB,YB,XB + 270,YB + 200));

Osc01.Title := ScopeArr[SelectInd].Title;

Osc01.YMes := ScopeArr[SelectInd].Measure;

end;

end;

// -------------------------------------------------------------------------

// Выключить осциллограф

procedure TUserApp01Form.Button4Click(Sender: TObject);

begin

if Assigned(Osc01)

then begin

Osc01.Free;

Osc01 := nil;

end;

end;

// =========================================================================

// ИНИЦИИРОВАНИЕ / ТЕРМИНИРОВАНИЕ ПРИЛОЖЕНИЯ

// =========================================================================

// -------------------------------------------------------------------------

procedure TUserApp01Form.FormCreate(Sender: TObject);

var Ind : integer;

begin

// -------------------------------

// Загрузить изображение объекта

ImgLoadSave1 := TLoadSaveImage.Create(nil,nil);

ImgLoadSave1.LoadImgFromFile(ApplicationDirectory

+ '\PICTUREs\destroyer01.jpg',

Image1);

// -------------------------------

ComboBox2.Clear;

for Ind := Low(ScopeArr) to High(ScopeArr)

do begin

ComboBox2.Items.Add(DescScope[Ind].Title);

ScopeArr[Ind] := TScope.Create(Image1);

with ScopeArr[Ind]

do begin

XBeg := DescScope[Ind].Xb;

YBeg := DescScope[Ind].Yb;

GLen := 150; // Длина графического индикатора

Transparent := False;

Compact := DescScope[Ind].Compact;

Measure := 'ньют.';

Min := DescScope[Ind].Min;

Max := DescScope[Ind].Max;

YZone := 80;

RZone := 90;

Title := DescScope[Ind].Title;

Visible := True;

end;

SelectInd := 0;

ComboBox2.ItemIndex := SelectInd;

end;

ComboBox1.ItemIndex := 0;

end;

// -------------------------------------------------------------------------

procedure TUserApp01Form.FormDestroy(Sender: TObject);

var Ind : integer;

begin

for Ind := Low(ScopeArr) to High(ScopeArr)

do begin

if Assigned(ScopeArr[Ind])

then ScopeArr[Ind].Free;

end;

if Assigned(Osc01)

then begin

Osc01.Free;

Osc01 := nil;

end;

if Assigned(ImgLoadSave1) then ImgLoadSave1.Free;

end;

// =========================================================================

// КОНЕЦ

// =========================================================================

end.

## unit Scope01;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,

Dialogs, ExtCtrls, StdCtrls;

type TScope = class(TObject)

private

// ------------------------

fImg : TImage; // Арендуемый Image

// ------------------------

// Абсолютные координаты индикатора

fXBeg : integer; // X - позиция индикатора

fYBeg : integer; // Y - позиция индикатора

fSRect : TRect; // Полный прямоугольник индикатора

// ------------------------

fCompact : boolean; // Компактное отображение

fTransparent : boolean; // Поля на прозрачной области индикатора

// ------------------------

// Размеры поля графического индикатора

fCGW : integer; // Ширина графического поля

fCGH : integer; // Высота графического поля

// ------------------------

// Абсолютные координаты полей индикатора

fIRect : TRect; // Прямоугольник иконки индикатора

fVRect : TRect; // Прямоугольник значений индикатора

fGRect : TRect; // Прямоугольник графического индикатора

fTRect : TRect; // Прямоугольник подписи индикатора

fTitle : string; // Заголовок индикатора

fBmp : TBitMap; // Копия площадки занимаемой индикатором

// ------------------------

// Параметры отображаемых значений

fMeasure : string; // Единица измерения

fMin : extended; // Минимальное значение индикатора

fMax : extended; // Абсолютное максимальное значение

fScale : extended; // Масштаб

fYZone : extended; // Начало Желтой зоны значений

fRZone : extended; // Начало Красной зоны значений

fValue : extended; // Текущее значение

// Параметры управления

fVisible : boolean; // Отображение индикатора

// ------------------------

// Расчет полной площадки под индикатор

procedure CalcSRect();

// ------------------------

// Установка позиции индикатора на fImg

procedure SetXBeg(XBeg : integer);

procedure SetYBeg(YBeg : integer);

procedure SetTitle (Title : string);

// ------------------------

// Обсуживание графического поля индикатора

procedure CalcScale();

// Получить значение графического поля в пикселях

function GetGW(Value : extended) : integer;

// ------------------------

// Управление видимостью индикатора на fImg

procedure RestoreBkGround();

procedure SaveBkGround();

function TestBkGround() : boolean;

procedure SetVisible(Visible : boolean);

procedure SetCompact(Compact : boolean);

procedure SetTransparent(Transparent : boolean);

// ------------------------

// Управвление диапазоном значений

procedure SetMeasure(Measure : string);

procedure SetMin (Min : extended);

procedure SetMax (Max : extended);

procedure SetCGW (GLen : integer);

// ------------------------

// Управвление зонами значений

procedure SetYZone (YZone : extended);

procedure SetRZone (RZone : extended);

public

// ------------------------

// Инициирование / терминирование

constructor Create(RqImage : TImage);

procedure Free();

// ------------------------

// Прорисовка индикатора

procedure DrawScope();

// ------------------------

// Геометрия индикатора

property SRect : TRect read fSRect;

// ------------------------

// Управление отображением

property Visible : boolean read fVisible write SetVisible;

property Compact : boolean read fCompact write SetCompact;

property Transparent : boolean read fTransparent write SetTransparent;

property XBeg : integer read fXBeg write SetXBeg;

property YBeg : integer read fYBeg write SetYBeg;

property Title : string read fTitle write SetTitle;

// ------------------------

// Границы значений для графического поля

property Measure : string read fMeasure write SetMeasure;

property Min : extended read fMin write SetMin;

property Max : extended read fMax write SetMax;

property GLen : integer read fCGW write SetCGW;

// ------------------------

// Текущее значение

property Value : extended read fValue write fValue;

// ------------------------

// Цветные зоны на индикаторе

property YZone : extended read fYZone write SetYZone;

property RZone : extended read fRZone write SetRZone;

// ------------------------

end;

implementation

// -------------------------------------------------------------------------

const cWIcon = 12;

cHIcon = 12;

cWBound = 4;

cHBound = 4;

cTxtPic = ' 123,123 ';

cOutPic = '%6.3f';

// -------------------------------------------------------------------------

constructor TScope.Create(RqImage : TImage);

begin

inherited Create;

// -------------------------------

// Подготовка fImg

fImg := RqImage;

if not Assigned(fImg.Picture) then fImg.Picture := TPicture.Create;

with fImg.Picture.Bitmap

do begin

if Width <> fImg.Width then Width := fImg.Width;

if Height <> fImg.Height then Height := fImg.Height;

if PixelFormat <> pf24bit then PixelFormat := pf24bit;

end;

// -------------------------------

// Подготовка геометрии индикатора

fXBeg := 0;

fYBeg := 0;

// Подпись индикатора

fTitle := 'Kонтрольная точка';

fCGW := 100; // Ширина графического поля

fCGH := 8; // Высота графического поля

fCompact := False;

fTransparent := True;

// -------------------------------

// Сохранение полной площадки индикатора

fBmp := TBitMap.Create;

CalcSRect();

SaveBkGround();

// -------------------------------

// Подготовка масштабов

fMeasure := ' ед. ';

fMin := 20;

fMax := 80;

fValue := fMin;

CalcScale();

end;

// -------------------------------------------------------------------------

procedure TScope.Free();

begin

if Assigned(fBmp)

then begin

RestoreBkGround();

fBmp.Free();

end;

inherited Free();

end;

// -------------------------------------------------------------------------

// -------------------------------------------------------------------------

// Расчет абсолютных габаритов полей индикатора

procedure TScope.CalcSRect();

var wH, wW : integer;

begin

with fImg

do begin

// Прямоугольник иконки индикатора

fIRect.Top := fYBeg + cHBound;

fIRect.Left := fXBeg + cWBound;

fIRect.Bottom := fIRect.Top + cHIcon;

fIRect.Right := fIRect.Left + cWIcon;

// Правая точка

wW := fIRect.Right + cWBound;

// Прямоугольник значений индикатора

fVRect.Top := fIRect.Top;

fVRect.Left := wW + cWBound;

fVRect.Bottom := fVRect.Top + Canvas.TextHeight('0');

fVRect.Right := fVRect.Left

+ Canvas.TextWidth(cTxtPic + ' ' + fMeasure + ' ');

// Нижняя правая точка

if fVRect.Bottom > fIRect.Bottom

then wH := fVRect.Bottom + cHBound

else wH := fIRect.Bottom + cHBound;

wW := fVRect.Right + cWBound;

// Дополнительные поля

if not fCompact

then begin

// Прямоугольник графического индикатора

fGRect.Top := wH;

fGRect.Left := fXBeg + cWBound;

fGRect.Bottom := fGRect.Top + fCGH;

fGRect.Right := fGRect.Left + fCGW;

// Нижняя правая точка

if fGRect.Right + cWBound > wW

then wW := fGRect.Right + cHBound;

wH := fGRect.Bottom + cHBound;

if fTitle <> ''

then begin

// Прямоугольник подписи индикатора

fTRect.Top := wH;

fTRect.Left := fXBeg + cWBound;

fTRect.Bottom := fTRect.Top + Canvas.TextHeight('0');

fTRect.Right := fTRect.Left + Canvas.TextWidth(' ' + fTitle + ' ');

// Нижняя правая точка

if fTRect.Right + cWBound > wW

then wW := fTRect.Right + cHBound;

wH := fTRect.Bottom + cHBound;

end;

end;

// Полный прямоугольник индикатора

fSRect.Top := fYBeg;

fSRect.Left := fXBeg;

fSRect.Bottom := wH;

fSRect.Right := wW;

end;

end;

// -------------------------------------------------------------------------

function SaveBitMap(FromBitMap, Bitmap : TBitmap;

FomRect : TRect) : boolean;

// ------------------------------------

// SysUtils:

// PByteArray = ^TByteArray;

// TByteArray = array[0..32767] of Byte;

// ------------------------------------

var

RowBeg, ColBeg : Integer;

RowEnd, ColEnd : Integer;

PFrom, PTo : PByteArray;

RowFrom, RowTo : Integer;

ColFrom, ColTo : Integer;

begin

Result := False;

if not (FromBitMap.PixelFormat = pf24bit) then Exit;

// Контроль геометрии наложения FomRec

// 1) FomRect за пределами FromBitMap

if FomRect.Top >= FromBitMap.Height then Exit;

if FomRect.Left >= FromBitMap.Width then Exit;

// 2) Нормировка диапазона строк

RowBeg := FomRect.Top;

if RowBeg < 0 then RowBeg := 0;

RowEnd := FomRect.Bottom;

if RowEnd > FromBitMap.Height - 1 then RowEnd := FromBitMap.Height - 1;

// 3) Нормировка диапазона столбцов

ColBeg := FomRect.Left;

if ColBeg < 0 then ColBeg := 0;

ColEnd := FomRect.Right;

if ColEnd > FromBitMap.Width - 1 then ColEnd := FromBitMap.Width - 1;

// 4) Проверка на пустой FomRec

if RowEnd < RowBeg then Exit;

if ColEnd < ColBeg then Exit;

// Начинаем копирование

try

Bitmap.PixelFormat := pf24bit;

Bitmap.Height := RowEnd - RowBeg + 1;

Bitmap.Width := ColEnd - ColBeg + 1;

RowTo := 0;

for RowFrom := RowBeg to RowEnd do

begin

// Представим строку BitMap через указатель на массив байтов

PFrom := FromBitMap.ScanLine[RowFrom];

PTo := Bitmap.ScanLine[RowTo];

// Побайтовое копирование. Порядок байтов в пикселе pf24bit : B,G,R

ColTo := 0;

for ColFrom := ColBeg \* 3 to ColEnd \* 3

do begin

PTo^[ColTo] := PFrom^[ColFrom];

Inc(ColTo);

end;

Inc(RowTo);

end;

Result := True;

except

ShowMessage('Не могу сохранить BitMap');

end;

end; // of function

// -------------------------------------------------------------------------

function RestoreBitMap(BitMap : TBitmap; // BitMap источник

ToBitMap : TBitmap; // BitMap приемник

Xb, Yb : integer) // Начальные координаты в приемнике

: boolean;

var

RowBeg, ColBeg : Integer;

RowEnd, ColEnd : Integer;

PFrom, PTo : PByteArray;

RowFrom, RowTo : Integer;

ColFrom, ColTo : Integer;

begin

Result := False;

if not (ToBitmap.PixelFormat = pf24bit) then Exit;

// Контроль геометрии наложения FomRec

// 1) Xb, Yb за пределами ToBitMap

if Yb >= ToBitmap.Height then Exit;

if Xb >= ToBitmap.Width then Exit;

// 2) Нормировка диапазона строк

RowBeg := Yb;

if Yb < 0

then begin

RowBeg := 0;

RowEnd := BitMap.Height - 1;

end

else RowEnd := Yb + BitMap.Height - 1;

if RowEnd > ToBitMap.Height - 1 then RowEnd := ToBitMap.Height - 1;

// 3) Нормировка диапазона столбцов

ColBeg := Xb;

if Xb < 0

then begin

ColBeg := 0;

ColEnd := BitMap.Width - 1;

end

else ColEnd := Xb + BitMap.Width - 1;

if ColEnd > ToBitMap.Width - 1 then ColEnd := ToBitMap.Width - 1;

// 4) Проверка на пустой ToRec

if RowEnd < RowBeg then Exit;

if ColEnd < ColBeg then Exit;

// Начинаем копирование

try

RowFrom := 0;

for RowTo := RowBeg to RowEnd do

begin

// Представим каждую строку BitMap как

// указатель на массив байтов

PFrom := BitMap.ScanLine[RowFrom];

PTo := ToBitmap.ScanLine[RowTo];

// побайтовое копирование изображения

ColFrom := 0;

for ColTo := ColBeg \* 3 to ColEnd \* 3

do begin

PTo^[ColTo] := PFrom^[ColFrom];

Inc(ColFrom);

end;

Inc(RowFrom);

end;

Result := True;

except

ShowMessage('Не могу восстановить BitMap');

end;

end; // of function

// -------------------------------------------------------------------------

// Проверить соответствие fSRect размерам fBmp

function TScope.TestBkGround() : boolean;

begin

Result := False;

if Assigned(fBmp)

then begin

if fBmp.Height <> (fSRect.Bottom - fSRect.Top) + 1 then Exit;

if fBmp.Width <> (fSRect.Right - fSRect.Left) + 1 then Exit;

Result := True;

end;

end;

// -------------------------------------------------------------------------

// Восстановить площадку

procedure TScope.RestoreBkGround();

begin

if TestBkGround() and fVisible

then begin

// Убрать индикатор с fImg

RestoreBitMap(fBmp, fImg.Picture.Bitmap, fSRect.Left,fSRect.Top);

fImg.Refresh;

end;

end;

// -------------------------------------------------------------------------

// Сохранить площадку под индикатор

procedure TScope.SaveBkGround();

begin

if Assigned(fBmp)

then SaveBitMap(fImg.Picture.Bitmap, fBmp, fSRect);

end;

// -------------------------------------------------------------------------

// Управление видимостью индикатора

procedure TScope.SetVisible(Visible : boolean);

begin

if Visible

then begin

if TestBkGround() and (not fVisible)

then begin

// Отобразить индикатор на fImg

fVisible := Visible;

DrawScope();

end;

end

else begin

if TestBkGround() and fVisible

then begin

// Убрать индикатор с fImg

RestoreBitMap(fBmp, fImg.Picture.Bitmap, fSRect.Left,fSRect.Top);

fImg.Refresh;

fVisible := Visible;

end;

end;

end;

// -------------------------------------------------------------------------

// Переключение между полной и компактной формой отображения

procedure TScope.SetCompact(Compact : boolean);

begin

if fVisible then RestoreBkGround();

fCompact := Compact;

CalcSRect();

SaveBkGround();

if fVisible then DrawScope();

end;

// -------------------------------------------------------------------------

// Переключение между прозрачной и непрозрачной формой отображения

procedure TScope.SetTransparent(Transparent : boolean);

begin

if fVisible then RestoreBkGround();

fTransparent := Transparent;

if fVisible then DrawScope();

end;

// -------------------------------------------------------------------------

// Изменить подпись единицы измерения

procedure TScope.SetMeasure(Measure : string);

begin

if fVisible then RestoreBkGround();

fMeasure := Measure;

CalcSRect();

SaveBkGround();

if fVisible then DrawScope();

end;

// -------------------------------------------------------------------------

// -------------------------------------------------------------------------

// Установка XBeg позиции индикатора на fImg

procedure TScope.SetXBeg(XBeg : integer);

begin

if fVisible then RestoreBkGround();

if (XBeg < 0) or (XBeg > fImg.Width) then Exit;

fXBeg := XBeg;

CalcSRect();

SaveBkGround();

if fVisible then DrawScope();

end;

// -------------------------------------------------------------------------

// Установка YBeg позиции индикатора на fImg

procedure TScope.SetYBeg(YBeg : integer);

begin

if fVisible then RestoreBkGround();

if (YBeg < 0) or (YBeg > fImg.Height) then Exit;

fYBeg := YBeg;

CalcSRect();

SaveBkGround();

if fVisible then DrawScope();

end;

// -------------------------------------------------------------------------

// Установка титульной подписи

procedure TScope.SetTitle (Title : string);

begin

if fVisible then RestoreBkGround();

fTitle := Title;

CalcSRect();

SaveBkGround();

if fVisible then DrawScope();

end;

// -------------------------------------------------------------------------

// Вычисление масштаба для графического поля

procedure TScope.CalcScale();

begin

fScale := 0;

if fMax <= fMin then Exit;

fScale := fCGW / (fMax - fMin);

end;

// -------------------------------------------------------------------------

// Преобразовать значение в пиксельную длину для графического поля

function TScope.GetGW(Value : extended) : integer;

begin

Result := 0;

if fScale > 0

then begin

if Value > fMax

then begin

Result := Round(fScale \* Abs(fMax - fMin));

Exit;

end;

if Value < fMin

then begin

Result := 0;

Exit;

end;

Result := Round(fScale \* Abs(Value - fMin));

end;

end;

// -------------------------------------------------------------------------

// Изменить нижнюю границу значений

procedure TScope.SetMin(Min : extended);

begin

if Min >= fMax then Exit;

fMin := Min;

fYZone := 0;

fRZone := 0;

CalcScale();

end;

// -------------------------------------------------------------------------

// Изменить верхнюю границу значений

procedure TScope.SetMax(Max : extended);

begin

if Max <= fMin then Exit;

fMax := Max;

fYZone := 0;

fRZone := 0;

CalcScale();

end;

// -------------------------------------------------------------------------

// Изменить длину поля графического индикатора

procedure TScope.SetCGW(GLen : integer);

begin

if fVisible then RestoreBkGround();

if GLen < cWIcon then fCGW := cWIcon;

fCGW := GLen;

CalcSRect();

CalcScale();

SaveBkGround();

if fVisible then DrawScope();

end;

// -------------------------------------------------------------------------

// Установить начало желтой зоны значений

procedure TScope.SetYZone (YZone : extended);

begin

fYZone := 0;

if (YZone >= fMin) and (YZone <= fMax)

then fYZone := YZone

else ShowMessage('Начало желтой зоны должно быть вутри диапазона индикатора');

end;

// -------------------------------------------------------------------------

// Установить начало красной зоны значений

procedure TScope.SetRZone (RZone : extended);

begin

fRZone := 0;

if fYZone <> 0

then begin

if (RZone > fYZone) and (RZone <= fMax)

then fRZone := RZone

else ShowMessage('Начало красной зоны должно быть больше начала желтой');

end

else ShowMessage('Красная зона может быть задано только после желтой');

end;

// -------------------------------------------------------------------------

// Отображение полей индикатора

procedure TScope.DrawScope();

var CFields : TColor;

CValue : TColor;

begin

if not fVisible then Exit;

// Цветовые параметрв полей и значений

CFields := RGB(200,220,240);

if fValue < 0

then begin

CValue := RGB(180,180,255);

if Abs(fValue) > fYZone then CValue := RGB(100,100,255);

if Abs(fValue) > fRZone then CValue := RGB(0,0,255);

end

else begin

CValue := clLime;

if fValue > fYZone then CValue := clYellow;

if fValue > fRZone then CValue := clRed;

end;

// Прорисовка индикаторов

with fImg.Canvas

do begin

// -----------------------------

if not fTransparent

then begin

// Прорисовать полное поле индикатора

Brush.Style := bsSolid;

Pen.Color := clWhite;

Brush.Color := CFields;

Rectangle(fSRect);

Pen.Color := clBlack;

MoveTo(fSRect.Left, fSRect.Bottom-1);

LineTo(fSRect.Right-1, fSRect.Bottom-1);

MoveTo(fSRect.Right-1, fSRect.Top);

LineTo(fSRect.Right-1, fSRect.Bottom);

end;

// -----------------------------

// Иконка индикатора

Brush.Color := CValue;

Brush.Style := bsSolid;

Ellipse(fIRect);

// -----------------------------

// Поле значения

Brush.Color := CFields;

Brush.Style := bsSolid;

FillRect(fVRect);

TextOut(fVRect.Left, fVRect.Top,

' ' + Format(cOutPic, [fValue])

+ ' ' + fMeasure + ' ' );

// -----------------------------

if not fCompact

then begin

// -----------------------------

// Рамка графического поля

Brush.Color := CFields;

Brush.Style := bsSolid;

Rectangle(fGRect);

// -----------------------------

// Значение графического поля

Brush.Style := bsSolid;

Brush.Color := CValue;

Rectangle(fGRect.Left, fGRect.Top,

fGRect.Left + Abs(GetGW(fValue)),

fGRect.Bottom);

// -----------------------------

// Заголовок индикатора

if fTitle <> ''

then begin

Brush.Color := CFields;

Brush.Style := bsSolid;

FillRect(fTRect);

TextOut(fTRect.Left, fTRect.Top, ' ' + fTitle + ' ');

end;

end;

end;

end;

// =========================================================================

// КОНЕЦ

// =========================================================================

end.

## unit Oscillograph01;

interface

uses Windows, Messages, Classes, Controls, SysUtils, Graphics, Dialogs,

StdCtrls, ExtCtrls, Menus;

// ====================================================================

// TItem1 (ВЕРСИЯ 1.0)

// Первичный класс для описания одной точки

// ====================================================================

type TItem1 = class(TObject)

protected

fNext : TItem1; // Ссылка на следующий экземпляр

fData : extended; // Данные, связанные с текущим экземпляром

end;

// ====================================================================

// TQueue1 (ВЕРСИЯ 1.0)

// Класс, который поддерживает очередь типа FIFO

// ====================================================================

type TQueue1 = class(TItem1)

private

fHead : TItem1; // Ссылка на голову очереди

fTail : TItem1; // Ссылка на хвост очереди

fCurrent : TItem1; // Указатель на текущий элемент

fCount : integer; // Текущее число элементов очереди

fMaxCount : integer; // Максимальное число элементов очереди

procedure InsertToTail(RqData : extended);

procedure CutFromHead();

procedure SetQueueMaxCount(RqCount : integer);

public

constructor Create();

procedure AddData (RqData : extended);

function SetCurrentItem(Cmd : char) : boolean;

function GetCurrentData () : extended;

procedure ClearQueue();

procedure Free();

property QueueMaxCount : integer read fMaxCount write SetQueueMaxCount;

end;

// ====================================================================

// TOscillograph

// Однолучевой осциллограф

// ====================================================================

type TOscillograph = class(TObject)

private

// -----------------------------

fTitle : string; // Заголовочная надпись

// -----------------------------

fPanelRect : Trect; // Габариты несущей панели

fPanel : TPanel; // Несущая панель

fDisplay : TImage; // Image для осциллограмы

fPM : TPopupMenu; // PopUpMenu для осциллограмы

fBeam1 : TQueue1; // FIFO для точек осциллограмы

fLbT : TLabel; // Label для заголовка

fLbV : TLabel; // Label для значений

// -----------------------------

// Управление диапазонами

fYMes : string; // Текст размерности данных осциллограмы

fYMax : extended; // Макс. абсолютная амплитуда на экране

fXMax : integer; // Макс. число точек в осциллограме

// -----------------------------

// Управление прорисовкой

fDigit : boolean; // Запрос цифровой осцилограммы

fdYShow : boolean; // Запрос отображения последнего приращения

fYPrev : extended; // Предшествующая амплитуда сигнала

fXCount : integer; // Счетчик прорисованных точек осциллограмы

// -----------------------------

// Линии сетки

fGridY : extended; // Шаг сетки по уровням Y

fGridX : integer; // Шаг сетки по уровням X

// -----------------------------

// Цветовые характеристики

fBeamColor : TColor; // Цвет луча

fFonColor : TColor; // Цвет фона экрана

fGridColor : TColor; // Цвет сетки на экране

// -----------------------------

// Пороговый контроль уровня

fAlarmOn : boolean; // Отображать зоны порогового контроля

fY1Alarm : extended; // Уровень порогового контроля 1

fY2Alarm : extended; // Уровень порогового контроля 2

fc1Alarm : TColor; // Цвет луча за поорогом 1

fc2Alarm : TColor; // Цвет луча за поорогом 2

// -----------------------------

// Поддержка property

procedure SetTitle(RqTitle : string);

procedure SetYMax (RqYMax : extended);

procedure SetXMax (RqXMax : integer);

// -----------------------------

// Обработчик событий на PopupMenu

procedure MenuClick(Sender : TObject);

// -----------------------------

// Прорисовать горизонтальные и вертикальные линии сетки

procedure ShowGrids();

// Очистить осциллограму

procedure ImageClear(RqColor : TColor);

// Вычислить пиксельное значение на оси X

function CalcX() : integer;

// Вычислить пиксельное значение на оси Y

function CalcY(RqData : extended) : integer;

// Прорисовать первую точку осциллограммы

procedure ShowFirstPoint(RqData : extended);

// Прорисовать очередную точку осциллограммы

procedure ShowNextPoint(RqData : extended);

public

// -----------------------------

// Указать форму или панель для размещения несущей панели

constructor Create(RqParent : TWinControl; RqPanelRect : TRect);

procedure Free;

// -----------------------------

// Добавить точку в осциллограму

procedure AddNewData (RqData : extended);

// Прорисовать текущую осциллограму

procedure ShowBeam();

// -----------------------------

// Титульная надпись

property Title : string read fTitle write SetTitle;

// Габариты несущей панели

property PanelRect : Trect read fPanelRect;

// Текст размерности данных осциллограмы

property YMes : string read fYMes write fYMes;

// Максимальная амплитуда значений

property YMax : extended read fYMax write SetYMax;

// Максимальное число точек на оси X

property XMax : integer read fXMax write SetXMax;

// Цифровой стиль осциллограмы

property Digit : boolean read fDigit write fDigit;

// Цвет луча осциллограмы

property BeamColor : TColor read fBeamColor write fBeamColor;

// Цвет фона осциллограмы

property FonColor : TColor read fFonColor write fFonColor;

// Цвет сетки осциллограмы

property GridColor : TColor read fGridColor write fGridColor;

// Шаг сетки по оси Y (в размерности входных данных)

property GridY : extended read fGridY write fGridY;

// Шаг сетки по оси X (в точках)

property GridX : integer read fGridX write fGridX;

// Уровень желтой зоны (в размерности входных данных)

property Y1Alarm : extended read fY1Alarm write fY1Alarm;

// Уровень красной зоны (в размерности входных данных)

property Y2Alarm : extended read fY2Alarm write fY2Alarm;

// Цвет луча в желтой зоне

property Alarm1Color : TColor read fc1Alarm write fc1Alarm;

// Цвет луча в красной зоне

property Alarm2Color : TColor read fc2Alarm write fc2Alarm;

end;

// ====================================================================

// ====================================================================

implementation

// ====================================================================

// ====================================================================

// ====================================================================

// TQueue1 (ВЕРСИЯ 1.0)

// Класс, который поддерживает очередь типа FIFO

// ====================================================================

// --------------------------------------------------------------------

// Создать элемент очереди, загрузить его данными и вставить в хвост очереди

procedure TQueue1.InsertToTail(RqData : extended);

var wp : TItem1;

begin

wp := TItem1.Create;

wp.fNext := nil;

wp.fData := RqData;

if not Assigned(fTail)

then begin // Вставка первого элемента в очередь

fTail := wp;

fHead := fTail;

fCurrent := fTail;

fCount := 1;

end

else begin // Вставка очередного элемента в очередь

fTail.fNext := wp;

fTail := wp;

fCount := fCount + 1;

end;

end;

// --------------------------------------------------------------------

// Удалить элемент из головы очереди

procedure TQueue1.CutFromHead();

var wp : TItem1;

begin

if not Assigned(fHead)

then begin

// Очередь пустая

fTail := nil;

fCurrent := nil;

fCount := 0;

end

else begin

// Сохранить ссылку на удаляемый элемент

wp := fHead;

if not Assigned(wp.fNext)

then begin // После выполнения операции очередь будет пустой

fHead := nil;

fTail := nil;

fCurrent := nil;

fCount := 0;

end

else begin // Отредактировать новое состояние очереди

fHead := wp.fNext;

// Откорректировать ссылку на текущий элемент

if fCurrent = wp then fCurrent := fHead;

fCount := fCount - 1;

if fCount < 1 then ShowMessage('Авария Queue');

end;

// Ликвидировать вырезанный элемент

wp.Free;

end;

end;

// --------------------------------------------------------------------

// Создать очередь

constructor TQueue1.Create();

begin

inherited Create;

fHead := nil; // Начальная инициализация

fTail := nil; // Начальная инициализация

fCurrent := nil; // Начальная инициализация

fCount := 0; // Начальная инициализация

fMaxCount := 10; // Значение по умолчанию

end;

// --------------------------------------------------------------------

// Добавить новую точку в осциллограмму

procedure TQueue1.AddData (RqData : extended);

begin

InsertToTail(RqData);

if fCount > fMaxCount then CutFromHead();

end;

// --------------------------------------------------------------------

// Установить указатель текущего элемента

function TQueue1.SetCurrentItem(Cmd : char) : boolean;

begin

Result := False;

if not Assigned(fHead)

then begin

fCurrent := nil;

end

else begin

// Для непустой очереди

case Cmd of

'H' : begin // Установка указателя на начало (голову) очереди

fCurrent := fHead;

Result := True;

end;

'N' : begin // Установка указателя на следующий элемент очереди

if Assigned(fCurrent)

then begin

fCurrent := fCurrent.fNext;

if Assigned(fCurrent) then Result := True;

end

else fCurrent := nil;

end;

end;

end;

end;

// --------------------------------------------------------------------

// Получить данные из текущего элемента очереди

function TQueue1.GetCurrentData () : extended;

begin

Result := 0;

if Assigned(fCurrent) then Result := fCurrent.fData;

end;

// --------------------------------------------------------------------

// Удалить из очереди все элементы

procedure TQueue1.ClearQueue();

begin

while Assigned(fHead) do CutFromHead();

end;

// --------------------------------------------------------------------

// Установить максимальное число элементов очереди до начала сдвига

procedure TQueue1.SetQueueMaxCount(RqCount : integer);

begin

if RqCount > 0

then begin

ClearQueue();

fMaxCount := RqCount;

end;

end;

// --------------------------------------------------------------------

// Удалить очередь

procedure TQueue1.Free();

begin

ClearQueue();

inherited Free();

end;

// ====================================================================

// TOscillograph

// Однолучевой осциллограф

// ====================================================================

const MinPanelW = 200; // Мин. ширина несущей панели

MinPanelH = 120; // Мин. высота несущей панели

// --------------------------------------------------------------------

// Создать осциллограф

constructor TOscillograph.Create(RqParent : TWinControl; RqPanelRect : TRect);

var MenuItems : array of TMenuItem;

begin

inherited Create;

fPanelRect := RqPanelRect;

with RqPanelRect do

begin

if Left < 0 then fPanelRect.Left := 0;

if Top < 0 then fPanelRect.Top := 0;

if Right - fPanelRect.Left < MinPanelW

then fPanelRect.Right := fPanelRect.Left + MinPanelW;

if Bottom - fPanelRect.Top < MinPanelH

then fPanelRect.Bottom := fPanelRect.Top + MinPanelH;

end;

// -----------------------------

// Создание несущей панели

fPanel := TPanel.Create(RqParent);

fPanel.Parent := RqParent;

fPanel.Visible := False;

with fPanelRect do fPanel.SetBounds(Left,Top,Right-Left,Bottom-Top);

// -----------------------------

// Создание Image для осциллограмы

fDisplay := TImage.Create(fPanel);

fDisplay.Parent := fPanel;

fDisplay.SetBounds(2,20,fPanel.Width-4,fPanel.Height-40);

// -----------------------------

// Создание Label для отображения значений осциллограмы

fLbV := TLabel.Create(fPanel);

fLbV.Parent := fPanel;

fLbV.SetBounds(2,fPanel.Height-18,fPanel.Width-4,16);

fLbV.AutoSize := False;

// -----------------------------

// Создание Label для заголовка осциллограмы

fLbT := TLabel.Create(fPanel);

fLbT.Parent := fPanel;

fLbT.SetBounds(2,3,fPanel.Width-4,16);

fLbT.AutoSize := False;

fLbT.Alignment := taCenter;

// -----------------------------

fPanel.Visible := True;

// -----------------------------

// Создать и подключить PopupMenu

SetLength(MenuItems, 3);

MenuItems[0]:= NewItem('Пороговые зоны', TextToShortCut(''),

True, True, MenuClick, 0, 'ITEM1');

MenuItems[0].Tag := 1;

MenuItems[1]:= NewItem('Цифровой вид', TextToShortCut(''),

True, True, MenuClick, 0, 'ITEM2');

MenuItems[1].Tag := 2;

MenuItems[2]:= NewItem('Приращение', TextToShortCut(''),

False, True, MenuClick, 0, 'ITEM3');

MenuItems[2].Tag := 3;

fPM := NewPopupMenu(fDisplay, 'Menu',

paLeft, True, MenuItems);

SetLength(MenuItems,0);

fDisplay.PopupMenu := fPM;

// -----------------------------

// Создать FIFO для хранения данных луча

fBeam1 := TQueue1.Create;

fXMax := 50;

fBeam1.QueueMaxCount := fXMax;

// -----------------------------

// Максимальная амплитуда значений

fYMax := 100;

// -----------------------------

// Параметры сетки по умолчанию

fGridY := fYMax / 5;

fGridX := fXMax div 5;

fDigit := True;

// -----------------------------

// Установки цвета по умолчанию

fBeamColor := clLime; // RGB(64,64,128);

fFonColor := RGB(32,32,32); // clBlack;

fGridColor := clGray;

ImageClear(fFonColor);

// -----------------------------

// Пороговый контроль

fAlarmOn := True; // Отображать зоны порогового контроля

fY1Alarm := fYMax \* 0.7; // Уровень порогового контроля 1

fY2Alarm := fYMax \* 0.9; // Уровень порогового контроля 2

fc1Alarm := clYellow; // RGB(180,180,0); // Цвет луча за поорогом 1

fc2Alarm := clRed; // RGB(200,0,0); // Цвет луча за поорогом 1

end;

// --------------------------------------------------------------------

procedure TOscillograph.Free;

begin

if Assigned(fBeam1) then fBeam1.Free;

if Assigned(fPM) then fPM.Free;

if Assigned(fLbV) then fLbV.Free;

if Assigned(fLbT) then fLbT.Free;

if Assigned(fDisplay) then fDisplay.Free;

if Assigned(fPanel) then fPanel.Free;

inherited Free;

end;

// --------------------------------------------------------------------

// --------------------------------------------------------------------

// Обработчик событий на меню

procedure TOscillograph.MenuClick(Sender : TObject);

var Item : TMenuItem;

begin

Item := TMenuItem(Sender);

case Item.Tag of

1 : begin // Раскраска зон

if not Item.Checked

then begin

fAlarmOn := True;

Item.Checked := True;

end

else begin

fAlarmOn := False;

Item.Checked := False;

end;

end;

2 : begin // Цифровой вид

if not Item.Checked

then begin

fDigit := True;

Item.Checked := True;

end

else begin

fDigit := False;

Item.Checked := False;

end;

end;

3 : begin // Приращение

if not Item.Checked

then begin

fdYShow := True;

Item.Checked := True;

end

else begin

fdYShow := False;

Item.Checked := False;

end;

end;

else begin end;

end;

end;

// --------------------------------------------------------------------

procedure TOscillograph.SetTitle(RqTitle : string);

begin

if Assigned(fLbT) then fLbT.Caption := RqTitle;

end;

// --------------------------------------------------------------------

// Изменение максимальной абсолютной амплитуду на экране

procedure TOscillograph.SetYMax(RqYMax : extended);

begin

if RqYMax < 1e-10

then fYMax := 1e-10

else fYMax := RqYMax;

ImageClear(fFonColor);

end;

// --------------------------------------------------------------------

// Изменение максимального числа точек в осциллограме

procedure TOscillograph.SetXMax(RqXMax : integer);

begin

if RqXMax < 8

then fBeam1.QueueMaxCount := 8

else fBeam1.QueueMaxCount := RqXMax;

ImageClear(fFonColor);

end;

// --------------------------------------------------------------------

// Прорисовать горизонтальные и вертикальные линии сетки

procedure TOscillograph.ShowGrids();

var wY : extended;

wX : integer;

begin

with fDisplay.Canvas

do begin

// Сетка по уровням Y

// Нулевая линия

Pen.Color := (not fFonColor) and $00FFFFFF;

MoveTo(0, CalcY(0));

LineTo(fDisplay.Width,CalcY(0));

// Линии сетки

Pen.Color := fGridColor;

// Линии сетки ниже нулевой

wY := - fGridY;

repeat

MoveTo(0, CalcY(wY));

LineTo(fDisplay.Width,CalcY(wY));

wY := wY - fGridY;

until (wY < - fYMax);

// Линии сетки выше нулевой

wY := fGridY;

repeat

MoveTo(0, CalcY(wY));

LineTo(fDisplay.Width,CalcY(wY));

wY := wY + fGridY;

until (wY > fYMax);

// Сетка по уровням X

wX := 0;

repeat

MoveTo(wX, 0,);

LineTo(wX, fDisplay.Height);

wX := wX + fGridX;

until (wX > fDisplay.Width);

end;

end;

// --------------------------------------------------------------------

// Очистка экрана осциллографа выбранным цветом

procedure TOscillograph.ImageClear(RqColor : TColor);

begin

with fDisplay.Canvas do

begin

Brush.Color := RqColor;

Brush.Style := bsSolid;

FillRect(Rect(0,0, fDisplay.Width, fDisplay.Height));

end;

fXCount := 0;

end;

// --------------------------------------------------------------------

// Вычислить X - координату на экране

function TOscillograph.CalcX() : integer;

begin

Result := Trunc((fDisplay.Width / fBeam1.QueueMaxCount) \* fXCount);

end;

// --------------------------------------------------------------------

// Вычислить Y - координату на экране

function TOscillograph.CalcY(RqData : extended) : integer;

begin

Result := (fDisplay.Height div 2);

Result := Result - Trunc((Result / fYMax) \* RqData);

end;

// --------------------------------------------------------------------

// Отобразить первую точку осциллограмыы

procedure TOscillograph.ShowFirstPoint(RqData : extended);

begin

// Начальная установка осциллограмы

ImageClear(fFonColor);

ShowGrids();

// Начальное позиционирование осциллограммы

fDisplay.Canvas.MoveTo(CalcX(),CalcY(RqData));

fYPrev := RqData;

fXCount := fXCount + 1;

end;

// --------------------------------------------------------------------

// Отобразить очередную точку осциллограмыы

procedure TOscillograph.ShowNextPoint(RqData : extended);

var fwBeamColor : TColor;

begin

fwBeamColor := fBeamColor;

if fAlarmOn

then begin

// Усвтановка цвета при пороговом контроле

if Abs(RqData) >= fY1Alarm then fwBeamColor := fc1Alarm;

if Abs(RqData) >= fY2Alarm then fwBeamColor := fc2Alarm;

end;

fDisplay.Canvas.Pen.Color := fwBeamColor;

if fDigit

then fDisplay.Canvas.LineTo(CalcX(),CalcY(fYPrev));

fDisplay.Canvas.LineTo(CalcX(),CalcY(RqData));

fYPrev := RqData;

fXCount := fXCount + 1;

end;

// --------------------------------------------------------------------

// Прикладные PUBLIC методы

// --------------------------------------------------------------------

// Добавить новую точку в осциллограму

procedure TOscillograph.AddNewData(RqData : extended);

begin

if Assigned (fBeam1) then fBeam1.AddData(RqData);

if Assigned (fLbV)

then begin

if not fdYShow

then fLbV.Caption := ' Текущее значение : '

+ Format('%6.3f',[RqData]) + ' ' + YMes

else fLbV.Caption := ' Текущее приращение : '

+ Format('%6.3f',[RqData-fYPrev]) + ' ' + YMes;

end;

end;

// --------------------------------------------------------------------

// Отобразить все точки осциллограмыы

procedure TOscillograph.ShowBeam();

begin

if Assigned (fBeam1)

then begin

if fBeam1.SetCurrentItem('H')

then begin

ShowFirstPoint(fBeam1.GetCurrentData());

while fBeam1.SetCurrentItem('N')

do ShowNextPoint(fBeam1.GetCurrentData());

end;

end;

end;

// =========================================================================

// КОНЕЦ

// =========================================================================

end.

## unit Service01;

// НАБОР СЕРВИСОВ ДЛЯ ОБСЛУЖИВАНИЯ РАБОТ С ВЕКТОРАМИ

// И РАЗЛИЧНЫХ ОПЕРАЦИЙ НА IMAGE

// РЕДАКЦИЯ ОТ 06.02.2017

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,

Dialogs;

// =========================================================================

// -------------------------------------------------------------------------

// Описание двухмерного вектора

type TVector2D = record

X : extended;

Y : extended;

end;

// -------------------------------------------------------------------------

// Описание трехмерного вектора

type TVector3D = record

X : extended;

Y : extended;

Z : extended;

end;

// -------------------------------------------------------------------------

// Описание двухмерного вектора в декартовых и полярных координатах

type TVector2DP = record

X : extended;

Y : extended;

Len : extended;

Ang : extended;

end;

// -------------------------------------------------------------------------

// -------------------------------------------------------------------------

// Суммирование двух векторов типа TVector2D

function SumVector2D(const V1, V2 : TVector2D): TVector2D;

// Вычитание (V1 - V2) векторов типа TVector2D

function SubVector2D(const V1, V2 : TVector2D): TVector2D;

// -------------------------------------------------------------------------

// Суммирование двух векторов типа TVector3D

function SumVector3D(const V1, V2 : TVector3D): TVector3D;

// Вычитание (V1 - V2) векторов типа TVector3D

function SubVector3D(const V1, V2 : TVector3D): TVector3D;

// -------------------------------------------------------------------------

// Переход от декартовых координат к полярным

procedure DecartToPolar (var V : TVector2DP); register;

// -------------------------------------------------------------------------

// Переход от полярных координат к декартовым

procedure PolarToDecart(var V : TVector2DP); register;

// =========================================================================

// -------------------------------------------------------------------------

// Контроль попадания точки RqX,RqY внутрь RqRect

function IsXYInRect(const RqRect : TRect; RqX, RqY : integer): boolean;

// Контроль попадания точки RqPoint внутрь RqRect

// см. PtInRect

function IsPointInRect(const RqRect : TRect; RqPoint : TPoint): boolean;

// Контроль попадания RqRect внутрь RqMainRect

function IsRectInMainRect(const RqMainRect, RqRect : TRect): boolean;

// Контроль пустого RqRect (один пиксель считается пустым Rect)

// см. IsRectEmpty

function IsEmptyRect(const RqRect : TRect): boolean;

// Контроль пересечения двух RqRect (один пиксель считается не пустым Rect)

// см. IntersectRect

function IsCrossRect(const RqRect1, RqRect2 : TRect): boolean;

// Контроль равенства габаритов TBmp габаритам RqRect

function IsBmpEqualRect(RqBmp : TBitMap; RqRect : TRect) : boolean;

// Переместить RqRect на начальные координаты RqX, RqY

function MoveRectangle (RqRect : TRect; RqX, RqY : integer) : TRect;

// Сохранить в Bitmap область из FromBitMap указанную FomRect

function SaveBitMap(FromBitMap, Bitmap : TBitmap;

FomRect : TRect) : boolean;

// Восстановить из Bitmap область в ToBitMap указанную начальными координатами

function RestoreBitMap(BitMap : TBitmap; // BitMap источник

ToBitMap : TBitmap; // BitMap приемник

Xb, Yb : integer) // Начальные координаты в приемнике

: boolean;

// =========================================================================

// =========================================================================

implementation

// =========================================================================

// =========================================================================

// НАБОР СЕРВИСОВ ДЛЯ ОБСЛУЖИВАНИЯ РАБОТ С ВЕКТОРАМИ

// =========================================================================

// -------------------------------------------------------------------------

// Суммирование двух векторов типа TVector2D

// 06.02.2017

function SumVector2D(const V1, V2 : TVector2D): TVector2D;

begin

Result.X := V1.X + V2.X;

Result.Y := V1.Y + V2.Y;

end;

// -------------------------------------------------------------------------

// Вычитание (V1 - V2) векторов типа TVector2D

// 06.02.2017

function SubVector2D(const V1, V2 : TVector2D): TVector2D;

begin

Result.X := V1.X - V2.X;

Result.Y := V1.Y - V2.Y;

end;

// -------------------------------------------------------------------------

// Суммирование двух векторов типа TVector3D

// 06.02.2017

function SumVector3D(const V1, V2 : TVector3D): TVector3D;

begin

Result.X := V1.X + V2.X;

Result.Y := V1.Y + V2.Y;

Result.Y := V1.Z + V2.Z;

end;

// -------------------------------------------------------------------------

// Вычитание (V1 - V2) векторов типа TVector3D

// 06.02.2017

function SubVector3D(const V1, V2 : TVector3D): TVector3D;

begin

Result.X := V1.X - V2.X;

Result.Y := V1.Y - V2.Y;

Result.Y := V1.Z - V2.Z;

end;

// -------------------------------------------------------------------------

// 12.02.2017

// Переход от декартовых координат к полярным

procedure DecartToPolar (var V : TVector2DP); register;

asm

FINIT // Сброс регистров и флагов сопроцессора (FPU)

// Стек регистров FPU после выполнения команды

FLD TVector2DP.Y [EAX] // (V.Y);

FLD TVector2DP.X [EAX] // (V.X); (V.Y);

// Вычисление фазы в радианах

FLD ST(1) // (V.Y); (V.X); (V.Y)

FLD ST(1) // (V.X); (V.Y); (V.X); (V.Y);

FPATAN // ArcTan((V.X); (V.Y)); (V.X); (V.Y);

FSTP TVector2DP.Ang [EAX] // (V.X); (V.Y)

// Вычисление модуля

FLD ST(0) // (V.X); (V.X); (V.Y);

FMUL // (V.X \* V.X); (V.Y);

FXCH ST(1) // (V.Y); (V.X \* V.X);

FLD ST(0) // (V.Y); (V.Y); (V.X \* V.X)

FMUL // (V.Y \* V.Y); (V.X \* V.X);

FADD // ((V.Y \* V.Y) + (V.X \* V.X));

FSQRT // sqrt((V.Y \* V.Y) + (V.X \* V.X));

FSTP TVector2DP.Len [EAX] // [empty]

FWAIT

end;

// -------------------------------------------------------------------------

// 12.02.2017

// Переход от полярных координат к декартовым

procedure PolarToDecart(var V : TVector2DP); register;

asm

FINIT // Сброс регистров и флагов сопроцессора (FPU)

// Стек регистров FPU после выполнения команды

FLD TVector2DP.Len [EAX] // (V.Len)

FLD TVector2DP.Ang [EAX] // (V.Len)

FSINCOS // (cos(V.Ang)); (sin(V.Ang)); (V.Len)

FMUL ST, ST(2) // (V.Len \* cos(V.Ang)); (sin(V.Ang)); (V.Len)

FSTP TVector2DP.X [EAX] // (sin(V.Ang)); (V.Len)

FMUL // (V2P.Len \* sin(V2P.Ang))

FSTP TVector2DP.Y [EAX] // [empty]

FWAIT

end;

// =========================================================================

// НАБОР СЕРВИСОВ ДЛЯ ОБСЛУЖИВАНИЯ РАЗЛИЧНЫХ ОПЕРАЦИЙ НА IMAGE

// =========================================================================

// -------------------------------------------------------------------------

// 29.01.2017

// Контроль попадания точки RqX,RqY внутрь RqRect

function IsXYInRect(const RqRect : TRect; RqX, RqY : integer): boolean;

begin

Result := (RqX >= RqRect.Left) and (RqX <= RqRect.Right) and

(RqY >= RqRect.Top) and (RqY <= RqRect.Bottom);

end;

// -------------------------------------------------------------------------

// Контроль попадания точки RqPoint внутрь RqRect

// см. PtInRect

// 29.01.2017

function IsPointInRect(const RqRect : TRect; RqPoint : TPoint): boolean;

begin

Result := (RqPoint.X >= RqRect.Left) and

(RqPoint.X <= RqRect.Right) and

(RqPoint.Y >= RqRect.Top) and

(RqPoint.Y <= RqRect.Bottom);

end;

// -------------------------------------------------------------------------

// Контроль попадания RqRect внутрь RqMainRect

// 29.01.2017

function IsRectInMainRect(const RqMainRect, RqRect : TRect): boolean;

begin

Result := IsXYInRect(RqMainRect, RqRect.Left, RqRect.Top) and

IsXYInRect(RqMainRect, RqRect.Right, RqRect.Bottom);

end;

// -------------------------------------------------------------------------

// Контроль пустого RqRect (один пиксель считается пустым Rect)

// см. IsRectEmpty

// 29.01.2017

function IsEmptyRect(const RqRect : TRect): boolean;

begin

Result := IsRectEmpty(RqRect);

end;

// -------------------------------------------------------------------------

// Контроль пересечения двух RqRect (один пиксель считается не пустым Rect)

// см. IntersectRect

// 29.01.2017

function IsCrossRect(const RqRect1, RqRect2 : TRect): boolean;

begin

Result := IsXYInRect(RqRect1, RqRect2.Left, RqRect2.Top) or

IsXYInRect(RqRect1, RqRect2.Right, RqRect2.Bottom) or

IsXYInRect(RqRect1, RqRect2.Right, RqRect2.Top) or

IsXYInRect(RqRect1, RqRect2.Left, RqRect2.Bottom) or

IsXYInRect(RqRect2, RqRect1.Left, RqRect1.Top) or

IsXYInRect(RqRect2, RqRect1.Right, RqRect1.Bottom) or

IsXYInRect(RqRect2, RqRect1.Right, RqRect1.Top) or

IsXYInRect(RqRect2, RqRect1.Left, RqRect1.Bottom);

end;

// -------------------------------------------------------------------------

// Контроль равенства габаритов TBmp габаритам RqRect

// 29.01.2017

function IsBmpEqualRect(RqBmp : TBitMap; RqRect : TRect) : boolean;

begin

Result := False;

if Assigned(RqBmp)

then begin

if RqBmp.Height <> (RqRect.Bottom - RqRect.Top) + 1 then Exit;

if RqBmp.Width <> (RqRect.Right - RqRect.Left) + 1 then Exit;

Result := True;

end;

end;

// -------------------------------------------------------------------------

// Переместить RqRect на начальные координаты RqX, RqY

// 29.01.2017

function MoveRectangle (RqRect : TRect; RqX, RqY : integer) : TRect;

begin

Result := Rect(RqX, RqY,

RqX + (RqRect.Right - RqRect.Left),

RqY + (RqRect.Bottom - RqRect.Top ));

end;

// -------------------------------------------------------------------------

// Сохранить в Bitmap область из FromBitMap указанную FomRect

// 29.01.2017

function SaveBitMap(FromBitMap, Bitmap : TBitmap;

FomRect : TRect) : boolean;

// ------------------------------------

// SysUtils:

// PByteArray = ^TByteArray;

// TByteArray = array[0..32767] of Byte;

// ------------------------------------

var

RowBeg, ColBeg : Integer;

RowEnd, ColEnd : Integer;

PFrom, PTo : PByteArray;

RowFrom, RowTo : Integer;

ColFrom, ColTo : Integer;

begin

Result := False;

if not (FromBitMap.PixelFormat = pf24bit) then Exit;

// Контроль геометрии наложения FomRec

// 1) FomRect за пределами FromBitMap

if FomRect.Top >= FromBitMap.Height then Exit;

if FomRect.Left >= FromBitMap.Width then Exit;

// 2) Нормировка диапазона строк

RowBeg := FomRect.Top;

if RowBeg < 0 then RowBeg := 0;

RowEnd := FomRect.Bottom;

if RowEnd > FromBitMap.Height - 1 then RowEnd := FromBitMap.Height - 1;

// 3) Нормировка диапазона столбцов

ColBeg := FomRect.Left;

if ColBeg < 0 then ColBeg := 0;

ColEnd := FomRect.Right;

if ColEnd > FromBitMap.Width - 1 then ColEnd := FromBitMap.Width;

// 4) Проверка на пустой FomRec

if RowEnd < RowBeg then Exit;

if ColEnd < ColBeg then Exit;

// Начинаем копирование

try

Bitmap.PixelFormat := pf24bit;

Bitmap.Height := RowEnd - RowBeg + 1;

Bitmap.Width := ColEnd - ColBeg + 1;

RowTo := 0;

for RowFrom := RowBeg to RowEnd do

begin

// Представим строку BitMap через указатель на массив байтов

PFrom := FromBitMap.ScanLine[RowFrom];

PTo := Bitmap.ScanLine[RowTo];

// Побайтовое копирование. Порядок байтов в пикселе pf24bit : B,G,R

ColTo := 0;

for ColFrom := ColBeg \* 3 to ColEnd \* 3

do begin

PTo^[ColTo] := PFrom^[ColFrom];

Inc(ColTo);

end;

Inc(RowTo);

end;

Result := True;

except

ShowMessage('Не могу сохранить BitMap');

end;

end; // of function

// -------------------------------------------------------------------------

// Восстановить из Bitmap область в ToBitMap указанную начальными координатами

// 29.01.2017

function RestoreBitMap(BitMap : TBitmap; // BitMap источник

ToBitMap : TBitmap; // BitMap приемник

Xb, Yb : integer) // Начальные координаты в приемнике

: boolean;

var

RowBeg, ColBeg : Integer;

RowEnd, ColEnd : Integer;

PFrom, PTo : PByteArray;

RowFrom, RowTo : Integer;

ColFrom, ColTo : Integer;

begin

Result := False;

if not (ToBitmap.PixelFormat = pf24bit) then Exit;

// Контроль геометрии наложения FomRec

// 1) Xb, Yb за пределами ToBitMap

if Yb >= ToBitmap.Height then Exit;

if Xb >= ToBitmap.Width then Exit;

// 2) Нормировка диапазона строк

RowBeg := Yb;

if Yb < 0

then begin

RowBeg := 0;

RowEnd := BitMap.Height - 1;

end

else RowEnd := Yb + BitMap.Height - 1;

if RowEnd > ToBitMap.Height - 1 then RowEnd := ToBitMap.Height - 1;

// 3) Нормировка диапазона столбцов

ColBeg := Xb;

if Xb < 0

then begin

ColBeg := 0;

ColEnd := BitMap.Width - 1;

end

else ColEnd := Xb + BitMap.Width - 1;

if ColEnd > ToBitMap.Width - 1 then ColEnd := ToBitMap.Width;

// 4) Проверка на пустой ToRec

if RowEnd < RowBeg then Exit;

if ColEnd < ColBeg then Exit;

// Начинаем копирование

try

RowFrom := 0;

for RowTo := RowBeg to RowEnd do

begin

// Представим каждую строку BitMap как

// указатель на массив байтов

PFrom := BitMap.ScanLine[RowFrom];

PTo := ToBitmap.ScanLine[RowTo];

// побайтовое копирование изображения

ColFrom := 0;

for ColTo := ColBeg \* 3 to ColEnd \* 3

do begin

PTo^[ColTo] := PFrom^[ColFrom];

Inc(ColFrom);

end;

Inc(RowFrom);

end;

Result := True;

except

ShowMessage('Не могу восстановить BitMap');

end;

end; // of function

// =========================================================================

// КОНЕЦ

// =========================================================================

end.

## unit ImgLoadSave01;

(\*

========================================================================

ПОДСИСТЕМА "ГРАФИЧЕСКИХ СЕРВИСОВ" GraphEngine.

ИНСТРУМЕНТАЛЬНЫЙ ЮНИТ ЗАГРУЗКИ СОХРАНЕНИЯ ИЗОБРАЖЕНИЙ.

ТРЕБОВАНИЕ: BitMap.PixelFormat = pf24bit

========================================================================

Определяет сервисы посредством объектов TImageDesc и TLoadsaveImage:

1) Выполняет обмен изображениями (JPEG и BMP) между Image и файлами.

2) Выполняет обмен изображениями между Image и Clipboard.

========================================================================

(С) Кафедра СУЛА, Иститут аэронавигации, НАУ, Киев.

========================================================================

\*)

interface

uses

// Системные юниты

Windows, Classes, Controls, Graphics, Dialogs, Forms,

SysUtils, ExtDlgs, ComCtrls, ExtCtrls, JPEG, Clipbrd;

// ------------------------------------------------------------------------

// Формат изображения в файле

const FileFormatJPEG = 1; // Код графического формата JPEG

FileFormatBMP = 2; // Код графического формата BMP

ClipboardFormat = 128; // Код графического формата BitMap Clipboard

FileFormatUnkn = 255; // Код неизвестного графического формата

// ------------------------------------------------------------------------

// ------------------------------------------------------------------------

type TLoadSaveImage = class(Tobject)

private

fInitDir : string; // Директория последней загрузки

// СЕГМЕНТ ДИНАМИЧЕСКИ СОЗДАВАЕМЫХ ОБЪЕКТОВ

fLoadDlg : TOpenPictureDialog; // ОБЪЕКТ диалога загрузки

fSaveDlg : TSavePictureDialog; // ОБЪЕКТ диалога сохранения

// СЕГМЕНТ ССЫЛОК НА АРЕНДУЕМЫЕ КОМПОНЕНТЫ

fProgressBar : TProgressBar; // Ссылка на ProgressBar

fMsgPanel : TStatusPanel; // Ссылка на StatusPanel

// СЕГМЕНТ ПАРАМЕТРОВ для JPEG

fJpegQuality : byte; // CompressionQuality для JPEG

fJpegProgressive : boolean; // Прогрессивное кодирование JPEG

// ПРИВАТНЫЕ МЕТОДЫ КЛАССА

// Проверка допустимого пиксельного формата

function VerifyPixFormat (RqBitMap : TBitMap) : boolean;

// Получить идентификатор пиксельного формата

function GetIdPixFormat (RqBitMap : TBitMap) : string;

// Загрузка картинки и ее преобразование в BitMap

function LoadJpegPicFromFile (RqFileName : string;

RqImage : TImage) : boolean;

// Загрузка BMP файла в Image.Pictures.BitMap

function LoadBmpPicFromFile (RqFileName : string;

RqImage : TImage) : boolean;

// Преобразование Image.Pictures.BitMap в JPEG и сохранение в файл

function SaveJpegToFile (RqFileName : string;

RqImage : TImage) : boolean;

// Сохранение изображения из Image.Picture.Bitmap в файл

function SaveBmpToFile (RqFileName : string;

RqImage : TImage) : boolean;

public

// КОНСТРУКТОР и ДЕСТРУКТОР ОБЪЕКТА

constructor Create (RqProgressBar : TProgressBar; // Ссылка на ProgressBar

RqMsgPanel : TStatusPanel); // Ссылка на StatusPanel

// ДЕСТРУКТОР ОБЪЕКТА

procedure Free;

// -------------------------------------------------

// ОПЕРАЦИИ Image - File

// Загрузка изображения из файла в Image

function LoadImgFromFile (RqFileName : string;

RqImg : TImage) : boolean;

// Диалоговая загрузка изображения из файла в Image

function DlgLoadImgFromFile (RqImg : TImage) : boolean;

// -------------------------------------------------

// Сохранение изображения из Image в файл

function SaveImgToFile (RqFileName : string;

RqImg : TImage) : boolean;

// Диалоговое сохранение изображения из Image в файл

function DlgSaveImgToFile (RqImg : TImage) : boolean;

// -------------------------------------------------

// ОПЕРАЦИИ Image - Clipboard

// Копировать содержимое Image в Clipboard

function CopyImageToClipboard(RqImg : TImage) : boolean;

// Вставка из Clipboard в Image

function PasteClipboardToImage(RqImage : TImage) : boolean;

// -------------------------------------------------

// СВОЙСТВА

// Переустановка параметров для JPEG

property JpegQuality : byte read fJpegQuality

write fJpegQuality;

property JpegProgressive : boolean read fJpegProgressive

write fJpegProgressive;

end;

// ========================================================================

// ========================================================================

implementation

// ========================================================================

// ========================================================================

// Строка фильтра СТРОГО связана с константами

// FileFormatJPEG и FileFormatBMP в методе DlgSaveImgToFile

// (НЕ ИЗМЕНЯТЬ!)

const DialogFilter = 'JPEG (\*.jpg;\*.jpeg)|\*.jpg;\*.jpeg | BMP (\*.bmp)|\*.bmp';

// ========================================================================

// КОНСТРУКТОР и ДЕСТРУКТОР ОБЪЕКТОВ

// ========================================================================

// КОНСТРУКТОР ОБЪЕКТОВ

constructor TLoadsaveImage.Create

(RqProgressBar : TProgressBar; // Ссылка на ProgressBar

RqMsgPanel : TStatusPanel); // Ссылка на StatusPanel;

begin

inherited Create;

// Динамисеское создание компонента OpenPictureDialog

fLoadDlg := TOpenPictureDialog.Create(nil);

with fLoadDlg do

begin

Filter := DialogFilter;

FilterIndex := FileFormatJPEG;

Ctl3D := True;

end;

fInitDir := ''; // Директория последней загрузки

// Динамисеское создание компонента SavePictureDialog

fSaveDlg := TSavePictureDialog.Create(nil);

with fSaveDlg do

begin

Filter := DialogFilter;

FilterIndex := FileFormatJPEG;

Ctl3D := True;

end;

// Наилучшие параметры качества для JPEG

fJpegQuality := 100;

fJpegProgressive := False;

end;

// ------------------------------------------------------------------------

// ДЕСТРУКТОР ОБЪЕКТОВ

procedure TLoadsaveImage.Free;

begin

fLoadDlg.FreeOnRelease;

fSaveDlg.FreeOnRelease;

inherited Free;

end;

// ========================================================================

// PUBLIC ПРОЦЕДУРЫ И МЕТОДЫ

// ========================================================================

// ------------------------------------------------------------------------

// Загрузка изображения из файла в Image

function TLoadsaveImage.LoadImgFromFile (RqFileName : string;

RqImg : TImage) : boolean;

var wFileName : string;

wExt : string;

begin

Result := False;

wFileName := Trim(RqFileName);

if wFileName = '' then Exit;

if not FileExists(wFileName)

then begin

MessageDlg('НЕ НАЙДЕН ФАЙЛ С ИМЕНЕМ : '

+ #13#10 + wFileName,

mtError, [mbOk], 0);

Exit;

end;

wExt:= UpperCase(ExtractFileExt(wFileName));

fInitDir := ExtractFileDir(wFileName);

if (wExt = '.JPG') or (wExt = '.JPEG')

then Result := LoadJpegPicFromFile (wFileName, RqImg)

else if (wExt = '.BMP')

then Result := LoadBmpPicFromFile (wFileName, RqImg)

else MessageDlg('НЕ ПОДДЕРЖИВАЕМЫЙ ФОРМАТ : ' + wExt,

mtError, [mbOk], 0);

end;

// ------------------------------------------------------------------------

// Диалоговая загрузка изображения из файла в Image

function TLoadsaveImage.DlgLoadImgFromFile (RqImg : TImage) : boolean;

begin

Result := False;

if fInitDir <> '' then fLoadDlg.InitialDir := fInitDir;

if fLoadDlg.Execute

then begin

// Сохранить директорию последней загрузки

fInitDir := ExtractFileDir(fLoadDlg.FileName);

// Загрузка изображения из файла в Image

Result := LoadImgFromFile (fLoadDlg.FileName, RqImg);

end;

end;

// ------------------------------------------------------------------------

// Сохранение изображения из Image в файл

function TLoadsaveImage.SaveImgToFile (RqFileName : string;

RqImg : TImage) : boolean;

var wFileName : string; // Буфер имени файла

wExt : string; // Буфер расширения файла

wURep : word; // Буфер ответа пользователя

begin

Result := False;

wFileName := Trim(RqFileName);

wExt := UpperCase(ExtractFileExt(wFileName));

if not ((wExt = '.JPG') or (wExt = '.JPEG') or (wExt = '.BMP'))

then begin

MessageDlg('НЕ ПОДДЕРЖИВАЕМЫЙ ФОРМАТ : ' + wExt,

mtError, [mbOk], 0);

Exit;

end;

if FileExists(wFileName)

then begin

wURep := MessageDlg('Файл с именем : '

+ #13#10 + wFileName

+ #13#10 + 'Уже существует. Заменить его?',

mtInformation,[mbYes,mbNo],0);

// Перезаписать существующий файл

if wURep = mrYes

then begin

if (wExt = '.JPG') or (wExt = '.JPEG')

then Result := SaveJpegToFile (wFileName, RqImg)

else if (wExt = '.BMP')

then Result := SaveBmpToFile (wFileName, RqImg);

end;

end

else begin

// Сохранить файл с новым именем

if (wExt = '.JPG') or (wExt = '.JPEG')

then Result := SaveJpegToFile (wFileName, RqImg)

else if (wExt = '.BMP')

then Result := SaveBmpToFile (wFileName, RqImg);

end;

end;

// ------------------------------------------------------------------------

// Диалоговое сохранение изображения из Image в файл

function TLoadsaveImage.DlgSaveImgToFile (RqImg : TImage) : boolean;

var wFileName : string; // Буфер имени файла

wExt : string; // Буфер расширения файла

begin

Result := False;

if fSaveDlg.Execute

then begin

wFileName := Trim(fSaveDlg.FileName);

wExt := UpperCase(ExtractFileExt(wFileName));

case fSaveDlg.FilterIndex of

FileFormatJPEG : begin

// Если пользователь не указал расширение, то это выполним мы

if not ((wExt = '.JPG') or (wExt = '.JPEG'))

then wFileName := wFileName + '.jpg';

Result := SaveImgToFile(wFileName, RqImg);

end;

FileFormatBMP : begin

// Если пользователь не указал расширение, то это выполним мы

if not (wExt = '.BMP')

then wFileName := wFileName + '.bmp';

Result := SaveImgToFile(wFileName, RqImg);

end;

end;

end;

end;

// ========================================================================

// PIVATE ПРОЦЕДУРЫ И МЕТОДЫ

// ========================================================================

// Проверка допустимого пиксельного формата

function TLoadsaveImage.VerifyPixFormat (RqBitMap : TBitMap) : boolean;

begin

Result := False;

case RqBitMap.PixelFormat of

pfDevice : Result := True;

pf4bit : Result := True;

pf8bit : Result := True;

pf15bit : Result := True;

pf16bit : Result := True;

pf24bit : Result := True;

pf32bit : Result := True;

end;

end;

// ------------------------------------------------------------------------

// Получить идентификатор пиксельного формата

function TLoadsaveImage.GetIdPixFormat (RqBitMap : TBitMap) : string;

begin

case RqBitMap.PixelFormat of

pfDevice : Result := 'pfDevice';

pf1bit : Result := 'pf1bit';

pf4bit : Result := 'pf4bit';

pf8bit : Result := 'pf8bit';

pf15bit : Result := 'pf15bit';

pf16bit : Result := 'pf16bit';

pf24bit : Result := 'pf24bit';

pf32bit : Result := 'pf32bit';

pfCustom : Result := 'pfCustom';

else Result := 'pfCustom';

end;

end;

// ========================================================================

// ФАЙЛОВАЯ ЗАГРУЗКА и СОХРАНЕНИЕ ИЗОБРАЖЕНИЯ В JPEG ФОРМАТЕ

// ========================================================================

// Вспоимогательный класс для обслуживания прогресса при загрузке JPEG

type

TJPEGService = class(TObject)

fProgress : TProgressBar;

// Обслуживание индикатора прогресса при JPEG преобразованиях

procedure Progress(Sender: TObject; Stage: TProgressStage;

PercentDone: Byte; RedrawNow: Boolean; const R : TRect;

const Msg: String);

end;

// ------------------------------------------------------------------------

// Обработчик событий индикатора прогресса при JPEG загрузке

procedure TJPEGService.Progress(Sender: TObject;

Stage: TProgressStage; PercentDone: Byte;

RedrawNow: Boolean; const R: TRect;

const Msg: String);

begin

case Stage of

psStarting: begin

fProgress.Position := 0;

fProgress.Max := 100;

end;

psEnding: begin

fProgress.Position := 0;

end;

psRunning: begin

fProgress.Position := PercentDone;

end;

end;

end;

// ------------------------------------------------------------------------

// Загрузка JPEG файла в Image.Pictures.BitMap

function TLoadsaveImage.LoadJpegPicFromFile

(RqFileName : string;

RqImage : TImage) : boolean;

var WPicture : TPicture; // Буферный ОБЪЕКТ

JS : TJPEGService; // Буферный ОБЪЕКТ

begin

Result := False;

if (RqFileName <> '') and Assigned(RqImage)

then begin

// Создать рабочие объекты

WPicture := TPicture.Create;

JS := TJPEGService.Create;

try

// Загрузить WPicture картинкой

WPicture.LoadFromFile(RqFileName);

if WPicture.Graphic is TJpegImage then

begin

if fProgressBar <> nil

then begin

// Подключить ссылку на обработчик прогресса в загрузки JPEG

JS.fProgress := fProgressBar;

// Подключим процедуру обслуживания прогресса

TJpegImage(WPicture.Graphic).OnProgress := JS.Progress;

end;

// Преобразуем JPG в BitMap

TJpegImage(WPicture.Graphic).DIBNeeded;

// Подключим BitMap в ImageMain

RqImage.Picture.Bitmap.Assign(TBitmap(WPicture.Graphic));

// Принудительно установим формат пикселя 24 бита на пиксель

RqImage.Picture.Bitmap.PixelFormat := pf24bit;

// Финальный отчет

if fMsgPanel <> nil

then fMsgPanel.Text := ' Загрузка изображения успешно завершена';

Result := True;

end;

finally

// Освободить рабочие объекты

WPicture.Free;

JS.Free;

end;

end;

end;

// ------------------------------------------------------------------------

// Преобразование Image.Pictures.BitMap в JPEG и сохранение в файл

function TLoadsaveImage.SaveJpegToFile

(RqFileName : string;

RqImage : TImage

) : boolean;

var JI : TJpegImage; // Буферный ОБЪЕКТ

JS : TJPEGService; // Буферный ОБЪЕКТ

begin

Result := False;

if (RqFileName <> '') and Assigned(RqImage.Picture.Bitmap)

then begin

// Создать рабочие объекты

JS := TJPEGService.Create;

JI := TJpegImage.Create;

try

if fProgressBar <> nil

then begin

JS.fProgress := fProgressBar;

JI.OnProgress := JS.Progress; // процедура обслуживания прогресса

end;

JI.PixelFormat := jf24bit; // колисество битов на пиксель

if fJpegQuality > 100

then JI.CompressionQuality := 100

else JI.CompressionQuality := fJpegQuality;

JI.ProgressiveEncoding := fJpegProgressive;

JI.Assign(RqImage.Picture.Bitmap);

JI.SaveToFile(RqFileName);

// Финальный отчет

if fMsgPanel <> nil

then fMsgPanel.Text := ' Сохранеие изображения успешно завершено';

Result := True;

finally

// Освободить рабочие объекты

JI.Free;

JS.Free;

end;

end;

end;

// ========================================================================

// ФАЙЛОВАЯ ЗАГРУЗКА и СОХРАНЕНИЕ ИЗОБРАЖЕНИЯ В BMP ФОРМАТЕ

// ========================================================================

// Загрузка BMP файла в Image.Pictures.BitMap

function TLoadsaveImage.LoadBmpPicFromFile

(RqFileName : string;

RqImage : TImage) : boolean;

var WBitMap : TBitMap; // Буферный ОБЪЕКТ

begin

Result := False;

if (RqFileName <> '') and Assigned(RqImage)

then begin

// Создать рабочие объекты

WBitMap := TBitMap.Create;;

try

// Загрузить WBitMap картинкой

WBitMap.LoadFromFile(RqFileName);

if VerifyPixFormat(WBitMap)

then begin

// Принудительно установить формат пикселя pf24bit

WBitMap.PixelFormat := pf24bit;

// Проверка успешной установки формата пикселя

if WBitMap.PixelFormat = pf24bit

then begin

// ----------------------------------------------------------

// Подключим BitMap в RqImage

RqImage.Picture.Bitmap.Assign(WBitMap);

// ----------------------------------------------------------

// Финальный отчет

if fMsgPanel <> nil

then fMsgPanel.Text := ' Загрузка изображения успешно завершена';

Result := True;

end

else

if fMsgPanel <> nil

then fMsgPanel.Text := ' ОШИБКА установки формата pf24bit!';

end

else

if fMsgPanel <> nil

then fMsgPanel.Text := ' ОШИБКА!'

+ ' Недопустимый формат ( '

+ GetIdPixFormat (WBitMap)

+ ' ) пикселя в файле';

finally

// Освободить рабочие объекты

WBitMap.Free;

end;

end;

end;

// ------------------------------------------------------------------------

// Сохранение изображения из Image.Picture.Bitmap в файл

function TLoadsaveImage.SaveBmpToFile

(RqFileName : string;

RqImage : TImage) : boolean;

begin

Result := False;

if (RqFileName <> '') and Assigned(RqImage.Picture.Bitmap)

then begin

try

RqImage.Picture.SaveToFile(RqFileName);

// Финальный отчет

if fMsgPanel <> nil

then fMsgPanel.Text := ' Сохранеие изображения успешно завершено';

Result := True;

except

if fMsgPanel <> nil

then fMsgPanel.Text := ' ОШИБКА при сохранении файла';

end;

end;

end;

// ========================================================================

// ОБМЕН ГРАФИЧЕСКОЙ ИНФОРМАЦИЕЙ С Clipboard

// ========================================================================

// Копировать содержимое BitMap в Clipboard

function CopyBitMapToClipboard(RqBitMap : TBitMap) : boolean;

begin

Result := False;

if not Assigned (RqBitMap) then Exit;

if (RqBitMap.Width > 0) and (RqBitMap.Height > 0)

then begin

try

Clipboard.Assign(RqBitMap);

Result := True;

except

MessageDlg('CopyBitMapToClipboard : '

+ 'ОШИБКА копирования в Clipboard!',

mtWarning, [mbOk], 0);

end;

end;

end;

// ------------------------------------------------------------------------

// Копировать содержимое Image в Clipboard

function TLoadsaveImage.CopyImageToClipboard(RqImg : TImage) : boolean;

begin

Result := False;

if Assigned(RqImg.Picture.Bitmap)

then begin

Result := CopyBitMapToClipboard (RqImg.Picture.Bitmap);

end;

end;

// ------------------------------------------------------------------------

// Вставка из Clipboard в Image

function TLoadsaveImage.PasteClipboardToImage(RqImage : TImage) : boolean;

var WBitmap : TBitmap; // Рабочий Bitmap

begin

Result := False;

// Если содержимое Clipboard является Windows BitMap

if Clipboard.HasFormat(CF\_BITMAP)

then begin

WBitmap := TBitmap.Create;

try

// Скопировать Clipboard в BitMap.

WBitmap.Assign(Clipboard);

if VerifyPixFormat(WBitMap)

then begin

// Принудительно установить формат пикселя pf24bit

WBitmap.PixelFormat := pf24bit;

// Проверка успешной установки формата пикселя

if WBitmap.PixelFormat = pf24bit

then begin

// Подключим BitMap в RqImage

RqImage.Picture.Bitmap.Assign(WBitmap);

if fMsgPanel <> nil

then fMsgPanel.Text := ' Выполнена вставка изображения из Clipboard';

Result := True;

end

else

if fMsgPanel <> nil

then fMsgPanel.Text := ' ОШИБКА установки формата pf24bit!';

end

else

if fMsgPanel <> nil

then fMsgPanel.Text := ' ОШИБКА!'

+ ' Недопустимый формат ( '

+ GetIdPixFormat (WBitMap)

+ ' ) пикселя в Clipboard';

finally

WBitmap.Free;

end;

end

else begin

if fMsgPanel <> nil

then fMsgPanel.Text := ' Вставка ОТМЕНЕНА!'

+ ' Формат изображения в Clipboard НЕ BitMap';

end;

end;

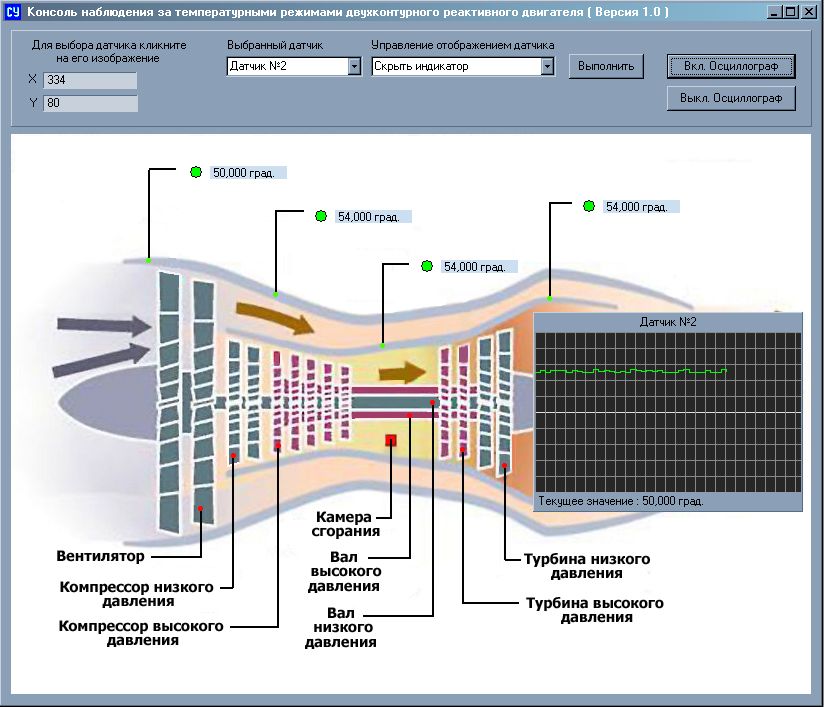
// ========================================================================

// END OF IMPLEMENTATION

// ========================================================================

end.

## UserApp02Main, образ экрана



## unit UserApp02Main;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, ExtCtrls, StdCtrls,

Common1, Service01, ImgLoadSave01, Scope01, Oscillograph01;

type

TUserApp02Form = class(TForm)

Image1: TImage;

Panel1: TPanel;

Label4: TLabel;

Label5: TLabel;

Label6: TLabel;

Button2: TButton;

Button3: TButton;

StaticText3: TStaticText;

StaticText4: TStaticText;

ComboBox1: TComboBox;

ComboBox2: TComboBox;

Button4: TButton;

Label1: TLabel;

Label2: TLabel;

procedure Button2Click(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure FormDestroy(Sender: TObject);

procedure Image1MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure Button4Click(Sender: TObject);

procedure Button3Click(Sender: TObject);

procedure ComboBox2Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

// Отобразить новое значение на датчиках

procedure ShowSensors();

end;

var

UserApp02Form: TUserApp02Form;

// =========================================================================

// =========================================================================

implementation

{$R \*.dfm}

// =========================================================================

// =========================================================================

// =========================================================================

// ОПИСАНИЕ ДАННЫХ

// =========================================================================

const ArrIndMax = 3; // Конечный индекс массивов датчиков

// Описатель параметров датчика

type TDescScope = array[0..ArrIndMax] of record

Xb, YB : integer;

Compact : boolean;

Title : string;

Min : extended;

Max : extended;

end;

// Массив статических параметров датчика

const DescScope : TDescScope =

(

(Xb:175; YB:28; Compact:True; Title:'Датчик №1'; Min:-110; Max:110),

(Xb:300; YB:72; Compact:True; Title:'Датчик №2'; Min:-110; Max:110),

(Xb:406; YB:122; Compact:True; Title:'Датчик №3'; Min:-110; Max:110),

(Xb:568; YB:62; Compact:True; Title:'Датчик №4'; Min:-110; Max:110)

);

// =========================================================================

// РАБОЧИЕ ДАННЫЕ ДАННЫХ

// =========================================================================

// Индекс выбранного датчика

var SelectInd : integer;

// Рабочий массив датчиков

var ScopeArr : array[0..ArrIndMax] of TScope;

// Объект - Осциллограф

var Osc01 : TOscillograph;

// Объект - загрузки / сохраниния картинки

var ImgLoadSave1 : TLoadSaveImage; // Обект загрузки и сохранения Image

// =========================================================================

// РАБОЧИЕ ПРОЦЕДУРЫ И ФУНКЦИИ

// =========================================================================

// -------------------------------------------------------------------------

// Выбор датчика

procedure TUserApp02Form.Image1MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

var Ind : integer;

begin

// Координаты MouseDown

StaticText3.Caption := ' ' + IntToStr(X);

StaticText4.Caption := ' ' + IntToStr(Y);

// Выбор сенсора

if Shift \* [ssAlt] = []

then begin

for Ind := Low(ScopeArr) to High(ScopeArr)

do begin

if not Assigned(ScopeArr[Ind]) then Continue;

if IsXYInRect(ScopeArr[Ind].SRect, X, Y)

then begin

SelectInd := Ind;

ComboBox2.ItemIndex := SelectInd;

if Assigned(Osc01)

then Osc01.Title := DescScope[SelectInd].Title;

end;

end;

end;

// Переместить осциллограф

if (Shift \* [ssAlt] <> []) and Assigned(Osc01)

then begin

Osc01.Free;

Osc01 := TOscillograph.Create(UserApp02Form,

Rect(X + Image1.Left,

Y + Image1.Top,

X + Image1.Left + 270,

Y + Image1.Top + 200));

Osc01.Title := DescScope[SelectInd].Title;

end;

end;

// -------------------------------------------------------------------------

// Выбор датчика

procedure TUserApp02Form.ComboBox2Click(Sender: TObject);

var Ind : integer;

begin

Ind := ComboBox2.ItemIndex;

if (Ind >= Low(ScopeArr)) and (Ind <= High(ScopeArr))

then begin

SelectInd := Ind;

if Assigned(Osc01)

then Osc01.Title := DescScope[SelectInd].Title;

end;

end;

// -------------------------------------------------------------------------

// Операции с индикаторами (Выполнить)

procedure TUserApp02Form.Button2Click(Sender: TObject);

var Ind : integer;

begin

Ind := ComboBox2.ItemIndex;

if Assigned(ScopeArr[Ind])

then begin

case ComboBox1.ItemIndex of

// Скрыть индикатор

0 : ScopeArr[Ind].Visible := False;

//Показать индикатор

1 : ScopeArr[Ind].Visible := True;

// Сделать компактным

2 : ScopeArr[Ind].Compact := True;

// Сделать полным

3 : ScopeArr[Ind].Compact := False;

// Сделать прозрачным

4 : ScopeArr[Ind].Transparent := True;

//Сделать непрозрачным

5 : ScopeArr[Ind].Transparent := False;

end;

end;

end;

// -------------------------------------------------------------------------

// Отобразить новое значение на датчиках

procedure TUserApp02Form.ShowSensors();

var Ind : integer;

begin

for Ind := Low(ScopeArr) to High(ScopeArr)

do begin

if Assigned(ScopeArr[Ind])

then begin

ScopeArr[Ind].Value := REC2.Arr[Ind];

ScopeArr[Ind].DrawScope();

end;

end;

if Assigned(Osc01)

then begin

Osc01.AddNewData(REC2.Arr[SelectInd]);

Osc01.ShowBeam;

end;

end;

// -------------------------------------------------------------------------

// Включить осциллограф

procedure TUserApp02Form.Button3Click(Sender: TObject);

const XB = 530;

YB = 290;

begin

if not Assigned(Osc01)

then begin

Osc01 := TOscillograph.Create(UserApp02Form,

Rect(XB,YB,XB + 270,YB + 200));

Osc01.Title := ScopeArr[SelectInd].Title;

Osc01.YMes := ScopeArr[SelectInd].Measure;

end;

end;

// -------------------------------------------------------------------------

// Выключить осциллограф

procedure TUserApp02Form.Button4Click(Sender: TObject);

begin

if Assigned(Osc01)

then begin

Osc01.Free;

Osc01 := nil;

end;

end;

// =========================================================================

// ИНИЦИИРОВАНИЕ / ТЕРМИНИРОВАНИЕ ПРИЛОЖЕНИЯ

// =========================================================================

// -------------------------------------------------------------------------

procedure TUserApp02Form.FormCreate(Sender: TObject);

var Ind : integer;

begin

// -------------------------------

// Загрузить изображение объекта

ImgLoadSave1 := TLoadSaveImage.Create(nil,nil);

ImgLoadSave1.LoadImgFromFile(ApplicationDirectory

+ '\PICTUREs\engine01.jpg',

Image1);

// -------------------------------

ComboBox2.Clear;

for Ind := Low(ScopeArr) to High(ScopeArr)

do begin

ComboBox2.Items.Add(DescScope[Ind].Title);

ScopeArr[Ind] := TScope.Create(Image1);

with ScopeArr[Ind]

do begin

XBeg := DescScope[Ind].Xb;

YBeg := DescScope[Ind].Yb;

Compact := DescScope[Ind].Compact;

Measure := 'град.';

Min := DescScope[Ind].Min;

Max := DescScope[Ind].Max;

YZone := 80;

RZone := 90;

Title := DescScope[Ind].Title;

Visible := True;

end;

SelectInd := 0;

ComboBox2.ItemIndex := SelectInd;

end;

ComboBox1.ItemIndex := 0;

end;

// -------------------------------------------------------------------------

procedure TUserApp02Form.FormDestroy(Sender: TObject);

var Ind : integer;

begin

for Ind := Low(ScopeArr) to High(ScopeArr)

do begin

if Assigned(ScopeArr[Ind])

then ScopeArr[Ind].Free;

end;

if Assigned(Osc01)

then begin

Osc01.Free;

Osc01 := nil;

end;

if Assigned(ImgLoadSave1) then ImgLoadSave1.Free;

end;

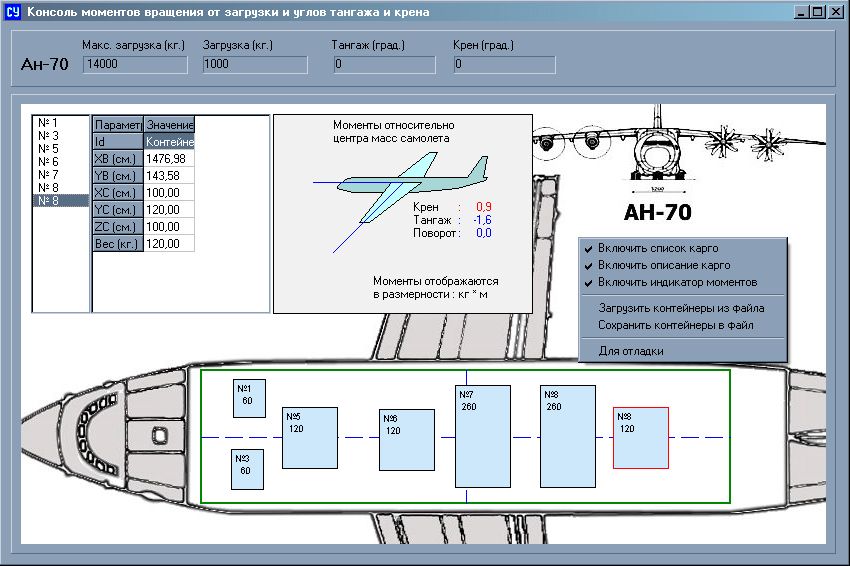
// =========================================================================

// КОНЕЦ

// =========================================================================

end.

## UserApp03Main, образ экрана



## unit UserApp03Main;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, ExtCtrls, StdCtrls, Grids, ComCtrls,

UserApp03Data, Service01, Cargo01, Loading01, AirCraft01;

type

TUserApp03Form = class(TForm)

Panel1: TPanel;

Panel2: TPanel;

Label3: TLabel;

sttxtLZW: TStaticText;

Label4: TLabel;

Label5: TLabel;

StaticText1: TStaticText;

Label10: TLabel;

Label11: TLabel;

stxtPitch: TStaticText;

stxtRoll: TStaticText;

procedure FormCreate(Sender: TObject);

procedure FormDestroy(Sender: TObject);

//procedure CbBoxCargoBaseClick(Sender: TObject);

//procedure edCargoWeightChange(Sender: TObject);

//procedure TrackBarPitchChange(Sender: TObject);

//procedure TrackBarRollChange(Sender: TObject);

// procedure edCargoZCChange(Sender: TObject);

private

procedure ShowCargoWeight (Sended : TObject);

procedure ShowPitchAndRoll(Sended : TObject);

public

end;

var

UserApp03Form: TUserApp03Form;

// -------------------------------------------------------------------------

var Loading : TLoading;

// ========================================================================

// ========================================================================

implementation

{$R \*.dfm}

// ========================================================================

// ========================================================================

// ========================================================================

// ПРИКЛАДНЫЕ МЕТОДЫ

// ========================================================================

// -------------------------------------------------------------------------

// Получить директорию старта приложения

function GetApplicationDirectory() : string;

begin

Result := Application.ExeName;

Result := ExtractFileDir(Result);

end;

// -------------------------------------------------------------------------

procedure TUserApp03Form.FormCreate(Sender: TObject);

begin

// Описания самолета по умолчанию

AirCraftRec := AirCraftRecAn70;

// Объект управления загрузкой

if not Assigned(Loading)

then begin

Loading := TLoading.Create(Panel1, 10,10,

Panel1.Width - 20,

Panel1.Height - 20 );

Loading.onWLChange := ShowCargoWeight;

Loading.onAngleChange := ShowPitchAndRoll;

end;

end;

// -------------------------------------------------------------------------

procedure TUserApp03Form.FormDestroy(Sender: TObject);

begin

if Assigned(Loading)

then begin

Loading.Free;

Loading := nil;

end;

end;

// -------------------------------------------------------------------------

// Обработчик события - изменился общий вес загрузки

procedure TUserApp03Form.ShowCargoWeight(Sended : TObject);

begin

if Assigned(Loading)

then begin

sttxtLZW.Caption := FloatToStr(Loading.WSumCargo);

end;

end;

// -------------------------------------------------------------------------

// Обработчик события - изменился тангаж или крен

procedure TUserApp03Form.ShowPitchAndRoll(Sended : TObject);

const ToDeg = 180 / Pi;

begin

if Assigned(Loading)

then begin

stxtPitch.Caption := FloatToStr(Loading.PitchAngle \* ToDeg);

stxtRoll.Caption := FloatToStr(Loading.RollAngle \* ToDeg);

end;

end;

// ========================================================================

// КОНЕЦ

// ========================================================================

end.

## unit UserApp03Data;

interface

uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,

Dialogs, ExtCtrls,

Service01;

// ========================================================================

// ------------------------------------------------------------------------

// Описание выборки одной записи самолета из базы данных

type TAirCraftRec = record

Id : string; // Название самолета

FName : string; // Короткое имя файла картинки зоны загрузки

// ---------------------

// Габариты зоны загрузки

LenX : extended; // Длина зоны загрузки (см.)

LenY : extended; // Ширина зоны загрузки (см.)

// ---------------------

// Сдвиг центра масс самолета относительно

// начала координат зоны загрузки

OfsX0 : extended; // Сдвиг для X0 (см.)

OfsY0 : extended; // Сдвиг для Y0 (см.)

OfsZ0 : extended; // Сдвиг для Z0 (см.)

// ---------------------

RectLZ : TRect; // Rect зоны загрузки на картинке

// ---------------------

Weight : extended; // Максимальный вес загрузки (кг.)

end;

// ------------------------------------------------------------------------

// Имитация выборки одной записи о конкретном самолете

const AirCraftRecAn70 : TAirCraftRec =

(

Id : 'Aн-70'; // Название самолета

FName : '.\PICTUREs\an70v2.jpg'; // Имя файла картинки зоны загрузки

// ---------------------

// Габариты зоны загрузки

LenX : 1900.0; // Длина зоны загрузки (см.)

LenY : 0520.0; // Ширина зоны загрузки (см.)

// ---------------------

// Сдвиг центра масс самолета относительно

// начала координат зоны загрузки

OfsX0 : 1900.0 / 2; // Сдвиг для X0 (см.)

OfsY0 : - 520.0 / 2; // Сдвиг для Y0 (см.)

OfsZ0 : - 400.0; // Сдвиг для Z0 (см.)

// ---------------------

// Rect зоны загрузки на картинке

RectLZ : ( Left : 180;

Top : 266;

Right : 710;

Bottom : 400;

);

// ---------------------

Weight : 14000; // Максимальный вес загрузки (кг.)

);

// ========================================================================

// ------------------------------------------------------------------------

// Описание выборки одной записи контейнера из базы данных

type TCargoRec = record

Id : string; // Идентификатор контейнера

LenX : extended; // Длина контейнера (см.)

LenY : extended; // Ширина контейнера (см.)

LenZ : extended; // Высота контейнера (см.)

Weight : extended; // Собственный вес контейнера (кг.)

end;

// ------------------------------------------------------------------------

// Имитация базы данных контейнеров

const CargoBase : array[0..3] of TCargoRec =

(

( Id : 'Контейнер 120 \* 160';

LenX : 120; LenY : 160; LenZ : 100; Weight : 60 ),

( Id : 'Контейнер 200 \* 240';

LenX : 200; LenY : 240; LenZ : 160; Weight : 120),

( Id : 'Контейнер 200 \* 400';

LenX : 200; LenY : 400; LenZ : 160; Weight : 260),

( Id : 'Контейнер 400 \* 200';

LenX : 400; LenY : 200; LenZ : 160; Weight : 260)

);

// ========================================================================

// ------------------------------------------------------------------------

// Директория запуска программы

var ApplicationDirectory : string;

// ------------------------------------------------------------------------

// Текущая запись данных о выбранном самолете

var AirCraftRec : TAirCraftRec;

// ------------------------------------------------------------------------

// Текущие данные о выбранном контейнере

var CargoRec : TCargoRec; // Запись о выбранном контейнере

CargoWeight : extended; // Полезный груз

CargoZC : extended; // Высота центра тяжести контейнера (см.)

// ========================================================================

// ========================================================================

implementation

// ========================================================================

// ========================================================================

// ========================================================================

// КОНЕЦ

// ========================================================================

end.

## unit Cargo01;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,

Dialogs, ExtCtrls,

Service01, UserApp03Data;

// -------------------------------------------------------------------------

// -------------------------------------------------------------------------

type TCargo = class(TObject)

private

fImg : TImage; // Арендуемый Image

// ---------------------

fBmp : TBitMap; // Область на Image под объектом пиктограммы

fVisible : boolean; // Видимость пиктограммы объекта

// ---------------------

// Сохраняемые в файле загрузки самолета поля

// ---------------------

// Отображаемые поля

fIndx : integer; // Уникальный индекс контейнера

fId : string; // Идентификатор контейнера

fXB : extended; // X -смещение (см.) относительно зоны загрузки

fYB : extended; // Y -смещение (см.) относительно зоны загрузки

fXC : extended; // X -смещение (см.) центра тяжести контейнера

fYC : extended; // Y -смещение (см.) центра тяжести контейнера

fZC : extended; // Z -смещение (см.) центра тяжести контейнера

fWFull : extended; // Вес (кг.) заполненного контейнера

// ---------------------

// Неотображаемые поля

fCutout : TRect; // Несущий Rect пиктограммы объекта

// ---------------------

// Несохраняемые в файле загрузки самолета поля

// ---------------------

// Вектор центра масс контейнера относительно центра масс самолета

fVector : TVector3D;

// ---------------------

fMRoll : extended; // Момент контейнера по крену

fMPitch : extended; // Момент контейнера по тангажу

fMTurn : extended; // Момент контейнера по развороту

// ---------------------

procedure ShowCargo();

// Обслуживание property

procedure SetVisible(RqVisible : boolean);

public

constructor Create(RqImage : TImage);

procedure Free();

// ------------------

// Управление отображением объекта

property Visible : boolean read fVisible write SetVisible;

// ------------------

// Сохраняемые в файле загрузки поля

// -----------------

property Indx : integer read fIndx write fIndx;

property Id : string read fId write fId;

property XB : extended read fXB write fXB;

property YB : extended read fYB write fYB;

property XC : extended read fXC write fXC;

property YC : extended read fYC write fYC;

property ZC : extended read fZC write fZC;

property WFull : extended read fWFull write fWFull;

// ------------------

// Несохраняемые в файле загрузки поля

// -----------------

// Неотображаемые поля

property Cutout : TRect read fCutout write fCutout;

// Вектор контейнера относительно центра масс самолета

property Vector : TVector3D read fVector write fVector;

// Текущие моменты, создаваемые контейнером

property MRoll : extended read fMRoll;

property MPitch : extended read fMPitch;

property MTurn : extended read fMTurn;

end;

// -------------------------------------------------------------------------

// -------------------------------------------------------------------------

implementation

// -------------------------------------------------------------------------

// -------------------------------------------------------------------------

// -------------------------------------------------------------------------

constructor TCargo.Create(RqImage : TImage);

begin

inherited Create;

fImg := RqImage;

fBmp := TBitMap.Create;

fBmp.PixelFormat := pf24bit;

FillChar(fCutout, SizeOf(TRect), #0);

fVisible := False;

end;

// -------------------------------------------------------------------------

procedure TCargo.Free();

begin

if Assigned(fBmp) then fBmp.Free;

inherited Free();

end;

// -------------------------------------------------------------------------

procedure TCargo.ShowCargo();

var wFontName : string;

wFontSize : integer;

begin

with fImg.Canvas do

begin

Pen.Width := 1;

Brush.Color := RGB(200,230,250);

Brush.Style := bsSolid;

Rectangle(fCutout);

wFontName := Font.Name;

wFontSize := Font.Size;

Font.Color := clBlack;

Font.Name := 'Arial';

Font.Size := 7;

TextOut(fCutout.Left + 4, fCutout.Top + 4,

'№' + IntToStr(fIndx));

TextOut(fCutout.Left + 4, fCutout.Top + 4 + TextHeight('1'),

Format('%4.0f', [fWFull]));

Font.Name := wFontName;

Font.Size := wFontSize;

end;

end;

// -------------------------------------------------------------------------

// Установка видимости объекта

procedure TCargo.SetVisible(RqVisible : boolean);

begin

// Входной контроль

// -----------------------------

if RqVisible

then begin

if (not fVisible)

then begin

// Сохранить образ площадки под объектом

SaveBitMap(fImg.Picture.Bitmap, fBmp, fCutout);

// Отобразить объект

ShowCargo();

fVisible := True;

end;

end

else begin

if fVisible

then begin

// Спрятать объект

RestoreBitMap(fBmp, // BitMap источник

fImg.Picture.Bitmap, // BitMap приемник

fCutout.Left, fCutout.Top);

fImg.Refresh;

fVisible := False;

end;

end;

end;

// ========================================================================

// КОНЕЦ

// ========================================================================

end.

## unit Loading01;

interface

uses

Windows, Messages, SysUtils, StrUtils, Variants, Classes, Graphics,

Controls, Dialogs, Menus, Grids, ExtCtrls, StdCtrls,

UserApp03Data, Service01, ImgLoadSave01, Cargo01,

AirCraft01, md5, MomentScope01;

// -------------------------------------------------------------------------

type TonWLChange = procedure(Value : extended) of object;

// -------------------------------------------------------------------------

type TLoading = class(TObject)

private

// ----------------------------

// Динамические компоненты

// ----------------------------

fPanel : TPanel; // Динамическая несущая панель

// ----------------------------

fImg : TImage; // Рабочий Image

fPMenu : TPopupMenu; // PopupMenu на fImg

fSelectorBox : TListBox; // Селектор объектов TCargo

fGrid : TStringGrid; // Редактор объектов TCargo

// ----------------------------

// Встроенные объекты

// ----------------------------

fLoadSaveImage : TLoadSaveImage; // Объект - Загрузчик картинок

fAirCraft : TAirCraft; // Объект - Самолета

fMomentScope : TMomentScope; // Объект - Индикатор моментов

// ----------------------------

// Рабочие объекты и переменные

// ----------------------------

fLoadingList : TList; // Список ссылок на объекты TCargo

// ----------------------------

fSavePenMode : TPenMode; // Буфер хранения PenMode

fContourFlag : Boolean; // Режим перерисовки контура

fContourShow : Boolean; // Контур на экране

// ---------------------------

fUniqueCT : cardinal; // Уникальный счетчик объектов

fCargoInd : Integer; // Индекс Текущего объекта в fList

// ---------------------------

fWSumCargo : extended; // Суммарный вес текущей загрузки

fonWLChange : TNotifyEvent; // Изменился суммарный вес загрузки

// ---------------------------

fPitchAngle : extended; // Угол тангажа в радианах

fRollAngle : extended; // Угол крена в радианах

fonAngleChange : TNotifyEvent; // Изменился угол тангажа или крена

// ---------------------------

fFileName : string; // Полное имя файла загрузки

fMd5Str : string;

// ---------------------------

// Создать и подключить PopupMenu

procedure CreatePopUpMenu();

// Обработчик событий для PopUpMenu

procedure ImgMenuClick(Sender: TObject);

// ---------------------------

// Обработчик, выполняющий старт прорисовки скользящей фигуры

procedure MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

// Вернуть контур в область зоны загрузки

procedure ReturnRectToLZ(var RqRect : TRect);

// Обработчик, выполняющий перемещение контура скользящей фигуры

procedure MouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);

// Обработчик, завершающий прорисовку скользящей фигуры

procedure MouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

// ---------------------------

// Обработчик событий Click на fSelectorBox

procedure ListBoxClick(Sender: TObject);

// ---------------------------

// Установить и отметить текущий Cargo по указанному индексу

function SetCurrentCargo (RqInd : integer) : boolean;

// ---------------------------

// Вычислить и отобразить моменты

procedure CalcAndShowMoments();

// Вычислить вектор центра масс контейнера относительно центра масс самолета

procedure CargoVector (XB, YB : extended; var RqCargo : TCargo);

// ---------------------------

// Создать и инициализировать один объект Cargo

function CreateCargo(RqX, RqY : integer) : TCargo;

// Вычислить суммарный вес загрузки

procedure CalcWSumCargo();

// Добавить один объект в список Cargo

function AddCargoByXY(RqX, RQY : integer; RqTitle : string) : TCargo;

// Удалить один объект из списока Cargo

procedure DeleteCargo(RqInd : integer);

// Очистить все объекты в зоне загрузки

procedure ClearLoading();

// ---------------------------

// Отобразить текущий контейнер в fGrid

procedure CargoToGrid();

// ---------------------------

// Сохранить вычисленный fMd5Str в файл

procedure Md5SaveToFile(RqFileName : string);

// Прочитать md5 - строку из файла \*.md5

function ReadMd5FromFile(RqFileName : string) : string;

// ---------------------------

// Обслуживание property

procedure SetCargoInd(Ind : integer);

function GetCargo(Ind : integer) : TCargo;

function GetCargoCount() : integer;

// Вычислить момент по тангажу

procedure SetPitchAngle(RqAngle : extended);

// Вычислить момент по крену

procedure SetRollAngle(RqAngle : extended);

public

// ---------------------------

constructor Create(RqWinControl : TWinControl;

XB, YB, RqWidth, RqHeight : integer);

procedure Free();

// Загрузить картинку зоны загрузки

procedure LoadPicture(RqFileName : string);

// ---------------------------

// Исполнитель сохранения файла

procedure SaveToFile(RqFileName : string; RqDelimitor : char);

// Выполнить в диалоге сохранения данных в файл

function DialogSaveToFile() : string;

// ---------------------------

// Исполнитель загрузки файла

procedure LoadFromFile(RqFileName : string; RqDelimitor : char);

// Выполнить в диалоге загрузку данных из файла

// с разделителями полей табуляцией (char($09))

function DialogLoadFromFile() : string;

// ---------------------------

// Основной Image

property MainImg : TImage read fImg;

// ---------------------------

// Полное имя файла загрузки

property FileName : string read fFileName;

// ---------------------------

// Список загрузки

property LoadingList : TList read fLoadingList;

// Индекс отмеченного контейнера в списке загрузки

property CargoInd : Integer read fCargoInd write SetCargoInd;

// Количество контейнеров в списке загрузки

property CargoCount : Integer read GetCargoCount;

// Объект контейнера по указанному индексу в списке загрузки

property Cargo[Ind : integer] : TCargo read GetCargo;

// ----------------------------

// Угол тангажа в градусах

property PitchAngle : extended read fPitchAngle write SetPitchAngle;

// Угол крена в градусах

property RollAngle : extended read fRollAngle write SetRollAngle;

property onAngleChange : TNotifyEvent read fonAngleChange

write fonAngleChange;

// ----------------------------

// Суммарный вес контейнеров в списке загрузки

property WSumCargo : extended read fWSumCargo;

// Событие - Изменился суммарный вес загрузки

property onWLChange : TNotifyEvent read fonWLChange write fonWLChange;

end;

// ----------------------------------------------------------------

// Префикс комментария в файле загрузки

const CommentID = '//';

// ================================================================

// ================================================================

implementation

// ================================================================

// ================================================================

// ================================================================

// ОБЩИЙ ПРОЦЕДУРНЫЙ СЕРВИС

// ================================================================

// ----------------------------------------------------------------

// 25.01.2017

// Программно создать компонент, являющийся потомком TControl.

// Функция CreateControl запрашивает ссылку на class (тип) того

// компонента, которого необходимо создать в контейнере RqOwner

// (Например, RqOwner : Form1 или Panel1).

function CreateControl(ControlClass : TControlClass;

RqOwner : TWinControl;

const ComponentName : string;

X, Y, W, H : Integer): TControl;

begin

Result := ControlClass.Create(RqOwner);

with Result do

begin

Parent := RqOwner; // Ссылка на объект контейнер

Name := ComponentName; // Переопределить имя компонента

SetBounds(X, Y, W, H); // Наследуется из TWinControl

Visible := True; // Требование отображать компонент

end;

end;

// ----------------------------------------------------------------

// 25.01.2017

constructor TLoading.Create(RqWinControl : TWinControl;

XB, YB, RqWidth, RqHeight : integer);

begin

inherited Create;

// ------------------------------------------

// Создать несущую панель

fPanel := TPanel(CreateControl(TPanel, RqWinControl, '',

XB, YB, RqWidth, RqHeight));

// ------------------------------------------

// Создать ListBox - селектор контейнеров

fSelectorBox := TListBox(CreateControl(TListBox, fPanel, '',

10, 10, 60, 200));

fSelectorBox.Clear;

fSelectorBox.OnClick := ListBoxClick;

// ------------------------------------------

// Создать StringGrid - табло параметров контейнера

fGrid := TStringGrid(CreateControl(TStringGrid, fPanel, '',

70, 10, 180, 200));

fGrid.ColCount := 2;

fGrid.RowCount := 2;

fGrid.DefaultRowHeight := 16;

fGrid.DefaultColWidth := 50;

fGrid.ScrollBars := ssBoth;

fGrid.Options := fGrid.Options + [goEditing,goColSizing];

fGrid.Cells[0,0] := 'Параметр';

fGrid.Cells[1,0] := 'Значение';

// ------------------------------------------

// Создать объект - Загрузчик картинок

fLoadSaveImage := TLoadSaveImage.Create(nil, nil);

// ------------------------------------------

// Создать Image - Рабочий Image

fImg := TImage(CreateControl(TImage, fPanel, '',

0, 0, fPanel.Width, fPanel.Height));

fImg.Picture.Bitmap.PixelFormat := pf24bit;

fImg.Picture.Bitmap.Width := fImg.Width;

fImg.Picture.Bitmap.Height := fImg.Height;

// ------------------------------------------

// Установить обработчики событий

fImg.OnMouseDown := MouseDown;

fImg.OnMouseMove := MouseMove;

fImg.OnMouseMove := nil;

fImg.OnMouseUp := MouseUp;

// ------------------------------------------

// Создать и подключить PopupMenu

CreatePopUpMenu();

// ------------------------------------------

// Создать объект - Самолет

fAirCraft := TAirCraft.Create(AirCraftRec);

LoadPicture(ApplicationDirectory + fAirCraft.FName);

fAirCraft.ShowLoadingZone(fImg);

// ------------------------------------------

// Создать объект - Индикаторов моментов

fMomentScope := TMomentScope.Create(fImg, 252, 10);

// ------------------------------------------

// Создать объект - Пустая загрузка

fLoadingList := TList.Create;

fCargoInd := -1;

fUniqueCT := 1;

// ------------------------------------------

fPitchAngle := 0; // Угол тангажа в радианах

fRollAngle := 0; // Угол крена в радианах

// ------------------------------------------

end;

// ----------------------------------------------------------------

// 25.01.2017

procedure TLoading.Free();

var Ind : integer;

wCargo : TCargo;

begin

fAirCraft.Free;

fMomentScope.Free;

// Удалть все Cargo объекты

if Assigned(fLoadingList)

then begin

if fLoadingList.Count > 0

then begin

for Ind := 0 to fLoadingList.Count - 1

do if Assigned(fLoadingList.Items[Ind])

then begin

wCargo := TCargo(fLoadingList.Items[Ind]);

wCargo.Free;

end;

end;

fLoadingList.Free;

end;

fPMenu.Free;

fImg.Free;

fLoadSaveImage.Free;

fGrid.Free;

fSelectorBox.Free;

fPanel.Free;

inherited Free();

end;

// ----------------------------------------------------------------

// 21.01.2017

// Создать и подключить PopupMenu

procedure TLoading.CreatePopUpMenu();

var MenuItems : array of TMenuItem;

begin

// -----------------

// Создать PopupMenu

SetLength(MenuItems, 8);

MenuItems[0]:= NewItem('Включить список карго',

TextToShortCut(''),

True, True, ImgMenuClick, 0, 'ITEM1');

MenuItems[0].Tag := 1;

//-------------------

MenuItems[1]:= NewItem('Включить описание карго',

TextToShortCut(''),

True, True, ImgMenuClick, 0, 'ITEM2');

MenuItems[1].Tag := 2;

//-------------------

MenuItems[2]:= NewItem('Включить индикатор моментов',

TextToShortCut(''),

True, True, ImgMenuClick, 0, 'ITEM3');

MenuItems[2].Tag := 3;

//-------------------

MenuItems[3]:= NewItem('-', TextToShortCut(''),

False, True, nil, 0, '');

//-------------------

MenuItems[4]:= NewItem('Загрузить контейнеры из файла',

TextToShortCut(''),

False, True, ImgMenuClick, 0, 'ITEM4');

MenuItems[4].Tag := 4;

//-------------------

MenuItems[5]:= NewItem('Сохранить контейнеры в файл',

TextToShortCut(''),

False, True, ImgMenuClick, 0, 'ITEM5');

MenuItems[5].Tag := 5;

//-------------------

MenuItems[6]:= NewItem('-', TextToShortCut(''),

False, True, nil, 0, '');

//-------------------

MenuItems[7]:= NewItem('Для отладки',

TextToShortCut(''),

False, True, ImgMenuClick, 0, 'ITEM6');

MenuItems[7].Tag := 6;

//-------------------

// Подключить PopupMenu

fPMenu := NewPopupMenu(fImg, 'Menu', paLeft, True, MenuItems);

fImg.PopupMenu := fPMenu;

// -----------------

SetLength(MenuItems,0);

end;

// ----------------------------------------------------------------

// 21.01.2017

// Обработчик событий в PopUpMenu на fGrid

procedure TLoading.ImgMenuClick(Sender: TObject);

begin

with TMenuItem(Sender)

do begin

case Tag of

1 : begin // Включить/выключить список карго

Checked := not Checked;

if Checked

then fSelectorBox.Visible := True

else fSelectorBox.Visible := False;

end;

2 : begin // Включить/выключить описание карго

Checked := not Checked;

if Checked

then fGrid.Visible := True

else fGrid.Visible := False;

end;

3 : begin // Включить/выключить индикатор моментов

Checked := not Checked;

if Checked

then fMomentScope.Visible := True

else fMomentScope.Visible := False;

end;

4 : begin // Загрузить контейнеры из файла

if fLoadingList.Count > 0

then begin

if MessageDlg('Текущая загрузка борта'+ #13#10

+ 'будет очищена. Продолжить?',

mtInformation,[mbYes,mbNo],0) = mrYes

then DialogLoadFromFile();

end

else DialogLoadFromFile();

end;

5 : begin // Сохранить контейнеры в файл

DialogSaveToFile();

end;

6 : begin // Для отладки

//

end;

else begin end;

end;

end;

end;

// ================================================================

// ОБЩИЙ ПРОЦЕДУРНЫЙ СЕРВИС

// ================================================================

// ----------------------------------------------------------------

// 25.01.2017

// Загрузить картинку зоны загрузки

procedure TLoading.LoadPicture(RqFileName : string);

begin

if FileExists(RqFileName)

then begin

if fLoadSaveImage.LoadImgFromFile(RqFileName, fImg)

then begin

if fImg.Picture.Bitmap.PixelFormat <> pf24bit

then fImg.Picture.Bitmap.PixelFormat := pf24bit;

fImg.Picture.Bitmap.Width := fImg.Width;

fImg.Picture.Bitmap.Height := fImg.Height;

end;

end;

end;

// ----------------------------------------------------------------

// 30.01.2017

// Вычислить суммарный вес загрузки

procedure TLoading.CalcWSumCargo();

var Ind : integer;

begin

fWSumCargo := 0;

if fLoadingList.Count < 0

then Exit;

for Ind := 0 to fLoadingList.Count - 1

do fWSumCargo := fWSumCargo + TCargo(fLoadingList[Ind]).WFull;

// Вызов внешнего обработчика событий

if Assigned(fonWLChange) then fonWLChange(Self);

end;

// ================================================================

// РСЧЕТ ВЕКТОРОВ И МОМЕНТОВ

// ================================================================

// ----------------------------------------------------------------

// Установить угол по тангажу

procedure TLoading.SetPitchAngle(RqAngle : extended);

const ToPi = Pi/180;

begin

fPitchAngle := RqAngle \* ToPi;

CalcAndShowMoments();

if Assigned(fonAngleChange)

then fonAngleChange(Self);

end;

// ----------------------------------------------------------------

// Установить угол по крену

procedure TLoading.SetRollAngle(RqAngle : extended);

const ToPi = Pi/180;

begin

fRollAngle := RqAngle \* ToPi;

CalcAndShowMoments();

if Assigned(fonAngleChange)

then fonAngleChange(Self);

end;

// ----------------------------------------------------------------

// Вычислить момент контейнера по тангажу

function CalcPitchMoment(RqVector : TVector3D;

RqAngle : extended;

RqWFull : extended) : extended;

var wV : TVector2DP;

begin

wV.X := RqVector.X;

wV.Y := RqVector.Z;

DecartToPolar(wV);

wV.Ang := wV.Ang + RqAngle;

PolarToDecart(wV);

Result := wV.X \* RqWFull;

end;

// ----------------------------------------------------------------

// Вычислить момент контейнера по крену

function CalcRollMoment(RqVector : TVector3D;

RqAngle : extended;

RqWFull : extended) : extended;

const ToRad = 180 / Pi;

var wV : TVector2DP;

begin

wV.X := RqVector.Y;

wV.Y := RqVector.Z;

DecartToPolar(wV);

wV.Ang := wV.Ang + RqAngle;

PolarToDecart(wV);

Result := wV.X \* RqWFull;

end;

// ----------------------------------------------------------------

// Вычислить и отобразить моменты

procedure TLoading.CalcAndShowMoments();

var wCargo : TCargo;

Ind : integer;

wMRoll : extended; // Момент по крену

wMPitch : extended; // Момент по тангажу

begin

wMRoll := 0;

wMPitch := 0;

if fLoadingList.Count > 0

then begin

for Ind := 0 to fLoadingList.Count - 1

do begin

wCargo := TCargo(fLoadingList.Items[Ind]);

with wCargo do

begin

wMPitch := wMPitch

+ CalcPitchMoment(Vector, fPitchAngle, WFull) / 100;

wMRoll := wMRoll +

+ CalcRollMoment(Vector, fRollAngle, WFull) / 100;

end;

end;

end;

fMomentScope.MPitch := wMPitch;

fMomentScope.MRoll := wMRoll;

end;

// fMRoll : extended; // Момент по крену

// fMPitch : extended; // Момент по тангажу

// fMTurn : extended; // Момент по развороту

// ----------------------------------------------------------------

// Вычислить вектор центра масс контейнера относительно центра масс самолета

procedure TLoading.CargoVector (XB, YB : extended;

var RqCargo : TCargo);

var V0, V1, V2, V4 : TVector2D;

begin

// Расчет проекции карго - вектора на плоскость зоны загрузки

V0.X := AirCraftRec.OfsX0;

V0.Y := AirCraftRec.OfsY0;

V1.X := XB;

V1.Y := - YB;

V2.X := RqCargo.XC;

V2.Y := - RqCargo.YC;

// Окончательный расчет карго - вектора относительно центра масс самолета

// V4 = - ( V0 - ( V1 + V2 )) = ( V1 + V2 ) - V0

V4 := SubVector2D(SumVector2D(V1, V2), V0);

with RqCargo.Vector

do begin

X := V4.X;

Y := V4.Y;

Z := AirCraftRec.OfsZ0 + RqCargo.ZC;

(\* // Для отладки

ShowMessage( ' X = ' + Format('%6.2f', [X])

+ '; Y = ' + Format('%6.2f', [Y])

+ '; Z = ' + Format('%6.2f', [Z]));

\*)

end;

// Вычислить и отобразить моменты

CalcAndShowMoments();

end;

// ----------------------------------------------------------------

// 30.01.2017

// Создать и инициализировать один объект Cargo

function TLoading.CreateCargo(RqX, RqY : integer) : TCargo;

var wCargo : TCargo;

wWLd : extended;

wRect : TRect;

begin

// ------------------------------------

// Входной контроль

Result := nil;

if not Assigned(fAirCraft) then Exit;

// ------------------------------------

// Выбор карго - контейнера

// ------------------------------------

// Вычислить Rect контейнера на Image

wRect := fAirCraft.XYLenToRect(RqX, RqY, CargoRec.LenX, CargoRec.LenY);

// Контроль попадания контейнера в зону загрузки

if not IsRectInMainRect(fAirCraft.RectLZ, wRect)

then begin

MessageDlg('ВНЕ ЗОНЫ ЗАГРУЗКИ',

mtWarning, [mbOk], 0);

Exit;

end;

// ------------------------------------

// Полный вес контейнера и полезного груза

wWLd := CargoRec.Weight + CargoWeight; // UserApp01Data

if fWSumCargo + wWLd > fAirCraft.Weight

then begin

MessageDlg('ДОБАВЛЕНИЕ КОНТЕЙНЕРА ОТМЕНЕНО, ТАК КАК'

+ #13#10 + 'ПРЕВЫШАЕТСЯ МАКС.ВЕС ЗАГРУЗКИ БОРТА ...',

mtError, [mbOk], 0);

Exit;

end;

// ------------------------------------

// Создание объекта для загрузки

wCargo := TCargo.Create(fImg);

with wCargo

do begin

Cutout := wRect;

// ------------------------------------

// Инициализировать начальные параметры загрузки

Indx := fUniqueCT;

Id := CargoRec.Id;

XB := fAirCraft.XOfsetRect(wRect);

YB := fAirCraft.YOfsetRect(wRect);

XC := CargoRec.LenX / 2;

YC := CargoRec.LenY / 2;

ZC := CargoZC; // UserApp01Data

WFull := wWLd;

CargoVector(XB, YB, wCargo);

end;

Result := wCargo;

end;

// ----------------------------------------------------------------

// 25.01.2017

// Добавить один объект в список Cargo

function TLoading.AddCargoByXY(RqX, RqY : integer; RqTitle : string) : TCargo;

var Ind : integer;

begin

Result := CreateCargo(RqX, RqY);

if Assigned(Result)

then begin

// -------------------------------

// Добавить контейнер в список загрузки

Ind := fLoadingList.Add(Result);

Result.Visible := True;

SetCurrentCargo(Ind);

// --------------------------------

// Добавить контейнер в селектор

fSelectorBox.Items.Add(' № ' + IntToStr(fUniqueCT));

fSelectorBox.ItemIndex := fCargoInd;

// --------------------------------

// Вычислить суммарный вес загрузки

CalcWSumCargo();

// --------------------------------

Inc(fUniqueCT);

end;

end;

// ----------------------------------------------------------------

// 25.01.2017

// Удалить один объект из списока Cargo

procedure TLoading.DeleteCargo(RqInd : integer);

var wCargo : TCargo;

begin

if (RqInd < 0) or (RqInd > fLoadingList.Count - 1) then Exit;

wCargo := TCargo(fLoadingList.Items[RqInd]);

fLoadingList.Delete(RqInd);

// ---------------------------------

// Ликвидировать объект

if Assigned(wCargo)

then begin

if wCargo.Visible then wCargo.Visible := False;

wCargo.Free;

end;

// Вычислить суммарный вес загрузки

CalcWSumCargo();

// Вычислить и отобразить моменты

CalcAndShowMoments();

// ---------------------------------

if fLoadingList.Count > 0

then begin

fCargoInd := fLoadingList.Count - 1;

// Заполнить Grid из Cargo

CargoToGrid();

end

else begin

// Список загрузки стал пустым

fCargoInd := -1;

fUniqueCT := 1;

fGrid.RowCount := 2;

fGrid.Cells[1,1] :='';

end;

// ---------------------------------

fSelectorBox.Items.Delete(RqInd);

end;

// ----------------------------------------------------------------

// 01.02.2017

// Очистить все объекты в зоне загрузки

procedure TLoading.ClearLoading();

var Ind : integer;

begin

if fLoadingList.Count <= 0 then Exit;

for Ind := fLoadingList.Count downto 0

do DeleteCargo(Ind);

end;

// ----------------------------------------------------------------

// 25.01.2017

// Установить и отметить текущий Cargo по указанному индексу

function TLoading.SetCurrentCargo (RqInd : integer) : boolean;

var wCargo : TCargo;

begin

Result := False;

with fImg.Canvas

do begin

// Снять старую отметку

if (fCargoInd >= 0) and (fCargoInd < fLoadingList.Count)

then begin

wCargo := TCargo(fLoadingList.Items[fCargoInd]);

Pen.Color := clBlack;

Brush.Style := bsClear;

Rectangle(wCargo.Cutout);

end;

// Установить новую отметку

if (RqInd >= 0) and (RqInd < fLoadingList.Count)

then begin

fCargoInd := RqInd;

wCargo := TCargo(fLoadingList.Items[fCargoInd]);

Pen.Color := clRed;

Brush.Style := bsClear;

Rectangle(wCargo.Cutout);

Result := True;

// --------------------------------------------

fSelectorBox.ItemIndex := fCargoInd;

// Заполнить Grid из Cargo

CargoToGrid();

// --------------------------------------------

CargoVector(wCargo.XB, wCargo.YB, wCargo);

end;

end;

end;

// ----------------------------------------------------------------

// 30.01.2017

// Обработчик, выполняющий старт прорисовки скользящей фигуры

procedure TLoading.MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

var wCargo : TCargo;

Ind : integer;

begin

// ------------------------------

// Выбрать карго

if (Shift \* [ssShift] = []) and

(Shift \* [ssAlt] = []) and

(Shift \* [ssCtrl] = [])

then begin

fImg.OnMouseUp := nil;

fImg.OnMouseMove := nil;

for Ind := 0 to fLoadingList.Count - 1 do

begin

wCargo := TCargo(fLoadingList.Items[Ind]);

if IsXYInRect(wCargo.Cutout, X, Y)

then begin

SetCurrentCargo(Ind);

Break;

end;

end;

Exit;

end;

// ------------------------------

// Создать (ssShift) новвый карго

if (Shift \* [ssShift] <> []) and

(Shift \* [ssAlt] = []) and

(Shift \* [ssCtrl] = [])

then begin

fImg.OnMouseUp := nil;

fImg.OnMouseMove := nil;

AddCargoByXY(X,Y,'Cargo');

Exit;

end;

// ------------------------------

// Удалить (ssCtrl) карго

if (Shift \* [ssShift] = []) and

(Shift \* [ssAlt] = []) and

(Shift \* [ssCtrl] <> [])

then begin

fImg.OnMouseUp := nil;

fImg.OnMouseMove := nil;

if (fCargoInd < 0) or (fLoadingList.Count < 1) then Exit;

DeleteCargo(fCargoInd);

fCargoInd := -1;

Exit;

end;

// ------------------------------

// Переместить (ssAlt) карго

if (Shift \* [ssShift] = []) and

(Shift \* [ssAlt] <> []) and

(Shift \* [ssCtrl] = [])

then begin

// Контроль объекта TCargo

if (fCargoInd < 0) or (fLoadingList.Count < 1) then Exit;

wCargo := TCargo(fLoadingList.Items[fCargoInd]);

if not Assigned(wCargo) then Exit;

// Убрать изображение с Image

wCargo.Visible := False;

// Переместить wCargo.Rect на начальне координаты X,Y

wCargo.Cutout := MoveRectangle(wCargo.Cutout, X,Y);

// Подготовить параметры для перемещения контура

with fImg.Canvas do

begin

fSavePenMode := Pen.Mode; // Сохранить текущую моду

Pen.Mode := pmNotXor; // Установить моду NotXor

Brush.Style := bsClear; // Установить прозрачную заливку

end;

fContourFlag := True;

fContourShow := False;

// Включить обработчик MouseMove

fImg.OnMouseMove := MouseMove;

fImg.OnMouseUp := MouseUp;

end;

end;

// ----------------------------------------------------------------

// 30.01.2017

// Вернуть контур в область зоны загрузки

procedure TLoading.ReturnRectToLZ(var RqRect : TRect);

var wPix : integer;

begin

if not Assigned(fAirCraft) then Exit;

if IsRectInMainRect(fAirCraft.RectLZ, RqRect) then Exit;

with fAirCraft do

begin

// ---------------------------

// Возврат RqRect в границы RectLZ.Left и RectLZ.Right

if RqRect.Left < RectLZ.Left

then RqRect := MoveRectangle (RqRect, RectLZ.Left, RqRect.Top);

if RqRect.Right > RectLZ.Right

then begin

wPix := RqRect.Right - RqRect.Left;

RqRect := MoveRectangle (RqRect, RectLZ.Right - wPix , RqRect.Top);

end;

// ---------------------------

// Возврат RqRect в границы RectLZ.Top и RectLZ.Bottom

if RqRect.Top < RectLZ.Top

then RqRect := MoveRectangle (RqRect, RqRect.Left, RectLZ.Top);

if RqRect.Bottom > RectLZ.Bottom

then begin

wPix := RqRect.Bottom - RqRect.Top;

RqRect := MoveRectangle (RqRect, RqRect.Left, RectLZ.Bottom - wPix);

end;

end;

end;

// ----------------------------------------------------------------

// 30.01.2017

// Обработчик, выполняющий перемещение контура скользящей фигуры

procedure TLoading.MouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

var wCargo : TCargo;

wRect : TRect;

// XB, YB : extended;

begin

if fContourFlag = False then Exit;

wCargo := TCargo(fLoadingList.Items[fCargoInd]);

with fImg.Canvas do

begin

if fContourShow

then begin

// стереть контур на старом месте

Rectangle(wCargo.Cutout);

end;

// Переместить Rect начальне координаты X,Y

wRect := MoveRectangle(wCargo.Cutout, X,Y);

// Вернуть контур в область зоны загрузки

ReturnRectToLZ (wRect);

wCargo.Cutout := wRect;

// Нарисовать контур на новом месте

Rectangle(wCargo.Cutout);

// Поднять флаг перерисовки

fContourShow := True;

// --------------------------------------------

(\*

// Отобразить координаты перемещения

XB := fAirCraft.XOfsetRect(wRect);

YB := fAirCraft.YOfsetRect(wRect);

CargoVector(XB, YB, wCargo);

\*)

end;

end;

// ----------------------------------------------------------------

// 30.01.2017

// Обработчик, завершающий прорисовку перемещаемой фигуры

procedure TLoading.MouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

var wCargo : TCargo;

begin

// Выключить обработчик MouseMove

fImg.OnMouseMove := nil;

if not fContourFlag then Exit;

// Контроль объекта TCargo

if (fCargoInd < 0) or (fLoadingList.Count < 1) then Exit;

wCargo := TCargo(fLoadingList.Items[fCargoInd]);

if not Assigned(wCargo) then Exit;

// Снять флаг перерисовки

fContourFlag := False;

with fImg.Canvas do

begin

if fContourShow

then begin

// стереть контур на старом месте

Rectangle(wCargo.Cutout);

end;

// Финальная прорисовка

Pen.Mode := fSavePenMode;

end;

with wCargo do

begin

// Отобразить координаты перемещения

XB := fAirCraft.XOfsetRect(Cutout);

YB := fAirCraft.YOfsetRect(Cutout);

CargoVector(XB, YB, wCargo);

// Выполнить финальную прорисовку

Visible := True;

end;

CargoToGrid();

end;

// ----------------------------------------------------------------

// 30.01.2017

// Обработчик событий Click на fSelectorBox

procedure TLoading.ListBoxClick(Sender: TObject);

var Ind : integer;

begin

Ind := fSelectorBox.ItemIndex;

if (Ind >= 0) and (Ind < fLoadingList.Count)

then SetCurrentCargo(Ind);

end;

// =========================================================================

// ПОДДЕРЖКА PROPERTYES

// =========================================================================

// ----------------------------------------------------------------

// Для поддержки property CargoInd

procedure TLoading.SetCargoInd(Ind : integer);

begin

if fLoadingList.Count < 1

then begin

fCargoInd := -1;

Exit;

end;

if (Ind >= 0) and (Ind <= fLoadingList.Count - 1)

then fCargoInd := Ind;

end;

// ----------------------------------------------------------------

// Для поддержки property CargoOBJ

function TLoading.GetCargo(Ind : integer) : TCargo;

begin

Result := nil;

if fLoadingList.Count < 1 then Exit;

if (Ind >= 0) and (Ind <= fLoadingList.Count - 1)

then Result := TCargo(fLoadingList.Items[fCargoInd]);

end;

// ----------------------------------------------------------------

// Для поддержки property CargoCount

function TLoading.GetCargoCount() : integer;

begin

Result := fLoadingList.Count;

end;

// =========================================================================

// ПРИВАТНЫЕ МЕТОДЫ

// =========================================================================

// -------------------------------------------------------------------------

// Отобразить текущий контейнер в fGrid

procedure TLoading.CargoToGrid();

begin

if fCargoInd < 0 then Exit;

if not Assigned(fLoadingList.Items[fCargoInd]) then Exit;

with TCargo(fLoadingList.Items[fCargoInd]) do

begin

fGrid.RowCount := 8;

// Строка 1

fGrid.Cells[0,1] := 'Id';

fGrid.Cells[1,1] := Id;

// Строка 2

fGrid.Cells[0,2] := 'XB (см.)';

fGrid.Cells[1,2] := Format('%6.2f', [XB]);

// Строка 3

fGrid.Cells[0,3] := 'YB (см.)';

fGrid.Cells[1,3] := Format('%6.2f', [YB]);

// Строка 4

fGrid.Cells[0,4] := 'XC (см.)';

fGrid.Cells[1,4] := Format('%6.2f', [XC]);

// Строка 5

fGrid.Cells[0,5] := 'YC (см.)';

fGrid.Cells[1,5] := Format('%6.2f', [YC]);

// Строка 6

fGrid.Cells[0,6] := 'ZC (см.)';

fGrid.Cells[1,6] := Format('%6.2f', [ZC]);

// Строка 7

fGrid.Cells[0,7] := 'Вес (кг.)';

fGrid.Cells[1,7] := Format('%6.2f', [WFull]);

end;

end;

// ================================================================

// СОХРАНЕНИЕ СПИСКА ЗАГРУЗКИ В ФАЙЛ

// ================================================================

// ----------------------------------------------------------------

// 01.02.2017

procedure TLoading.Md5SaveToFile(RqFileName : string);

var wStream : TMemoryStream;

wPBuff : pointer;

begin

// Входной контроль

if (RqFileName = '') or (Length(fMd5Str) <> 32) then Exit;

// Выполнить сохранение в файл

wPBuff := Addr(fMd5Str[1]);

wStream := TMemoryStream.Create();

try

wStream.Write(wPBuff^,Length(fMd5Str));

wStream.SaveToFile(RqFileName + '.md5');

finally

wStream.Free;

end;

end;

// ----------------------------------------------------------------

// 01.02.2017

// Исполнитель сохранения файла

procedure TLoading.SaveToFile(RqFileName : string; RqDelimitor : char);

var wList : TStringList;

wCargo : TCargo;

wMD5Digest : TMD5Digest;

Ind : integer;

wStr : string;

begin

if fLoadingList.Count < 1 then Exit;

try

wList := TStringList.Create;

wList.Add(CommentID + ' ФАЙЛ ЗАГРУЗКИ. СОХРАНЕН : ' + DateToStr(Now()));

wList.Add(CommentID + ' Источник : ' + RqFileName);

for Ind := 0 to fLoadingList.Count - 1 do

begin

wStr := '';

wCargo := TCargo(fLoadingList.Items[Ind]);

// ------------------------------

// Сформировать одну строку файла

// ------------------------------

// Отображаемые в fGrid поля

with wCargo

do begin

wStr := wStr + Format('%3d', [Indx]) + RqDelimitor;

wStr := wStr + Format('%20s', [Id]) + RqDelimitor;

wStr := wStr + Format('%6.2f', [XB]) + RqDelimitor;

wStr := wStr + Format('%6.2f', [YB]) + RqDelimitor;

wStr := wStr + Format('%6.2f', [XC]) + RqDelimitor;

wStr := wStr + Format('%6.2f', [YC]) + RqDelimitor;

wStr := wStr + Format('%6.2f', [ZC]) + RqDelimitor;

wStr := wStr + Format('%6.2f', [WFull]) + RqDelimitor;

// Неотображаемые в fGrid поля

wStr := wStr + Format('%4d', [Cutout.Left]) + RqDelimitor;

wStr := wStr + Format('%4d', [Cutout.Top]) + RqDelimitor;

wStr := wStr + Format('%4d', [Cutout.Right]) + RqDelimitor;

wStr := wStr + Format('%4d', [Cutout.Bottom]) + RqDelimitor;

end;

// ------------------------------

// Сохранить строку

wList.Add(wStr);

end;

// Записать файл на диск

wList.SaveToFile(RqFileName);

fFileName := RqFileName;

wList.Free;

// ----------------------------

// Вычислить MD5

wMD5Digest := MD5File(fFileName);

fMd5Str := MD5DigestToStr(wMD5Digest);

// ----------------------------

// Сохранить Md5 в дополнительный файл

Md5SaveToFile(fFileName);

except

MessageDlg('Ошибка при сохранении файла: '

+ #13#10 + RqFileName,

mtError, [mbOk], 0);

end;

end;

// -----------------------------------------------------------------

// 01.02.2017

// Выполнить в диалоге сохранения данных в файл

function TLoading.DialogSaveToFile() : string;

var wExt : string;

wDialog : TSaveDialog;

wFileExt : string; // Буфер расширения файла

begin

Result := '';

wExt := '.txt';

wDialog := TSaveDialog.Create(nil);

// Установим фильтр

wDialog.Filter := 'CargoList files (\*'

+ LowerCase(wExt) + ')|\*' + UpperCase(wExt);

// Пообщаемся с пользователем

if wDialog.Execute

then begin

Result := wDialog.FileName;

wFileExt := UpperCase(ExtractFileExt(Result));

// Если пользователь не указал расширение, то это выполним мы

if not (wFileExt = UpperCase(wExt))

then Result := Result + LowerCase(wExt);

// Проверим существование файла

if FileExists(Result)

then begin

if MessageDlg('Указанный Файл уже существует.'+ #13#10

+ 'Заменить его на новый?',

mtInformation,[mbYes,mbNo],0) = mrYes

then SaveToFile(Result, char($09));

end

else SaveToFile(Result, char($09));

end;

wDialog.Free;

end;

// ================================================================

// ЗАГРУЗКА ИЗ ФАЙЛА

// ================================================================

// ----------------------------------------------------------------

// 01.02.2017

// Прочитать md5 - строку из файла \*.md5

function TLoading.ReadMd5FromFile(RqFileName : string) : string;

var wStream : TMemoryStream;

wChar : char;

Ind : integer;

begin

// Входной контроль

Result := '';

// Контроль существования файла

if not FileExists(RqFileName + '.md5')

then begin

MessageDlg('MD5 хеш - файл с именем: ' + #13#10

+ RqFileName + '.md5' + #13#10

+ 'Не найден! Проверьте его наличие ...',

mtWarning, [mbOk], 0);

Exit;

end;

// Создать MemoryStream

wStream := TMemoryStream.Create();

try

// Прочитать файл в TMemoryStream

wStream.LoadFromFile(RqFileName + '.md5');

if wStream.Size > 0

then begin

// Выборка потока в строку

for Ind := 0 to (wStream.Size - 1)

do begin

wStream.Read(wChar,1);

Result := Result + wChar;

end;

// ShowMessage(Result); // Для отладки

end;

finally

wStream.Free;

end;

end;

// ----------------------------------------------------------------

// 01.02.2017

// Проверка "строка комментария" для исполнителя загрузки файла

function DetectComment(RqStr : string) : boolean;

var wStr : string;

begin

Result := False;

wStr := Trim(RqStr);

if Length(wStr) >= 2

then begin

if LeftStr(RqStr, 2) = CommentID then Result := True;

end;

end;

// ----------------------------------------------------------------

// 01.02.2017

// Выборка подстроки для исполнителя загрузки файла

function GetSubStr(RqDelimiter : string; var RqStr : string) : string;

var wPos, wCount : integer;

begin

Result := '';

RqStr := Trim(RqStr);

wPos := PosEx(RqDelimiter,RqStr);

if wPos > 0

then begin

wCount := wPos - 1;

Result := Copy(RqStr, 1, wCount);

Delete(RqStr, 1, wPos);

end

else begin

Result := RqStr;

RqStr := '';

end;

end;

// ----------------------------------------------------------------

// 01.02.2017

// Исполнитель загрузки файла

procedure TLoading.LoadFromFile(RqFileName : string; RqDelimitor : char);

var wList : TStringList;

wCargo : TCargo;

wMD5Digest : TMD5Digest;

wMd5Str : string;

Ind1, Ind2 : integer;

wStr : string;

wSubStr : string;

wInt : integer;

wExt : extended;

wRect : TRect;

wErrCount : integer;

begin

// Контроль существования файла

if not FileExists(RqFileName)

then begin

MessageDlg('Файл с запрошенным именем: ' + #13#10

+ RqFileName + #13#10

+ 'Не найден! Проверьте его наличие ...',

mtWarning, [mbOk], 0);

Exit;

end;

if not Assigned(fAirCraft) then Exit;

// -------------------------------------

// ПОПРОБУЕМ ВЫПОЛНИТЬ ЗАГРУЗКУ ФАЙЛА

wErrCount := 0;

wList := TStringList.Create;

try

wList.LoadFromFile(RqFileName);

except

MessageDlg('Ошибка при загрузке файла: '

+ #13#10 + RqFileName

+ #13#10 + 'Проверьте права доступа ...',

mtError, [mbOk], 0);

fFileName := '';

wErrCount := 1;

end;

if wErrCount > 0 then Exit;

// -------------------------------------

// ПОПРОБУЕМ ВЫПОЛНИТЬ ЗАГРУЗКУ ФАЙЛА

// -------------------------------------

fFileName := RqFileName;

// -------------------------------------

// Очистить все объекты в зоне загрузки

ClearLoading();

// -------------------------------------

// Удалим все комментарии

if wList.Count > 0

then begin

for Ind1 := wList.Count -1 downto 0

do if DetectComment(wList.Strings[Ind1])

then wList.Delete(Ind1);

end;

// -------------------------------------

// Добавление объектов в список загрузки

if wList.Count > 0

then begin

fUniqueCT := 1;

for Ind1 := 0 to wList.Count -1 do

begin

// Создание объекта для загрузки

wCargo := TCargo.Create(fImg);

// Инициализация полей объекта

wErrCount := 0;

wStr := wList.Strings[Ind1];

// ---------------------------

// Отображаемые в fGrid поля

// ---------------------------

wSubStr := GetSubStr(RqDelimitor, wStr);

if TryStrToInt(wSubStr, wInt)

then wCargo.Indx := wInt

else Inc(wErrCount);

// ---------------------------

wCargo.Id := GetSubStr(RqDelimitor, wStr);

// ---------------------------

wSubStr := GetSubStr(RqDelimitor, wStr);

if TryStrToFloat(wSubStr, wExt)

then wCargo.XB := wExt

else Inc(wErrCount);

// ---------------------------

wSubStr := GetSubStr(RqDelimitor, wStr);

if TryStrToFloat(wSubStr, wExt)

then wCargo.YB := wExt

else Inc(wErrCount);

// ---------------------------

wSubStr := GetSubStr(RqDelimitor, wStr);

if TryStrToFloat(wSubStr, wExt)

then wCargo.XC := wExt

else Inc(wErrCount);

// ---------------------------

wSubStr := GetSubStr(RqDelimitor, wStr);

if TryStrToFloat(wSubStr, wExt)

then wCargo.YC := wExt

else Inc(wErrCount);

// ---------------------------

wSubStr := GetSubStr(RqDelimitor, wStr);

if TryStrToFloat(wSubStr, wExt)

then wCargo.ZC := wExt

else Inc(wErrCount);

// ---------------------------

wSubStr := GetSubStr(RqDelimitor, wStr);

if TryStrToFloat(wSubStr, wExt)

then wCargo.WFull := wExt

else Inc(wErrCount);

// ---------------------------

// Неотображаемые в fGrid поля

// ---------------------------

wSubStr := GetSubStr(RqDelimitor, wStr);

if TryStrToInt(wSubStr, wInt)

then wRect.Left := wInt

else Inc(wErrCount);

// ---------------------------

wSubStr := GetSubStr(RqDelimitor, wStr);

if TryStrToInt(wSubStr, wInt)

then wRect.Top := wInt

else Inc(wErrCount);

// ---------------------------

wSubStr := GetSubStr(RqDelimitor, wStr);

if TryStrToInt(wSubStr, wInt)

then wRect.Right := wInt

else Inc(wErrCount);

// ---------------------------

wSubStr := GetSubStr(RqDelimitor, wStr);

if TryStrToInt(wSubStr, wInt)

then wRect.Bottom := wInt

else Inc(wErrCount);

// ---------------------------

wCargo.Cutout := wRect;

// ---------------------------

if not (wErrCount > 0)

then begin

// -------------------------------

// Добавить контейнер в список загрузки

Ind2 := fLoadingList.Add(wCargo);

wCargo.Visible := True;

SetCurrentCargo(Ind2);

// --------------------------------

// Добавить контейнер в селектор

fSelectorBox.Items.Add(' № ' + IntToStr(wCargo.Indx));

fSelectorBox.ItemIndex := fCargoInd;

// --------------------------------

Inc(fUniqueCT);

end

else begin

MessageDlg('Ошибка при загрузке контейнера : '

+ #13#10 + wCargo.Id

+ #13#10 + 'Проверьте его формат его полей...',

mtError, [mbOk], 0);

wCargo.Free;

// -------------

fFileName := '';

Break;

end;

// ---------------------------

end; // of for

// --------------------------------

// Вычислить суммарный вес загрузки

CalcWSumCargo();

// --------------------------------

// ВЫПОЛНИТЬ КОНТРОЛЬ ПО ХЕШ-КОДАМ

if not (wErrCount > 0)

then begin

// Вычислить MD5 загруженного файла

wMD5Digest := MD5File(fFileName);

fMd5Str := MD5DigestToStr(wMD5Digest);

// Прочитать из файла контрольный Md5

wMd5Str := ReadMd5FromFile(fFileName);

if fMd5Str <> wMd5Str

then begin

MessageDlg('Вычисленный (В) и контрольный (К) MD5 различны:'

+ #13#10 + 'В = ' + fMd5Str

+ #13#10 + 'К = ' + wMd5Str,

mtError, [mbOk], 0);

end;

end;

end; // of if wList.Count > 0

end;

// -----------------------------------------------------------------

// 01.02.2017

// Выполнить в диалоге загрузку данных из файла

// с разделителями полей табуляцией (char($09))

function TLoading.DialogLoadFromFile() : string;

var wExt : string;

wDialog : TOpenDialog;

begin

Result := '';

wExt := '.txt';

wDialog := TOpenDialog.Create(nil);

// Установим фильтр

wDialog.Filter := 'CargoList files (\*'

+ LowerCase(wExt) + ')|\*' + UpperCase(wExt);

// Пообщаемся с пользователем

if wDialog.Execute

then begin

Result := wDialog.FileName;

LoadFromFile(Result, char($09));

end;

wDialog.Free;

end;

// ========================================================================

// КОНЕЦ

// ========================================================================

end.

## unit AirCraft01;

interface

uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,

Dialogs, ExtCtrls,

UserApp03Data;

// -------------------------------------------------------------------------

// -------------------------------------------------------------------------

type TAirCraft = class(TObject)

private

// -----------------

// Исходные данные

fId : string; // Название самолета

fFName : string; // Короткое имя файла картинки зоны загрузки

fLenX : extended; // Длина зоны загрузки (см.)

fLenY : extended; // Ширина зоны загрузки (см.)

fWeight : extended; // Максимальный вес загрузки

fRectLZ : TRect; // Rect зоны загрузки на картинке

// -----------------

// Расчетные данные

fScaleX : extended; // Масштаб для преобразования в пиксельные размеры

fScaleY : extended; // Масштаб для преобразования в пиксельные размеры

// -----------------

// Вычислить масштабы для преобразования длин в пиклельные размеры

function CalcScale() : boolean;

public

// ------------------------------------------

constructor Create(RqAirCraftRec : TAirCraftRec);

// ------------------------------------------

// Прорисовать зону загрузки на внешнем Image

procedure ShowLoadingZone(RqImg : TImage);

// Построить Rect по начальным координатам и длинам

function XYLenToRect (RqXB,RqYB : integer;

RqLenX,RqLenY : extended) : TRect;

// Y - Смещение RqRect в зоне загрузки

function YOfsetRect(RqRect : TRect) : extended;

// X - Смещение RqRect в зоне загрузки

function XOfsetRect(RqRect : TRect) : extended;

// ------------------------------------------

property Id : string read fId;

property FName : string read fFName;

property LenX : extended read fLenX;

property LenY : extended read fLenY;

property Weight : extended read fWeight;

property RectLZ : TRect read fRectLZ;

property ScaleX : extended read fScaleX;

property ScaleY : extended read fScaleY;

end;

// ========================================================================

// ========================================================================

implementation

// ========================================================================

// ========================================================================

// ------------------------------------------------------------------------

// Вычислить масштабы для преобразования длин в пиклельные размеры

function TAirCraft.CalcScale() : boolean;

begin

Result := False;

// Входной контроль

fScaleX := 0;

fScaleY := 0;

if (fLenX < 1) or (fLenY < 1)

then Exit;

if ((fRectLZ.Right - fRectLZ.Left) < 1) or

((fRectLZ.Bottom - fRectLZ.Top) < 1)

then Exit;

// Расчет масштабов

try

fScaleX := (fRectLZ.Right - fRectLZ.Left) / fLenX;

fScaleY := (fRectLZ.Bottom - fRectLZ.Top) / fLenY;

Result := True;

except

fScaleX := 0;

fScaleY := 0;

end;

end;

// ------------------------------------------------------------------------

// X - Смещение RqRect в зоне загрузки

function TAirCraft.XOfsetRect(RqRect : TRect) : extended;

begin

if fScaleX > 0

then Result := (RqRect.Left - fRectLZ.Left) / fScaleX

else Result := 0;

end;

// ------------------------------------------------------------------------

// Y - Смещение RqRect в зоне загрузки

function TAirCraft.YOfsetRect(RqRect : TRect) : extended;

begin

if fScaleY > 0

then Result := (RqRect.Top - fRectLZ.Top) / fScaleY

else Result := 0;

end;

// ------------------------------------------------------------------------

constructor TAirCraft.Create(RqAirCraftRec : TAirCraftRec);

begin

inherited Create;

with RqAirCraftRec do

begin

fId := Id;

fFName := FName;

fLenX := LenX;

fLenY := LenY;

fWeight := Weight;

fRectLZ := RectLZ;

end;

CalcScale();

end;

// ------------------------------------------------------------------------

// Прорисовать зону загрузки на внешнем Image

procedure TAirCraft.ShowLoadingZone(RqImg : TImage);

var wX, wY : integer;

begin

with RqImg.Canvas do

begin

// ------------------------------------------

// Прорисовка плоскоти зоны загрузки

Pen.Color := clGreen;

Pen.Width := 2;

Brush.Style := bsClear;

Rectangle(fRectLZ);

(\* // Для отладки

// Начало осей координат на плоскости зоны загрузки

Ellipse(fRectLZ.Left - 3, fRectLZ.Top - 3,

fRectLZ.Left + 3, fRectLZ.Top + 3 );

TextOut(fRectLZ.Left + 8, fRectLZ.Top + 4, 'X0,Y0 - зоны загрузки');

\*)

// ------------------------------------------

// Прорисовка проекции осей координат в центре

// масс самолета на плоскость зоны загрузки

Pen.Color := clBlue;

Pen.Width := 1;

Pen.Style := psDash;

wY := fRectLZ.Top + Round(fScaleY \* fLenY / 2);

MoveTo(fRectLZ.Left, wY);

LineTo(fRectLZ.Right, wY);

wX := fRectLZ.Left + Round(fScaleX \* fLenX / 2);

MoveTo(wX, fRectLZ.Top);

LineTo(wX, fRectLZ.Bottom);

Pen.Style := psSolid;

(\* // Для отладки

TextOut(wX + 8, wY + 4, 'X0,Y0 - проекции центра масс самолета');

\*)

end;

end;

// ------------------------------------------------------------------------

// Построить Rect по начальным координатам и длинам

function TAirCraft.XYLenToRect (RqXB,RqYB : integer;

RqLenX,RqLenY : extended) : TRect;

var wXL, wYL : integer;

begin

// X - Длина в пикселях

if fScaleX > 0

then wXL := Round(fScaleX \* RqLenX)

else wXL := 2;

// Y - Длина в пикселях

if fScaleY > 0

then wYL := Round(fScaleY \* RqLenY)

else wYL := 2;

// ------------------------------

Result := Rect(RqXB,RqYB, RqXB + wXL,RqYB + wYL);

end;

// ========================================================================

// КОНЕЦ

// ========================================================================

end.

## unit MomentScope01;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,

Dialogs, ExtCtrls, StdCtrls,

UserApp03Data, Service01;

type TMomentScope = class(TObject)

private

// ------------------------

fImg : TImage; // Арендуемый Image

// ------------------------

fScRect : TRect; // Несущий прямоугольник индикатора

fXYZero : TPoint; // Точка начала координат

fAZnRect : TRect; // Rect определяющий оси X и Y

fPnMRoll : TPoint; // Начало координат для момента по крену

fPnMPitch : TPoint; // Начало координат для момента по тангажу

fPnMTurn : TPoint; // Начало координат для момента по повороту

// ------------------------

fBmp : TBitMap; // Копия площадки занимаемой индикатором

fVisible : boolean; // Состояние видимости

// -----------------------

fMaxMoment : extended; // Максимальный момент для индикаторов

fMomentDim : string; // Размерность момента

// -----------------------

fMScale : extended; // Масштаб для пребразования значений в пиксели

fMRoll : extended; // Момент по крену

fMPitch : extended; // Момент по тангажу

fMTurn : extended; // Момент по развороту

// -----------------------

// Управление видимостью индикатора

procedure SetVisible(RqVisible : boolean);

// Расчет масштаба для пребразования значений момента в пиксели

procedure SetMScale(RqMaxMoment : extended);

// Установка максимального момента на всех индикаторах

procedure SetMaxMoment(RqMaxMoment : extended);

// Установка и прорисовка индикатора крена

procedure SetMRoll(RqMRoll : extended);

// Установка и прорисовка индикатора тангажа

procedure SetMPitch(RqMPitch : extended);

// Установка и прорисовка индикатора поворота

procedure SetTurn(RqMTurn : extended);

// Расчет пиксельной длины текущего значения для индикатора

function CalcScopePix(RqValue : extended) : integer;

// Прорисовать миниатюру самолета

procedure ShowAirCraft(RqX, RqY : integer);

// Отобразить графическое значение индикатора

procedure ShowGraphScope (RqPoint : TPoint;

RqOriente : char;

RqValue : extended);

// Отобразить цифровое значение на индикаторе

procedure ShowScopeValue(RqTop : integer;

RqValue : extended;

RqTitle : string);

// Полная прорисовка

procedure ShowAll();

public

// ------------------------

// Инициирование / терминирование

constructor Create(RqImage : TImage; XBSc, YBSc : integer);

procedure Free();

// ------------------------

property Visible : boolean read fVisible write SetVisible;

// Максимальный момент для индикаторов

property MaxMoment : extended read fMaxMoment write SetMaxMoment;

// Размерность для значений моментов

property MomentDim : string read fMomentDim write fMomentDim;

// Момент по крену

property MRoll : extended read fMRoll write SetMRoll;

// Момент по тангажу

property MPitch : extended read fMPitch write SetMPitch;

// Момент по развороту

property MTurn : extended read fMTurn write SetTurn;

end;

implementation

// ================================================================

// КОНСТАНТЫ И ТИПЫ ЮНИТА

// ================================================================

// ----------------------------------------------------------------

// Миниатюра самолета

const AC01 : array[0..13] of TPoint =

( // Верхняя часть копуса

(X : 000; Y : 034),

(X : 008; Y : 027),

(X : 016; Y : 025),

// Руль поворота

(X : 130; Y : 025),

(X : 145; Y : 004),

(X : 150; Y : 004),

(X : 146; Y : 025),

// Концевая точка копуса

(X : 150; Y : 025),

(X : 148; Y : 028),

// Нижняя часть копуса

(X : 120; Y : 036),

(X : 100; Y : 038),

(X : 075; Y : 040),

(X : 015; Y : 040),

(X : 000; Y : 036)

);

const AC02 : array[0..4] of TPoint =

( // Нижнее крыло

(X : 054; Y : 030),

(X : 022; Y : 070),

(X : 032; Y : 070),

(X : 084; Y : 030),

(X : 054; Y : 030)

);

const AC03 : array[0..4] of TPoint =

( // Верхнее крыло

(X : 060; Y : 025),

(X : 092; Y : 000),

(X : 100; Y : 000),

(X : 086; Y : 025),

(X : 060; Y : 025)

);

// ----------------------------------------------------------------

// Габаритные расчеты основных констант

const // ------------

XLenSc = 260; // X - габарит несущего Rect индикаторов

YLenSc = 200; // Y - габарит несущего Rect индикаторов

// ------------

XLenCr = 150; // X - габарит миниатюры самолета

YLenCr = 70; // Y - габарит миниатюры самолета

// ------------

XBAx = 80; // X - сдвиг Rect осей относительно несущего Rect

YBAx = 18; // Y - сдвиг Rect осей относительно несущего Rect

HLenAx = 50; // Половина длины и ширины Rect осей

// ------------

XOffCr = -66; // X - сдвиг миниатюры относительно начала координат

YOffCr = -30; // Y - сдвиг миниатюра относительно начала координат

// ----------------------------------------------------------------

const ScopeLen = HLenAx; // Максимум индикаторов в пикселях

ScopeMaxMScale = 1; // Максиммальный масштаб

// ------------

ScopeTxtXB = 140; // X - сдвиг подписей относительно fScRect

ScopeTxtYB = 86; // Y - сдвиг подписей относительно fScRect

// ================================================================

// ИНИЦИАЛИЗАЦИЯ / ТЕРМИНИРОВАНИЕ

// ================================================================

// ----------------------------------------------------------------

constructor TMomentScope.Create(RqImage : TImage; XBSc, YBSc : integer);

begin

inherited Create;

// ---------------

// Подготовка fImg

fImg := RqImage;

if not Assigned(fImg.Picture)

then begin

fImg.Picture := TPicture.Create;

with fImg.Picture.Bitmap

do begin

if Width <> fImg.Width then Width := fImg.Width;

if Height <> fImg.Height then Height := fImg.Height;

if PixelFormat <> pf24bit then PixelFormat := pf24bit;

end;

end;

// ---------------

// Расчет положения несущего Rect индикатора на арендуемом Image

fScRect := Rect(Point(XBSc,YBSc),Point(XBSc + XLenSc,YBSc + YLenSc));

// Расчет прямоугольника для прорисовки осей координат

fAZnRect := Rect(XBSc + XBAx, YBSc + YBAx,

XBSc + XBAx + 2 \* HLenAx,

YBSc + YBAx + 2 \* HLenAx);

// ---------------

// Расчет начала общих координат

fXYZero.X := fAZnRect.Left + HLenAx;

fXYZero.Y := fAZnRect.Top + HLenAx;

// ---------------

// Начало координат для нулей индикаторов моментов :

// по крену

fPnMRoll.X := fAZnRect.Left - 20;

fPnMRoll.Y := fAZnRect.Bottom + 20;

// по тангажу

fPnMPitch.X := fAZnRect.Left - 40;

fPnMPitch.Y := fXYZero.Y;

// по повороту

fPnMTurn := fPnMRoll;

// ---------------

SetMaxMoment(1000);

fMomentDim := 'кг \* м';

// ---------------

fBmp := TBitMap.Create;

// ---------------

fVisible := False;

self.SetVisible(True);

end;

// -----------------------------------------------------------------

procedure TMomentScope.Free();

begin

if Assigned(fBmp)

then begin

if fVisible then SetVisible(False);

fBmp.Free();

end;

inherited Free();

end;

// ================================================================

// ПРИВАТНЫЕ МЕТОДЫ

// ================================================================

// ----------------------------------------------------------------

// Расчет масштаба для пребразования значений момента в пиксели

procedure TMomentScope.SetMScale(RqMaxMoment : extended);

begin

fMScale := ScopeMaxMScale;

if Abs(RqMaxMoment) > 1e-6

then begin

fMScale := Abs(ScopeLen/RqMaxMoment);

if fMScale > ScopeMaxMScale then fMScale := ScopeMaxMScale;

end;

end;

// ----------------------------------------------------------------

// 02.02.2017

// Прорисовать миниатюру самолета

procedure TMomentScope.ShowAirCraft(RqX, RqY : integer);

var wArr : array of Tpoint;

Ind : integer;

begin

with fImg.Canvas

do begin

// ---------------------------

SetLength(wArr, Length(AC01));

for Ind := Low(wArr) to High(wArr)

do begin

wArr[Ind].X := RqX + AC01[Ind].X;

wArr[Ind].Y := RqY + AC01[Ind].Y;

end;

Brush.Color := RGB(160,210,210);

Polygon(wArr);

// ---------------------------

SetLength(wArr, Length(AC02));

for Ind := Low(wArr) to High(wArr)

do begin

wArr[Ind].X := RqX + AC02[Ind].X;

wArr[Ind].Y := RqY + AC02[Ind].Y;

end;

Brush.Color := RGB(190,255,255);

Polygon(wArr);

// ---------------------------

SetLength(wArr, Length(AC03));

for Ind := Low(wArr) to High(wArr)

do begin

wArr[Ind].X := RqX + AC03[Ind].X;

wArr[Ind].Y := RqY + AC03[Ind].Y;

end;

Brush.Color := RGB(190,255,255);

Polygon(wArr);

// ---------------------------

// Прорисовка осей

Pen.Color := clBlue;

// Ось X

MoveTo(fPnMPitch.X, fPnMPitch.Y);

LineTo(fXYZero.X, fXYZero.Y);

// Ось Y

MoveTo(fPnMRoll.X, fPnMRoll.Y);

LineTo(fXYZero.X, fXYZero.Y);

// LineTo(fAZnRect.Right, fAZnRect.Top);

end;

end;

// ----------------------------------------------------------------

// Расчет пиксельной длины текущего значения для индикатора

function TMomentScope.CalcScopePix(RqValue : extended) : integer;

begin

Result := Round(fMScale \* RqValue);

if Abs(Result) > ScopeLen

then if Result >= 0

then Result := ScopeLen

else Result := - ScopeLen;

end;

// ----------------------------------------------------------------

// Отобразить цифровое значение на индикаторе

procedure TMomentScope.ShowScopeValue(RqTop : integer;

RqValue : extended;

RqTitle : string);

var wLeft, wTop : integer;

begin

with fImg.Canvas do

begin

Brush.Style := bsClear;

wLeft := fScRect.Left + ScopeTxtXB;

wTop := fScRect.Top + ScopeTxtYB + RqTop;

// Вывод значения

if RqValue > 0

then Font.Color := RGB(255,0,0)

else Font.Color := RGB(0,0,255);

TextOut(wLeft + 45, wTop, ': ' + Format('%6.1f',[RqValue]));

// Вывод подписи значения

if Abs(RqValue) <= fMaxMoment

then Font.Color := clBlack;

TextOut(wLeft, wTop, RqTitle);

end;

end;

// ----------------------------------------------------------------

// Отобразить графическое значение индикатора

procedure TMomentScope.ShowGraphScope (RqPoint : TPoint;

RqOriente : char;

RqValue : extended);

var wPix : integer;

begin

with fImg.Canvas do

begin

Brush.Style := bsSolid;

if RqValue > 0

then Brush.Color := RGB(255,0,0)

else Brush.Color := RGB(0,0,255);

Pen.Color := Brush.Color;

wPix := CalcScopePix(RqValue);

case RqOriente of

'V' : Rectangle(RqPoint.X - 2, RqPoint.Y,

RqPoint.X + 2, RqPoint.Y - wPix);

'H' : Rectangle(RqPoint.X, RqPoint.Y - 2,

RqPoint.X + wPix, RqPoint.Y + 2);

end;

end;

end;

// ----------------------------------------------------------------

// Полная прорисовка

procedure TMomentScope.ShowAll();

begin

if not fVisible then Exit;

with fImg.Canvas do

begin

// ---------------------------

// Полная очистка

Pen.Width := 1;

Pen.Color := clBlack;

Font.Color := clBlack;

Brush.Style := bsSolid;

Brush.Color := RGB(240,240,240); // clWindow;

Rectangle(fScRect);

// ---------------------------

// Заголовок

TextOut(fScRect.Left + 60, fScRect.Top + 4, 'Моменты относительно');

TextOut(fScRect.Left + 60,

fScRect.Top + TextHeight('1') + 4, 'центра масс самолета');

// ---------------------------

// Прорисовка миниатюры самолета и осей

ShowAirCraft(fXYZero.X + XOffCr, fXYZero.Y + YOffCr);

// ---------------------------

// Прорисовка индикаторов моментов

// ---------------------------

// индикатор крена

ShowGraphScope (fPnMRoll, 'V', fMRoll);

ShowScopeValue( 0, fMRoll, 'Крен');

// индикатор тангажа

ShowGraphScope (fPnMPitch, 'V', fMPitch);

ShowScopeValue(TextHeight('1'), fMPitch, 'Тангаж');

// индикатор поворота

ShowGraphScope (fPnMTurn, 'H', fMTurn);

ShowScopeValue( 2\* TextHeight('1'), fMTurn, 'Поворот');

// ---------------------------

// Подвал

Font.Color := clBlack;

TextOut(fScRect.Left + 100,

fScRect.Bottom - 40,

'Моменты отображаются');

TextOut(fScRect.Left + 100,

fScRect.Bottom - 40 + TextHeight('1'),

'в размерности : ' + fMomentDim);

end;

end;

// ================================================================

// ПОДДЕРЖКА PROPERTYEs

// ================================================================

// ----------------------------------------------------------------

// Управление видимостью индикатора

procedure TMomentScope.SetVisible(RqVisible : boolean);

begin

if RqVisible

then begin

if (not fVisible)

then begin

// Сохранить площадку под индикатором

if SaveBitMap

(fImg.Picture.Bitmap, // BitMap источник

fBmp, // BitMap приемник

fScRect) // Rect источник

then begin

// Отобразить индикатор на fImg

fVisible := True;

ShowAll();

fImg.Repaint; // Обновить fImg

end;

end;

end

else begin

if fVisible

then begin

// Востановить площадку под индикатором

if RestoreBitMap

(fBmp, // BitMap источник

fImg.Picture.Bitmap, // BitMap приемник

fScRect.Left, fScRect.Top) // Начальные X,Y в приемнике

then begin

fVisible := False;

fImg.Repaint; // Обновить fImg

end;

end;

end;

end;

// ----------------------------------------------------------------

// Установка максимального момента на всех индикаторах

procedure TMomentScope.SetMaxMoment(RqMaxMoment : extended);

begin

fMaxMoment := Abs(RqMaxMoment);

SetMScale(fMaxMoment);

end;

// ----------------------------------------------------------------

// Установка и прорисовка индикатора крена

procedure TMomentScope.SetMRoll(RqMRoll : extended);

begin

fMRoll := RqMRoll;

if fVisible then ShowAll();

end;

// Установка и прорисовка индикатора тангажа

procedure TMomentScope.SetMPitch(RqMPitch : extended);

begin

fMPitch := RqMPitch;

if fVisible then ShowAll();

end;

// Установка и прорисовка индикатора поворота

procedure TMomentScope.SetTurn(RqMTurn : extended);

begin

fMTurn := RqMTurn;

if fVisible then ShowAll();

end;

// ================================================================

// КОНЕЦ

// ================================================================

end.

## unit md5;

{\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* }

{ Вычисление хеш-суммы MD5 }

{ Зависимости: Windows, SysUtils, Classes }

{ Источник : Indy 8.1 }

{ Дата : 19 июня 2002 г. }

{\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* }

{\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* }

{ MD5 Hashsum Evaluation Unit For Borland Delphi }

{ Derived from the RSA Data Security, Inc. }

{ MD5 Message-Digest Algorithm described in RFC 1321 }

{ http://www.faqs.org/rfcs/rfc1321.html }

{\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* }

interface

uses Windows, SysUtils, Classes;

type

{ Тип TMD5Digest используется для получения

результата функций вычисления хеш-суммы.

Содержимое записи можно использовать

как набор из 4 целых чисел, или как

массив из 16 байт }

PMD5Digest = ^TMD5Digest;

TMD5Digest = record

case Integer of

0: (A, B, C, D: LongInt);

1: (v: array[0..15] of Byte);

end;

// вычисление хеш-суммы для строки

function MD5String(const S: string): TMD5Digest;

// вычисление хеш-суммы для файла

function MD5File(const FileName: string): TMD5Digest;

// вычисление хеш-суммы для содержиого потока Stream

function MD5Stream(const Stream: TStream): TMD5Digest;

// вычисление хеш-суммы для произвольного буфера

function MD5Buffer(const Buffer; Size: Integer): TMD5Digest;

// преобразование хеш-суммы в строку из шестнадцатеричных цифр

function MD5DigestToStr(const Digest: TMD5Digest): string;

// сравнение двух хеш-сумм

function MD5DigestCompare(const Digest1, Digest2: TMD5Digest): Boolean;

implementation

{

Copyright (C) 1991-2, RSA Data Security, Inc. Created 1991. All

rights reserved.

License to copy and use this software is granted provided that it

is identified as the "RSA Data Security, Inc. MD5 Message-Digest

Algorithm" in all material mentioning or referencing this software

or this function.

License is also granted to make and use derivative works provided

that such works are identified as "derived from the RSA Data

Security, Inc. MD5 Message-Digest Algorithm" in all material

mentioning or referencing the derived work.

RSA Data Security, Inc. makes no representations concerning either

the merchantability of this software or the suitability of this

software for any particular purpose. It is provided "as is"

without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this

documentation and/or software.

}

type

UINT4 = LongWord;

PArray4UINT4 = ^TArray4UINT4;

TArray4UINT4 = array[0..3] of UINT4;

PArray2UINT4 = ^TArray2UINT4;

TArray2UINT4 = array[0..1] of UINT4;

PArray16Byte = ^TArray16Byte;

TArray16Byte = array[0..15] of Byte;

PArray64Byte = ^TArray64Byte;

TArray64Byte = array[0..63] of Byte;

PByteArray = ^TByteArray;

TByteArray = array[0..0] of Byte;

PUINT4Array = ^TUINT4Array;

TUINT4Array = array[0..0] of UINT4;

PMD5Context = ^TMD5Context;

TMD5Context = record

state: TArray4UINT4;

count: TArray2UINT4;

buffer: TArray64Byte;

end;

const

S11 = 7;

S12 = 12;

S13 = 17;

S14 = 22;

S21 = 5;

S22 = 9;

S23 = 14;

S24 = 20;

S31 = 4;

S32 = 11;

S33 = 16;

S34 = 23;

S41 = 6;

S42 = 10;

S43 = 15;

S44 = 21;

var

Padding: TArray64Byte =

($80, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);

function \_F(x, y, z: UINT4): UINT4;

begin

Result := (((x) and (y)) or ((not x) and (z)));

end;

function \_G(x, y, z: UINT4): UINT4;

begin

Result := (((x) and (z)) or ((y) and (not z)));

end;

function \_H(x, y, z: UINT4): UINT4;

begin

Result := ((x) xor (y) xor (z));

end;

function \_I(x, y, z: UINT4): UINT4;

begin

Result := ((y) xor ((x) or (not z)));

end;

function ROTATE\_LEFT(x, n: UINT4): UINT4;

begin

Result := (((x) shl (n)) or ((x) shr (32 - (n))));

end;

procedure FF(var a: UINT4; b, c, d, x, s, ac: UINT4);

begin

a := a + \_F(b, c, d) + x + ac;

a := ROTATE\_LEFT(a, s);

a := a + b;

end;

procedure GG(var a: UINT4; b, c, d, x, s, ac: UINT4);

begin

a := a + \_G(b, c, d) + x + ac;

a := ROTATE\_LEFT(a, s);

a := a + b;

end;

procedure HH(var a: UINT4; b, c, d, x, s, ac: UINT4);

begin

a := a + \_H(b, c, d) + x + ac;

a := ROTATE\_LEFT(a, s);

a := a + b;

end;

procedure II(var a: UINT4; b, c, d, x, s, ac: UINT4);

begin

a := a + \_I(b, c, d) + x + ac;

a := ROTATE\_LEFT(a, s);

a := a + b;

end;

procedure MD5Encode(Output: PByteArray; Input: PUINT4Array; Len: LongWord);

var

i, j: LongWord;

begin

j := 0;

i := 0;

while j < Len do

begin

output[j] := Byte(input[i] and $FF);

output[j + 1] := Byte((input[i] shr 8) and $FF);

output[j + 2] := Byte((input[i] shr 16) and $FF);

output[j + 3] := Byte((input[i] shr 24) and $FF);

Inc(j, 4);

Inc(i);

end;

end;

procedure MD5Decode(Output: PUINT4Array; Input: PByteArray; Len: LongWord);

var

i, j: LongWord;

begin

j := 0;

i := 0;

while j < Len do

begin

Output[i] := UINT4(input[j]) or (UINT4(input[j + 1]) shl 8) or

(UINT4(input[j + 2]) shl 16) or (UINT4(input[j + 3]) shl 24);

Inc(j, 4);

Inc(i);

end;

end;

procedure MD5\_memcpy(Output: PByteArray; Input: PByteArray; Len: LongWord);

begin

Move(Input^, Output^, Len);

end;

procedure MD5\_memset(Output: PByteArray; Value: Integer; Len: LongWord);

begin

FillChar(Output^, Len, Byte(Value));

end;

procedure MD5Transform(State: PArray4UINT4; Buffer: PArray64Byte);

var

a, b, c, d: UINT4;

x: array[0..15] of UINT4;

begin

a := State[0];

b := State[1];

c := State[2];

d := State[3];

MD5Decode(PUINT4Array(@x), PByteArray(Buffer), 64);

FF(a, b, c, d, x[0], S11, $D76AA478);

FF(d, a, b, c, x[1], S12, $E8C7B756);

FF(c, d, a, b, x[2], S13, $242070DB);

FF(b, c, d, a, x[3], S14, $C1BDCEEE);

FF(a, b, c, d, x[4], S11, $F57C0FAF);

FF(d, a, b, c, x[5], S12, $4787C62A);

FF(c, d, a, b, x[6], S13, $A8304613);

FF(b, c, d, a, x[7], S14, $FD469501);

FF(a, b, c, d, x[8], S11, $698098D8);

FF(d, a, b, c, x[9], S12, $8B44F7AF);

FF(c, d, a, b, x[10], S13, $FFFF5BB1);

FF(b, c, d, a, x[11], S14, $895CD7BE);

FF(a, b, c, d, x[12], S11, $6B901122);

FF(d, a, b, c, x[13], S12, $FD987193);

FF(c, d, a, b, x[14], S13, $A679438E);

FF(b, c, d, a, x[15], S14, $49B40821);

GG(a, b, c, d, x[1], S21, $F61E2562);

GG(d, a, b, c, x[6], S22, $C040B340);

GG(c, d, a, b, x[11], S23, $265E5A51);

GG(b, c, d, a, x[0], S24, $E9B6C7AA);

GG(a, b, c, d, x[5], S21, $D62F105D);

GG(d, a, b, c, x[10], S22, $2441453);

GG(c, d, a, b, x[15], S23, $D8A1E681);

GG(b, c, d, a, x[4], S24, $E7D3FBC8);

GG(a, b, c, d, x[9], S21, $21E1CDE6);

GG(d, a, b, c, x[14], S22, $C33707D6);

GG(c, d, a, b, x[3], S23, $F4D50D87);

GG(b, c, d, a, x[8], S24, $455A14ED);

GG(a, b, c, d, x[13], S21, $A9E3E905);

GG(d, a, b, c, x[2], S22, $FCEFA3F8);

GG(c, d, a, b, x[7], S23, $676F02D9);

GG(b, c, d, a, x[12], S24, $8D2A4C8A);

HH(a, b, c, d, x[5], S31, $FFFA3942);

HH(d, a, b, c, x[8], S32, $8771F681);

HH(c, d, a, b, x[11], S33, $6D9D6122);

HH(b, c, d, a, x[14], S34, $FDE5380C);

HH(a, b, c, d, x[1], S31, $A4BEEA44);

HH(d, a, b, c, x[4], S32, $4BDECFA9);

HH(c, d, a, b, x[7], S33, $F6BB4B60);

HH(b, c, d, a, x[10], S34, $BEBFBC70);

HH(a, b, c, d, x[13], S31, $289B7EC6);

HH(d, a, b, c, x[0], S32, $EAA127FA);

HH(c, d, a, b, x[3], S33, $D4EF3085);

HH(b, c, d, a, x[6], S34, $4881D05);

HH(a, b, c, d, x[9], S31, $D9D4D039);

HH(d, a, b, c, x[12], S32, $E6DB99E5);

HH(c, d, a, b, x[15], S33, $1FA27CF8);

HH(b, c, d, a, x[2], S34, $C4AC5665);

II(a, b, c, d, x[0], S41, $F4292244);

II(d, a, b, c, x[7], S42, $432AFF97);

II(c, d, a, b, x[14], S43, $AB9423A7);

II(b, c, d, a, x[5], S44, $FC93A039);

II(a, b, c, d, x[12], S41, $655B59C3);

II(d, a, b, c, x[3], S42, $8F0CCC92);

II(c, d, a, b, x[10], S43, $FFEFF47D);

II(b, c, d, a, x[1], S44, $85845DD1);

II(a, b, c, d, x[8], S41, $6FA87E4F);

II(d, a, b, c, x[15], S42, $FE2CE6E0);

II(c, d, a, b, x[6], S43, $A3014314);

II(b, c, d, a, x[13], S44, $4E0811A1);

II(a, b, c, d, x[4], S41, $F7537E82);

II(d, a, b, c, x[11], S42, $BD3AF235);

II(c, d, a, b, x[2], S43, $2AD7D2BB);

II(b, c, d, a, x[9], S44, $EB86D391);

Inc(State[0], a);

Inc(State[1], b);

Inc(State[2], c);

Inc(State[3], d);

MD5\_memset(PByteArray(@x), 0, SizeOf(x));

end;

procedure MD5Init(var Context: TMD5Context);

begin

FillChar(Context, SizeOf(Context), 0);

Context.state[0] := $67452301;

Context.state[1] := $EFCDAB89;

Context.state[2] := $98BADCFE;

Context.state[3] := $10325476;

end;

procedure MD5Update(var Context: TMD5Context; Input: PByteArray; InputLen:

LongWord);

var

i, index, partLen: LongWord;

begin

index := LongWord((context.count[0] shr 3) and $3F);

Inc(Context.count[0], UINT4(InputLen) shl 3);

if Context.count[0] < UINT4(InputLen) shl 3 then

Inc(Context.count[1]);

Inc(Context.count[1], UINT4(InputLen) shr 29);

partLen := 64 - index;

if inputLen >= partLen then

begin

MD5\_memcpy(PByteArray(@Context.buffer[index]), Input, PartLen);

MD5Transform(@Context.state, @Context.buffer);

i := partLen;

while i + 63 < inputLen do

begin

MD5Transform(@Context.state, PArray64Byte(@Input[i]));

Inc(i, 64);

end;

index := 0;

end

else

i := 0;

MD5\_memcpy(PByteArray(@Context.buffer[index]), PByteArray(@Input[i]), inputLen

- i);

end;

procedure MD5Final(var Digest: TMD5Digest; var Context: TMD5Context);

var

bits: array[0..7] of Byte;

index, padLen: LongWord;

begin

MD5Encode(PByteArray(@bits), PUINT4Array(@Context.count), 8);

index := LongWord((Context.count[0] shr 3) and $3F);

if index < 56 then

padLen := 56 - index

else

padLen := 120 - index;

MD5Update(Context, PByteArray(@PADDING), padLen);

MD5Update(Context, PByteArray(@Bits), 8);

MD5Encode(PByteArray(@Digest), PUINT4Array(@Context.state), 16);

MD5\_memset(PByteArray(@Context), 0, SizeOf(Context));

end;

function MD5DigestToStr(const Digest: TMD5Digest): string;

var

i: Integer;

begin

Result := '';

for i := 0 to 15 do

Result := Result + IntToHex(Digest.v[i], 2);

end;

function MD5String(const S: string): TMD5Digest;

begin

Result := MD5Buffer(PChar(S)^, Length(S));

end;

function MD5File(const FileName: string): TMD5Digest;

var

F: TFileStream;

begin

F := TFileStream.Create(FileName, fmOpenRead);

try

Result := MD5Stream(F);

finally

F.Free;

end;

end;

function MD5Stream(const Stream: TStream): TMD5Digest;

var

Context: TMD5Context;

Buffer: array[0..4095] of Byte;

Size: Integer;

ReadBytes: Integer;

TotalBytes: Integer;

SavePos: Integer;

begin

MD5Init(Context);

Size := Stream.Size;

SavePos := Stream.Position;

TotalBytes := 0;

try

Stream.Seek(0, soFromBeginning);

repeat

ReadBytes := Stream.Read(Buffer, SizeOf(Buffer));

Inc(TotalBytes, ReadBytes);

MD5Update(Context, @Buffer, ReadBytes);

until (ReadBytes = 0) or (TotalBytes = Size);

finally

Stream.Seek(SavePos, soFromBeginning);

end;

MD5Final(Result, Context);

end;

function MD5Buffer(const Buffer; Size: Integer): TMD5Digest;

var

Context: TMD5Context;

begin

MD5Init(Context);

MD5Update(Context, PByteArray(@Buffer), Size);

MD5Final(Result, Context);

end;

function MD5DigestCompare(const Digest1, Digest2: TMD5Digest): Boolean;

begin

Result := False;

if Digest1.A <> Digest2.A then

Exit;

if Digest1.B <> Digest2.B then

Exit;

if Digest1.C <> Digest2.C then

Exit;

if Digest1.D <> Digest2.D then

Exit;

Result := True;

end;

// ========================================================================

// КОНЕЦ

// ========================================================================

end.