Annexe - Le code du simulateur *B*

EtatClass.pas

```
unit EtatClass;

interface

type

// ------------ Mise en forme de la classe --------------------------------
TEtat = class

  constructor Create;
  constructor
CreateAV(SV:Integer;EV:Integer;CV:Integer;PV:Integer;PIV:Integer);
  Destructor Destroy;

private
  // -- Les Variables --
  SocleValeur : integer;
  EpauleValeur : integer;
  CoudeValeur : integer;
  PoignetValeur : integer;
  PinceValeur : integer;

public
  // -- Les Méthodes --
  procedure SetSocleValeur(Val:Integer);
  procedure SetEpauleValeur(Val:Integer);
  procedure SetCoudeValeur(Val:Integer);
  procedure SetPoignetValeur(Val:Integer);
  procedure SetPinceValeur(Val:Integer);

  function GetSocleValeur():Integer;
  function GetEpauleValeur():Integer;
  function GetCoudeValeur():Integer;
  function GetPoignetValeur():Integer;
  function GetPinceValeur():Integer;

end;
// ------------ Fin de la mise en forme ----------------------------------

implementation

// ------------ Implémentation des méthodes de TEtat ----------------------
  Constructor TEtat.Create();
  begin
      SocleValeur:=0;
      EpauleValeur:=0;
      CoudeValeur:=0;
      PoignetValeur:=0;
      PinceValeur:=0;
  end;
```

```
  Constructor
TEtat.CreateAV(SV:Integer;EV:Integer;CV:Integer;PV:Integer;PIV:Integer);
  begin
      SocleValeur:=SV;
      EpauleValeur:=EV;
      CoudeValeur:=CV;
      PoignetValeur:=PV;
      PinceValeur:=PIV;
  end;

  Destructor  TEtat.Destroy();
  begin
  end;

// ------------ Procedures Publique----------------------------------------

  { Pour chaque variable on a prevu une méthode pour l'écrire et le lire
    de la variable.}

  procedure TEtat.SetSocleValeur(Val:Integer);
  begin
    SocleValeur:=Val;
  end;

  procedure TEtat.SetEpauleValeur(Val:Integer);
  begin
    EpauleValeur:=Val;
  end;

  procedure TEtat.SetCoudeValeur(Val:Integer);
  begin
    CoudeValeur:=Val;
  end;

  procedure TEtat.SetPoignetValeur(Val:Integer);
  begin
    PoignetValeur:=Val;
  end;

  procedure TEtat.SetPinceValeur(Val:Integer);
  begin
    PinceValeur:=Val;
  end;


  function TEtat.GetSocleValeur():Integer;
  begin
    GetSocleValeur:=SocleValeur;
  end;

  function TEtat.GetEpauleValeur():Integer;
  begin
    GetEpauleValeur:=EpauleValeur;
  end;

  function TEtat.GetCoudeValeur():Integer;
  begin
    GetCoudeValeur:=CoudeValeur;
  end;
```

```
  function TEtat.GetPoignetValeur():Integer;
  begin
    GetPoignetValeur:=PoignetValeur;
  end;

  function TEtat.GetPinceValeur():Integer;
  begin
    GetPinceValeur:=PinceValeur;
  end;

end.
```

RobotClass.pas

```
unit RobotClass;

interface

uses Windows, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls,
ExtCtrls, Messages, EtatClass, RobotViewerClass;

type

TRobot = class;
// ------------ Mise en forme de la classe TMoteur ------------------------
TMoteur = class(TComponent)

  constructor
Create(AOwner:Tcomponent;v:integer;n:String;ax:integer;ix:integer;p:integer
;r:TRobot);//override;
  Destructor Destroy;

private
  // -- Les Variables --
  Timer:TTimer;
  Interval:Cardinal;

  OnOff:boolean; {S'il est en marche ou pas}
  DroiteGauche:boolean; // 1 --> droite, 0 --> gauche {La direction qu'il
prend}

  Nom:string;
  AngleMin:integer;
  AngleMax:integer;
  Position:integer;
  Vitesse:integer;

  Robot:TRobot;

public
  // -- Les Méthodes --
  procedure TourneDroite();
  procedure TourneGauche();
  procedure Bouger(Sender:TObject);
  procedure StopBouger();
  function GetPosition():integer;
  procedure SetPosition(val:integer);
  procedure SetPositionButee(val:integer);
```

```
end;
// ------------ Fin de la mise en forme ----------------------------------


// ------------ Mise en forme de la classe -------------------------------
TRobot = class

  Constructor Create;
  Destructor Destroy;

private
  // -- Les Variables --
  {Ces cinq moteurs}
  SocleMoteur : TMoteur;
  EpauleMoteur : TMoteur;
  CoudeMoteur : TMoteur;
  PinceMoteur : TMoteur;
  PoignetMoteur : TMoteur;
  {Son état}
  EtatRobot : TEtat;

  RobotViewer : TRobotViewer;
  {Les butées de ces cinq moteurs}
  ButeesMoteurs : array [0..5] of integer; //1 butée gauche, 2 pas butée
                                                  // 3 butée droite
public
  // -- Les Méthodes --
  procedure setRobotViewer(RV:TRobotViewer);
  function Bouche(FW:byte):String;
  procedure metMaValeur(MType:String; Valeur:Integer);
  procedure metMaButee(MType:String;BGoD:Integer);

  procedure
SetButees(SV:integer;EV:integer;CV:integer;PV:integer;PIV:integer);
  procedure SetPositionCoude(val:integer);
  procedure SetPositionPoignet(val:integer);

  procedure ArreteTout();
end;
// ------------ Fin de la mise en forme ----------------------------------


implementation

//---------------------------------------------------------------------------
// Des TROBOT
//---------------------------------------------------------------------------

  {Creation des 5 moteur du robot avec leurs valuers des vitesse, nom, etc,
   qui se trouvent dans un fichier "PropRobot.ini".}
  Constructor TRobot.Create();
  var
    t1,butee : integer;
    vitesse, anglemax, anglemin, position : integer;
    nom : string;

    iFileHandle, iFileLength, iBytesRead:Integer;
    Buffer:PChar;
    i,place:Integer;
    KindValue,Value:String;
```

```
  begin
    try

      place := 1; KindValue := ''; Value := '';

      {Ouvrir d'un fichier et lire les données dans un buffer}
      iFileHandle :=FileOpen('PropRobot.ini',fmOpenRead);
      iFileLength :=FileSeek(iFileHandle,0,2);
      FileSeek(iFileHandle,0,0);
      Buffer :=PChar(AllocMem(iFileLength +1));
      iBytesRead :=FileRead(iFileHandle,Buffer^,iFileLength);
      FileClose(iFileHandle);

      {L'analyse de cette buffer, trouver la partie devant '='
       et devant ';'}
      for i := 0 to iBytesRead-1 do
      begin
        if ((Buffer[i] <> '=') And (Buffer[i] <> ';') And (place = 1)) then
        begin
          KindValue := KindValue + Buffer[i];
        end;
        if (Buffer[i] = '=') then
        begin
          place := 2;
        end;
        if ((Buffer[i] <> '=') And (Buffer[i] <> ';') And (place = 2)) then
        begin
          Value := Value + Buffer[i];
        end;
        if (Buffer[i] = ';') then
        begin
          place := 3;
          {De quelle propriété il s'agit et mettre dans la propre variable}
          if (KindValue = 'nom') then
            nom := Value;
          if (KindValue = 'vitesse') then
            vitesse := StrToInt(Value);
          if (KindValue = 'angleMax') then
            anglemax := StrToInt(Value);
          if (KindValue = 'angleMin') then
            anglemin := StrToInt(Value);
          if (KindValue = 'position') then
          begin
            if (position <= angleMin) then   {La possibilé qu'un robot
                                               commence}
              butee := 3;                    {a une position deja butée}
            if (position >= angleMax) then
              butee := 1;
            if ((position > angleMin) And (position < angleMax))then
              butee := 2;
            position := StrToInt(Value);
            case nom[1] of                   {La création des cinq moteurs}
              'S'{ocle'} : SocleMoteur :=
TMoteur.Create(TComponent(Self),vitesse,nom,anglemax,anglemin,position,self
);
              'E'{paule'} : EpauleMoteur :=
TMoteur.Create(TComponent(Self),vitesse,nom,anglemin,-anglemax,-
position,self);
              'C'{oude'} : CoudeMoteur :=
TMoteur.Create(TComponent(Self),vitesse,nom,anglemin,-anglemax,-
position,self);
```

```
              'P'{oignet'} : PoignetMoteur :=
TMoteur.Create(TComponent(Self),vitesse,nom,anglemax,anglemin,position,self
);
              'N'{pince'} : PinceMoteur :=
TMoteur.Create(TComponent(Self),vitesse,nom,anglemax,anglemin,position,self
);
            end;
            case nom[1] of                  {Les valeurs des butées}
              'S'{ocle'} : ButeesMoteurs[1] := butee;
              'E'{paule'} : ButeesMoteurs[2] := butee;
              'C'{oude'} : ButeesMoteurs[3] := butee;
              'P'{oignet'} : ButeesMoteurs[4] := butee;
              'N'{pince'} : ButeesMoteurs[5] := butee;
            end;
          end;

          KindValue := '';
          Value := '';
        end;
        if (Buffer[i] = #$A) then           {Parce que après un ; il
vient}
          place := 1;                       {encore un caractère nouveau
ligne}

      end;
    finally
      FreeMem(Buffer);
    end;
                                    {L'état: les positions des tous les
moteurs}
    EtatRobot :=
TEtat.CreateAV(SocleMoteur.Position,EpauleMoteur.Position,CoudeMoteur.Posit
ion,PoignetMoteur.Position,PinceMoteur.Position);

  end;

  {Le déstructeur de la clase TRobot}
  Destructor TRobot.Destroy();
  begin
    SocleMoteur.Destroy;
    EpauleMoteur.Destroy;
    CoudeMoteur.Destroy;
    PinceMoteur.Destroy;
    PoignetMoteur.Destroy;
    RobotViewer := nil;

    EtatRobot.Free;

    SocleMoteur.Free;
    EpauleMoteur.Free;
    CoudeMoteur.Free;
    PinceMoteur.Free;
    PoignetMoteur.Free;
    RobotViewer.Free;
  end;

  procedure TRobot.setRobotViewer(RV:TRobotViewer);
  begin
    RobotViewer := RV;
    RobotViewer.setEtat(EtatRobot);
  end;
```

```
{Dire au robot, bouge un de tes moteurs à une direction déterminé. Il te
 renvoie un string avec dedans la valeur qui indique un butée du moteur}
function TRobot.Bouche(FW:byte):string;
begin

  if FW = $E1 then
    begin
      PinceMoteur.TourneGauche;
      Bouche := '5' + IntToStr(ButeesMoteurs[5]);
    end;
  if FW = $D2 then
    begin
      PoignetMoteur.TourneGauche;
      Bouche := '4' + IntToStr(ButeesMoteurs[4]);
    end;
  if FW = $E3 then
    begin
      CoudeMoteur.TourneGauche;
      Bouche := '3' + IntToStr(ButeesMoteurs[3]);
    end;
  if FW = $D4 then
    begin
      EpauleMoteur.TourneGauche;
      Bouche := '2' + IntToStr(ButeesMoteurs[2]);
    end;
  if FW = $D5 then
    begin
      SocleMoteur.TourneGauche;
      Bouche := '1' + IntToStr(ButeesMoteurs[1]);
    end;
  if FW = $D1 then
    begin
      PinceMoteur.TourneDroite;
      Bouche := '5' + IntToStr(ButeesMoteurs[5]);
    end;
  if FW = $E2 then
    begin
      PoignetMoteur.TourneDroite;
      Bouche := '4' + IntToStr(ButeesMoteurs[4]);
    end;
  if FW = $D3 then
    begin
      CoudeMoteur.TourneDroite;
      Bouche := '3' + IntToStr(ButeesMoteurs[3]);
    end;
  if FW = $E4 then
    begin
      EpauleMoteur.TourneDroite;
      Bouche := '2' + IntToStr(ButeesMoteurs[2]);
    end;
  if FW = $E5 then
    begin
      SocleMoteur.TourneDroite;
      Bouche := '1' + IntToStr(ButeesMoteurs[1]);
    end;
  if FW = $C1 then
    begin
      PinceMoteur.StopBouger;
      Bouche := '5' + IntToStr(ButeesMoteurs[5]);
    end;
```

```
    if FW = $C2 then
      begin
        PoignetMoteur.StopBouger;
        Bouche := '4' + IntToStr(ButeesMoteurs[4]);
      end;
    if FW = $C3 then
      begin
        CoudeMoteur.StopBouger;
        Bouche := '3' + IntToStr(ButeesMoteurs[3]);
      end;
    if FW = $C4 then
      begin
        EpauleMoteur.StopBouger;
        Bouche := '2' + IntToStr(ButeesMoteurs[2]);
      end;
    if FW = $C5 then
      begin
        SocleMoteur.StopBouger;
        Bouche := '1' + IntToStr(ButeesMoteurs[1]);
      end;
  end;

  {Un Moteur peut mettre sa valeur de son position actuel}
  procedure TRobot.metMaValeur(MType:String; Valeur:Integer);
  begin
    case MType[1] of
      'S'{ocle'} : EtatRobot.SetSocleValeur(Valeur); // 1 --> Socle
      'E'{paule'} : EtatRobot.SetEpauleValeur(Valeur); // 2 --> Epaule
      'C'{oude'} : EtatRobot.SetCoudeValeur(Valeur); // 3 --> Coude
      'P'{oignet'} : EtatRobot.SetPoignetValeur(Valeur); // 4 --> Poignet
      'N'{pince'} : EtatRobot.SetPinceValeur(Valeur); // 5 --> Pince
    end;

    RobotViewer.setEtat(EtatRobot)

  end;

  {Un Moteur peut mettre sa valeur s'il est butée ou pas}
  procedure TRobot.metMaButee(MType:String;BGoD:Integer);
  begin
    case MType[1] of
      'S'{ocle'} : ButeesMoteurs[1] := BGoD; // 1 --> Socle
      'E'{paule'} : ButeesMoteurs[2] := BGoD; // 2 --> Epaule
      'C'{oude'} : ButeesMoteurs[3] := BGoD; // 3 --> Coude
      'P'{oignet'} : ButeesMoteurs[4] := BGoD; // 4 --> Poignet
      'N'{pince'} : ButeesMoteurs[5] := BGoD; // 5 --> Pince
    end;
  end;

  {Faire suivre les valeurs des butées au moteurs}
  procedure
TRobot.SetButees(SV:integer;EV:integer;CV:integer;PV:integer;PIV:integer);
  begin
    SocleMoteur.SetPosition(SV);
    EpauleMoteur.SetPosition(EV);
    CoudeMoteur.SetPosition(CV);
    PoignetMoteur.SetPosition(PV);
    PinceMoteur.SetPosition(PIV);
  end;
```

```
  {Mettre la position de la coude}
  procedure TRobot.SetPositionCoude(val:integer);
  begin
    CoudeMoteur.SetPosition(360-val);
  end;

  {Mettre la position de le poignet}
  procedure TRobot.SetPositionPoignet(val:integer);
  begin
    PoignetMoteur.SetPosition(360-val);
  end;

  {Arrèter tous les moteurs du robot}
  procedure TRobot.ArreteTout;
  begin
    PoignetMoteur.StopBouger;
    CoudeMoteur.StopBouger;
    EpauleMoteur.StopBouger;
    SocleMoteur.StopBouger;
  end;
//---------------------------------------------------------------------------


//---------------------------------------------------------------------------
// Des TMOTEUR
//---------------------------------------------------------------------------

  constructor
TMoteur.Create(AOwner:Tcomponent;v:integer;n:String;ax:integer;ix:integer;p
:integer;r:TRobot);
  begin
    Timer := TTimer.Create(Self);
    vitesse := v;
    Interval := vitesse;
    Timer.Interval := Interval;
    Timer.Enabled := False;
    Timer.OnTimer := Bouger;

    Robot := r;
    Nom := n;
    AngleMin := ix;
    AngleMax := ax;
    Position := p;
    DroiteGauche := True; // true = droite, false = gauche
  end;

  Destructor  TMoteur.Destroy;
  begin
    Timer.Free;
  end;

  {Dire au robot qu'il doit tourner à droite}
  procedure TMoteur.TourneDroite;
  begin
    OnOff := True;
    DroiteGauche := False;
    Timer.Enabled := True;
  end;
```

```
  {Dire au robot qu'il doit tourner à gauche}
  procedure TMoteur.TourneGauche;
  begin
    OnOff := True;
    DroiteGauche := True;
    Timer.Enabled := True;
  end;

  {chaque fois si le moteur est en marche, il voit sa direction et s'il
n'est
  pas à la fin de son angle.}
  procedure TMoteur.Bouger;
  begin
    if OnOff = True then               {S'il est en marche}
    begin
      if DroiteGauche = true then      {Si la direction est Gauche}
        begin
          if Position < AngleMax then  {S'il n'arrive pas en butée}
          begin
            Position := Position + 1;
            robot.metMaValeur(nom,Position);
            robot.metMaButee(nom,2);
          end
          else
          begin                        {S'il arrive en butée}
            Timer.Enabled := False;
            robot.metMaButee(nom,1);
          end
        end
     else
        begin
          if Position > AngleMin then  {s'il n'arrive pas en butée}
          begin
            Position := Position - 1;
            robot.metMaValeur(nom,Position);
            robot.metMaButee(nom,2);
          end
          else
          begin                        {s'il arrive en butée}
            Timer.Enabled := False;
            robot.metMaButee(nom,3);
          end
        end
     end
  end;

  {Arrèter le moteur}
  procedure TMoteur.StopBouger;
  begin
    OnOff := False;
    Timer.Enabled := False;
  end;

  {retourne sa position}
  function TMoteur.GetPosition():integer;
  begin
    GetPosition := integer(position);
  end;
```

```
  {mettre un nouveau position}
  procedure TMoteur.SetPosition(val:integer);
  begin
      position := val;
      robot.metMaValeur(nom,Position);
  end;

  {mettre en butée indiqué}
  procedure TMoteur.SetPositionButee(val:integer);
  begin
    if val = 1 then
      position := AngleMin;
    if val = 3 then
      position := AngleMax;

    robot.metMaValeur(nom,Position);
    robot.metMaButee(nom,val);
  end;
//---------------------------------------------------------------------------

end.
```

RobotViewer.pas

```
unit RobotViewerClass;

interface

uses

  GLScene, GLObjects, EtatClass, GLWin32Viewer, GLMisc, GLGeomObjects,
Jpeg,
  Graphics, SysUtils;

type

// ------------ Mise en forme de la classe ------------------------------
TRobotViewer = class

  constructor Create(S:TGLCylinder; A1:TGLCylinder; A2:TGLCylinder;
P:TGLCylinder; D1:TGLDummyCube; D2:TGLDummyCube; GLSC:TGLSceneViewer);
  destructor Destroy;

private

  // -- Les choses pour le GLScene --
  Socle: TGLCylinder;
  Articul1: TGLCylinder;
  Articul2: TGLCylinder;
  Poignet: TGLCylinder;
  Doigt1: TGLDummyCube;
  Doigt2: TGLDummyCube;
  GLSViewer: TGLSceneViewer;
  // -- Les Méthodes --
  procedure save_image();
```

```
public

  // -- Les Méthodes --
  procedure setEtat(Etat:TEtat);

end;
// ------------ Fin de la mise en forme ----------------------------------

implementation

// ------------ Implementation des methodes de TRobotViewer ---------------
  constructor TRobotViewer.Create(S:TGLCylinder; A1:TGLCylinder;
A2:TGLCylinder; P:TGLCylinder; D1:TGLDummyCube; D2:TGLDummyCube;
GLSC:TGLSceneViewer);
  begin
    Socle:=S;
    Articul1:=A1;
    Articul2:=A2;
    Poignet:=P;
    Doigt1:=D1;
    Doigt2:=D2;
    GLSViewer:=GLSC;
  end;

  destructor TRobotViewer.Destroy;
  begin
    Socle := nil;
    Articul1 := nil;
    Articul2 := nil;
    Poignet := nil;
    Doigt1 := nil;
    Doigt2 := nil;
  end;

// ------------ Procedures Privé-------------------------------------------
  {Pour la visualisation sur internet, sauvegarde la scene de GLscene dans
un
   fichier du format JPEG}
  procedure TRobotViewer.save_image();
  var
    bmp:TBitmap;
    jpg:TJpegImage;
  begin
    bmp:=TBitmap.Create;
    bmp.PixelFormat:=pf32bit;
    bmp.Height:=250;
    bmp.Width:=250;
    GLSViewer.Buffer.RenderToBitmap(bmp,50);
    jpg:=TJpegImage.Create;
    jpg.Assign(bmp);
    jpg.CompressionQuality:=50;
    {Effacer la dernière temp}
    DeleteFile('c:\\webserver\www\robottmp.jpg');
    {Sauvegarder le nouveau}
    jpg.SaveToFile('c:\\webserver\www\robottmp.jpg');
    {Effacer le buffer du JPEG}
    jpg.Free;
    {Effacer la dernière}
    DeleteFile('c:\\webserver\www\robot.jpg');
    {rénomer le temp}
```

```
RenameFile('c:\\webserver\www\robottmp.jpg','c:\\webserver\www\robot.jpg');
    {Effacer le buffer du BMP}
    bmp.FreeImage
  end;

// ------------ Procedures Publique ---------------------------------------
  {Modifier la visualisation en 3D avec les valeurs des moteurs}
  procedure TRobotViewer.setEtat(Etat:TEtat);
  begin

    Doigt1.PitchAngle:= - Etat.GetPinceValeur;
    Doigt2.PitchAngle:= Etat.GetPinceValeur;
    socle.TurnAngle := Etat.GetSocleValeur;
    Articul1.TurnAngle :=  - Etat.GetEpauleValeur;
    Articul2.TurnAngle := - Etat.GetCoudeValeur;
    Poignet.TurnAngle := Etat.GetPoignetValeur;

    save_image();

  end;

end.
```

TcpClient.pas

```
unit TcpClass;

interface

uses Windows, Messages, SysUtils, Variants, Classes, Dialogs, Forms,
 WSocket, WSockets;

type

  TReceiveAction = procedure(Sender: TObject; Action: byte) of Object;

// ------- Mise en forme de la classe -------------------------------------
TcpClient = class(TWSocketClient)

private

  // -- Les Variables --
  FReceiveAction: TReceiveAction;

  // -- Les Méthodes --
  procedure SocketDataAvailable(Sender: TObject; Error: Word);

public
  // -- Les Méthodes --
  procedure Initialize;
  procedure Finalize;

  // -- Properties --
  property OnReceive: TReceiveAction read FReceiveAction write
FreceiveAction;

end;
// ------- Fin de la mise en forme ----------------------------------------
```

```
implementation

uses Unit1;

// ------- Implémentation des méthodes de TCPClient -----------------------

{La méthode OnDataAvailable est provoque dans un TWSocketClient, la classe
ou
 on dérive cette classe-ci. Alors comme ca on dit s'il y a des données qu
 arrivent tu doit provoqué la méthode SocketDataAvailable}
procedure TcpClient.Initialize;
begin
  OnDataAvailable := SocketDataAvailable;
end;

{Cette méthode est maintenant provoquer s'il y arrive quelque chose}
procedure TcpClient.SocketDataAvailable(Sender: TObject; Error: Word);
var
  Data : String;
  BData : byte;
begin

  Data := TWSocket(Sender).ReceiveStr;

  if Data <> '' then
    BData :=  ord(Data[1]); {On transforme en Byte}

  FReceiveAction(Self, BData); {On l'envoie aux écran}
end;

procedure TcpClient.Finalize;
begin

end;

end.
```

UdpClient.pas

```
unit UdpClass;

interface

uses Windows, Messages, SysUtils, Variants, Classes, Dialogs, Forms,
 WSocket, WSockets, ExtCtrls, POSITIONS;

type

// ------------ Mise en forme de la classe --------------------------------
TUdpClient = class(TComponent)

  constructor
Create(AOwner:Tcomponent;P:string;A:string;S:integer;E:TFrmPosEcran);
  Destructor Destroy;
```

```
  private
    // -- Les Variables --
    Port:string;
    Adress:string;
    UDPSocket: TWSocket;

    Timer:TTimer;
    Interval:Cardinal;

    Ecran:TFrmPosEcran;

    BufferTMot : array [0..5] of string;
    BufferTMotTeller : Integer;

    UdpClientOccuper : boolean;   {Si le socket est occupé}
    UdpClientFoB : boolean;       {Ce qu'il doit faire,False -> FW, True ->
Butees}

    Data:string;                  {Ce qu'il reçoit}

    FW1:string;                   {Les Flagwords, qu indique quel moteur
tourne}
    FW2:string;                   {à quelle direction}
    FW3:string;
    FW4:string;
    FW5:string;

    MEW0:string;                  {Input word0:consiste quelques valeurs
butées}
    MMW50:string;                 {Flag word50:consiste quelques valeurs
butées}

    SV,EV,CV,PV,PIV:integer;      {Valeurs des butée, chacun réprésente un
moteur}

    CoudePosition,PoignetPosition:integer;

    // -- Les Methodes --
    procedure GetFlagWords(Sender:TObject);
    procedure GetButees(Sender:TObject);
    procedure GetValues();
    procedure UDPSocketDataAvailable(Sender: TObject; Error: Word);
    procedure AnalyseData();
    function BinToDig(BinStr:String):Integer;
    function CalculeMew0EnHex():string;
    function pow(base, power: integer): integer;
    procedure SendButee();

  public

    procedure CommenceComIPCSR(); //Commence la com. avec l'IPC sans robot
    procedure CommenceComIPCAR(); //Commence la com. avec l'IPC avec robot

    procedure ArreteComIPC();

    procedure SetButee(RV:String);
    // -- Properties --
    property PFW1 : string read FW1; //FW1: fonctionnement de pince
    property PFW2 : string read FW2; //FW2: fonctionnement de poignet
    property PFW3 : string read FW3; //FW3: fonctionnement de coude
    property PFW4 : string read FW4; //FW4: fonctionnement de epaule
```

```
    property PFW5 : string read FW5; //FW5: fonctionnement de socle

    property PSV : integer read SV;
    property PEV : integer read EV;
    property PCV : integer read CV;
    property PPV : integer read PV;
    property PPIV : integer read PIV;

    property PCoudePosition : integer read CoudePosition;
    property PPoignetPosition : integer read PoignetPosition;

end;
// ------------ Fin de la mise en forme ----------------------------------

implementation

// ------------ Implementation des methodes de TUDPCLIENT -----------------

constructor
TUdpClient.Create(AOwner:Tcomponent;P:string;A:string;S:integer;E:TFrmPosEc
ran);
begin
  Port := P;
  Adress := A;
  UDPSocket := TWSocket.Create(Self);
  UDPSocket.OnDataAvailable := UDPSocketDataAvailable;

  Interval := S;
  Timer := TTimer.Create(Self);
  Timer.Interval := Interval;
  Timer.Enabled := False;

  Ecran := E;
  BufferTMotTeller := 1;
  UdpClientOccuper := false;

  FW1 := '193';
  FW2 := '194';
  FW3 := '195';
  FW4 := '196';
  FW5 := '197';

  MEW0 := '00000000';
  MMW50 := '00000000';

  SV:=2; EV:=2; CV:=2; PV:=2; PIV:=2;
end;

Destructor TUdpClient.Destroy;
begin
    Timer.Free;
    UDPSocket.Free;
    inherited Destroy;
end;
```

```
// ------------ Procedures Privé-----------------------------------------

{Met tous les mots qu'on a besoin pour retrouver les flagwords dans un
chaîne
 et commence la boucle des envoyer.}
procedure TUdpClient.GetFlagWords(Sender:TObject);
begin
  if UdpClientOccuper = False then
  begin
    BufferTmotTeller := 1;
    BufferTMot[1] := 'DMW1'; //Pince
    BufferTMot[2] := 'DMW2'; //Poignet
    BufferTMot[3] := 'DMW3'; //Coude
    BufferTMot[4] := 'DMW4'; //Epaule
    BufferTMot[5] := 'DMW5'; //Socle
    UdpClientOccuper := True;
    GetValues();
  end;
end;

{Met tous les mots qu'on a besoin pour retrouver les valeurs des butées et
des
 angles de la coude et du poignet. Commence la boucle des envoyer.}
procedure TUdpClient.GetButees(Sender:TObject);
begin
  if UdpClientOccuper = False then
  begin
    BufferTmotTeller := 1;
    BufferTMot[1] := 'DEW0'; //Epaule et Socle
    BufferTMot[2] := 'DMW50'; //Pince et poignet
    BufferTMot[3] := 'DEW12'; //Angle coude
    BufferTMot[4] := 'DEW13'; //Angle poignet
    UdpClientOccuper := True;
    GetValues();
  end;
end;

{Envoyer un mot au IPC, il peut savoir quel mot avec le compteur
BufferTMotTeller
 Qui est augmenter après chaque reçu.}
procedure TUdpClient.GetValues();
begin

  UdpSocket.Proto:='udp';
  UdpSocket.Addr:=Adress;
  UdpSocket.Port:=Port;
  UdpSocket.Connect;
  UdpSocket.SendStr(BufferTMot[BufferTMotTeller]);

end;
```

```
{Chaque fois il a envoié un mot, il attande sur une réponse. Il n'envoie
pas un
 nouveau mot parce que le variable UdpClientOccuper est Vrai pour le
moment. Si
 la réponse arrive, il l'analyse et augmente le compteur}
procedure TUdpClient.UDPSocketDataAvailable(Sender: TObject; Error: Word);
begin

  Data := UdpSocket.ReceiveStr;
  UdpSocket.Close;

  AnalyseData();

  if UdpClientFoB = False then //recevoir les FlagWords
  begin
    if BufferTMotTeller < 5 then
    begin
      BufferTMotTeller := BufferTMotTeller + 1;
      GetValues();
    end
    else
    begin
      SendButee();
    end;
  end;

  if UdpClientFoB = True then //recevoir les Butees
  begin
    if BufferTMotTeller < 4 then
    begin
      BufferTMotTeller := BufferTMotTeller + 1;
      GetValues();
    end;
  end;

end;

{Une réponse consiste de deux partie, la première avec le mot envoié et la
 deuxième avec la vrai réponse. Cela on doit d'abord dériver. Après on peut
 voir avec la première partie qoui on doit faire avec la deuxième.}
procedure TUdpClient.AnalyseData();
var
  reqmot,valeur : String;
  t1,t2,chiffre : Integer;
begin

  //cherche dans le resultat le nom du demande
  reqmot := '';
  for t1 := 1 to 3 do
  begin
    reqmot := reqmot + Data[t1];
  end;

  //cherche dan le resultat le valeur du demande
  t1 := pos('=',Data);
  valeur := '';
  for t2 := t1+1 to length(Data) do
  begin
    valeur := valeur + Data[t2];
  end;
```

```
  if UdpClientFoB = False then //recevoir les FlagWords
  begin
    if reqmot = 'DMW' then
    begin
      case Data[4] of
        '1':FW1 := valeur;
        '2':FW2 := valeur;
        '3':FW3 := valeur;
        '4':FW4 := valeur;
        '5':FW5 := valeur;
      end;
    end;
  end;

  if UdpClientFoB = True then //recevoir les valeurs des angles et des
butées
  begin
    chiffre := StrToInt(valeur);

    if (reqmot = 'DEW') And (BufferTMotTeller = 1) then
    begin                                                {Mettre un
masque}
      SV:=2; EV:=2;
      if (chiffre And $08) = $08 then //Butee Epaule basse
        EV := 3;
      if (chiffre And $10) = $10 then //Butee Epaule haute
        EV := 1;
      if (chiffre And $20) = $20 then //Butee Socle gauche
        SV := 1;
      if (chiffre And $40) = $40 then //Butee Socle milieux
        SV := 2;
      if (chiffre And $80) = $80 then //Butee Socle droite
        SV := 3;
    end;
    if (reqmot = 'DMW') And (BufferTMotTeller = 2) then
    begin
      PV:=2; PIV:=2;
      if (chiffre And $10) = $10 then //Butee Pince fermee
        PIV := 1;
      if (chiffre And $20) = $20 then //Butee Pince ouvert
        PIV := 3;
      if (chiffre And $40) = $40 then //Butee Poignet gauche
        PV := 1;
      if (chiffre And $80) = $80 then //Butee Poignet droite
        PV := 3;
    end;
    if (reqmot = 'DEW') And (BufferTMotTeller = 3) then //Angle Coude
    begin
      chiffre := -((chiffre div 9)-237);
      CoudePosition := chiffre;
      Ecran.MiseDegreeCoude(chiffre);
    end;
    if (reqmot = 'DEW') And (BufferTMotTeller = 4) then //Angle Poignet
    begin
      PoignetPosition := chiffre;
      Ecran.MiseDegreePoignet(chiffre);
      UdpClientOccuper := False;
    end;
    ecran.MiseLesValeurs(SV,EV,CV,PV,PIV);
  end;
end;
```

```
{Si un des moteurs est venu en butée avec la simulation des Flag Words on
doit
 écrire l'écrire dans le mot input et le mot flag word.}
procedure TUdpClient.SendButee();
var
  SS:String;
begin
  UdpSocket.Proto:='udp';
  UdpSocket.Addr:=Adress;
  UdpSocket.Port:=Port;
  UdpSocket.Connect;
  SS := 'MEW0=' + IntToStr(BinToDig(MEW0));
  UdpSocket.SendStr(SS);
  UdpSocket.Close;
  UdpSocket.Proto:='udp';
  UdpSocket.Addr:=Adress;
  UdpSocket.Port:=Port;
  UdpSocket.Connect;
  SS := 'MMW50=' + IntToStr(BinToDig(MMW50));
  UdpSocket.SendStr(SS);
  UdpSocket.Close;
  UdpClientOccuper := False;
end;

{Calculation d'un numéro binaire en hexadecimal}
function TUdpClient.CalculeMew0EnHex():String;
const
  BinArray: array[0..15, 0..1] of string =
    (('0000', '0'), ('0001', '1'), ('0010', '2'), ('0011', '3'),
     ('0100', '4'), ('0101', '5'), ('0110', '6'), ('0111', '7'),
     ('1000', '8'), ('1001', '9'), ('1010', 'A'), ('1011', 'B'),
     ('1100', 'C'), ('1101', 'D'), ('1110', 'E'), ('1111', 'F'));
var
  Error: Boolean;
  j: Integer;
  BinPart,BinStr: string;
begin
  BinStr := MEW0;
  Result:='';

  Error:=False;
  for j:=1 to Length(BinStr) do
    if not (BinStr[j] in ['0', '1']) then
    begin
      Error:=True;
      ShowMessage('This is not binary number');
      Break;
    end;

  if not Error then
  begin
    case Length(BinStr) mod 4 of
      1: BinStr:='000'+BinStr;
      2: BinStr:='00'+BinStr;
      3: BinStr:='0'+BinStr;
    end;
```

```
    while Length(BinStr)>0 do
    begin
      BinPart:=Copy(BinStr, Length(BinStr)-3, 4);
      Delete(BinStr, Length(BinStr)-3, 4);
      for j:=1 to 16 do
        if BinPart=BinArray[j-1, 0] then
          Result:=BinArray[j-1, 1]+Result;
    end;
  end;
end;

{Calculation d'un numéro binaire en decimal}
function TUdpClient.BinToDig(BinStr:String):Integer;
var
  i,counter: Integer;
begin
  if length(BinStr)>16 then
    raise ERangeError.Create(#13+BinStr+#13+
            'is not within the valid range of a 16 bit binary.'+#13);

  Result:=0;

  for counter:=1 to length(BinStr) do
      if BinStr[Counter]='1' then
        Result:=Result+pow(2,length(BinStr)-counter);

end;

{Elevation à une puissance}
function TUdpClient.pow(base, power: integer): integer;
var counter : integer;
begin
  Result:=1;

  for counter:=1 to power do
    Result:=Result*base;
end;


// ------------ Procedures Publique----------------------------------------
{Commence la communication avec l'IPC sans Robot}
procedure TUdpClient.CommenceComIPCSR();
begin
  Timer.OnTimer := GetFlagWords;
  UdpClientFoB := False;
  Timer.Enabled := True;
end;

{Commence la communication acev l'IPC avec Robot}
procedure TUdpClient.CommenceComIPCAR();
begin
  Timer.OnTimer := GetButees;
  UdpClientFoB := True;
  Timer.Enabled := True;
end;

{Arrêter toutes les communications}
procedure TUdpClient.ArreteComIPC();
begin
  Timer.Enabled := False;
end;
```

```
{Mettre un valeur d'un butée d'un de les moteurs}
procedure TUdpClient.SetButee(RV:String);
begin
  if RV = '11' then //butee socle gauche
    MEW0[3] := '1';
  if RV = '13' then //butee socle droite
    MEW0[1] := '1';
  if RV = '12' then //Socle ne pas butee
  begin
    MEW0[3]:='0';
    MEW0[1]:='0';
  end;
  if RV = '21' then // Epaule haute
    MEW0[4] := '1';
  if RV = '23' then // Epaule basse
    MEW0[5] := '1';
  if RV = '22' then // Epaule ne pas butee
  begin
    MEW0[4] := '0';
    MEW0[5] := '0';
  end;
  if RV = '41' then //butee Poignet gauche
    MMW50[2] := '1';
  if RV = '43' then //butee Poignet droite
    MMW50[1] := '1';
  if RV = '42' then //Poignet ne pas butee
  begin
    MMW50[1] := '0';
    MMW50[2] := '0';
  end;
  if RV = '51' then // butee pince fermée
    MMW50[3] := '1';
  if RV = '53' then
    MMW50[4] := '1';
  if RV = '52' then
  begin
    MMW50[3] := '0';
    MMW50[4] := '0';
  end;
end;

end.
```

Unit1.pas

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, GLWin32Viewer, GLScene, GLObjects, GLMisc, GLTexture, GLFireFX,
  ExtCtrls, RobotClass, EtatClass, RobotViewerClass, WSocket,
  StdCtrls, WSocketS, TcpClass, GLGeomObjects, POSITIONS, UdpClass, Menus,
  frmOptionIpc, frmOptionServeur, frmOptionRobot, sauveVRML;
```

```
type
// ------------ Mise en forme de la classe --------------------------------
  TFrmSimulRobot = class(TForm)

    // -- Les choses pour le GLScene --
    MatLib: TGLMaterialLibrary;
    GLScene1: TGLScene;
    Lum1: TGLLightSource;
    Lum2: TGLLightSource;
    Lum3: TGLLightSource;
    Lum4: TGLLightSource;
    Ecorce: TGLSphere;
    Sol: TGLDisk;
    DC1: TGLDummyCube;
    Pied: TGLDummyCube;
    Bloc01: TGLCube;
    Bloc02: TGLCube;
    Socle: TGLCylinder;
    Bloc11: TGLCube;
    Articul1: TGLCylinder;
    Bras1: TGLCylinder;
    Articul2: TGLCylinder;
    Bras2: TGLCylinder;
    Poignet: TGLCylinder;
    Metacarpe: TGLFrustrum;
    Doigt1: TGLDummyCube;
    Pha11: TGLCube;
    ArtiD1: TGLDummyCube;
    Rot1: TGLCylinder;
    Pha12: TGLCube;
    Doigt2: TGLDummyCube;
    Pha21: TGLCube;
    ArtiD2: TGLDummyCube;
    Rot2: TGLCylinder;
    Pha22: TGLCube;
    CentreParts: TGLDummyCube;
    Cam1: TGLCamera;
    Viewer: TGLSceneViewer;
    SupportViewer: TPanel;
    GLSceneViewer1: TGLSceneViewer;
    EffetFeu: TGLFireFXManager;
    Particules: TGLFireFXManager;

    // -- Les Choses Visuel --
    WSocketServer1: TWSocketServer;
    btnValeurs: TButton;
    btnSimuleIpc: TButton;
    TmrGetValeurs: TTimer;
    MainMenu1: TMainMenu;
    Fichier1: TMenuItem;
    Proprits1: TMenuItem;
    IPC1: TMenuItem;
    Serveur1: TMenuItem;
    btnChargeValeurs: TButton;
    mmoServerMessages: TMemo;
    btnTcpServeur: TButton;
    lblTcpServeurOnOff: TLabel;
    Robot1: TMenuItem;
    lblMontreValeursOnOff: TLabel;
    lblSimuleIpcOnOff: TLabel;
    Fermer1: TMenuItem;
```

```
    // -- Form methodes --
    procedure OnCreate(Sender: TObject);
    procedure OnClose(Sender: TObject; var Action: TCloseAction);
    procedure WSocketServer1ClientConnect(Sender: TObject;
      Client: TWSocketClient; Error: Word);
    procedure WSocketServer1ClientDisconnect(Sender: TObject;
      Client: TWSocketClient; Error: Word);
    procedure WSocketServer1SessionClosed(Sender: TObject; Error: Word);

    procedure btnValeursClick(Sender: TObject);
    procedure btnSimuleIpcClick(Sender: TObject);
    procedure btnChargeValeursClick(Sender: TObject);
    procedure btnTcpServeurClick(Sender: TObject);

    procedure TmrGetValeursTimer(Sender: TObject);
    procedure TmrGetValeursButeeTimer(Sender: TObject);

    procedure IPC1Click(Sender: TObject);
    procedure Serveur1Click(Sender: TObject);
    procedure OnExit(Sender: TObject);
    procedure Robot1Click(Sender: TObject);
    procedure Fermer1Click(Sender: TObject);

  private

    // -- Les Objects --
    RobotViewer : TRobotViewer;
    Robot : TRobot;
    UdpClient : TUdpClient;
    SauveVRML : TSauveVRML;

    // -- Les Ecrans --
    FrmPosEcran : TFrmPosEcran;
    FrmOptionIPC : TFrmOptionIpc;
    FrmOptionServeur : TFrmOptionServeur;
    FrmOptionRobot : TFrmOptionRobot;

    // -- Les Variables --
    IpcPort : String;
    IpcAdresse : String;
    IpcVitesse : String;

    ServeurPort : String;
    ServeurAdresse : String;

    // -- Les Méthodes --
    procedure LireProprieteServeur();
    procedure LireProprieteIpc();

  public
    // -- Les Méthodes --
    procedure ReceiveAction(Sender: TObject; Action: byte);
    procedure MiseOptionIPC(P:string;A:string;V:String);

  end;
// ------------ Fin de la mise en forme ----------------------------------
```

```
var
  FrmSimulRobot: TFrmSimulRobot;

implementation

{$R *.dfm}

// ------------ Form Methodes ---------------------------------------------
procedure TFrmSimulRobot.OnCreate(Sender: TObject);
begin

  {Lire les fichier .ini}
  LireProprieteServeur;
  LireProprieteIpc;

  {Les autres ecrans}
  FrmPosEcran := TFrmPosEcran.Create(Self);
  FrmOptionIPC := TfrmOptionIpc.Create(Self);
  FrmOptionServeur := TFrmOptionServeur.Create(Self);
  FrmOptionRobot := TFrmOptionRobot.Create(Self);

  {Les Object des Classes}
  RobotViewer :=
TRobotViewer.Create(Socle,Articul1,Articul2,Poignet,Doigt1,Doigt2,GLSceneVi
ewer1);
  Robot := TRobot.Create;
  Robot.setRobotViewer(RobotViewer);
  SauveVRML := TSauveVRML.Create;
  UdpClient :=
TUdpClient.Create(Self,IpcPort,IpcAdresse,StrToInt(IpcVitesse),FrmPosEcran)
;

  {SocketServeur}
  WSocketServer1.Port := ServeurPort;
  WSocketServer1.Addr := '0.0.0.0';
  WSocketServer1.ClientClass := TcpClient;
  WSocketServer1.Listen;
end;

procedure TFrmSimulRobot.OnClose(Sender: TObject; var Action:
TCloseAction);
begin
  //WSocketServer1.Destroy;
end;

procedure TFrmSimulRobot.OnExit(Sender: TObject);
begin
  RobotViewer.Free;
  Robot.Destroy;
  Robot.Free;
  UDPClient.Free;
end;
```

```
// ------------ WSocketServer1 Methodes ----------------------------------
{Cette méthode est provoqué quand un client se connecte sur TCP}
procedure TFrmSimulRobot.WSocketServer1ClientConnect(Sender: TObject;
  Client: TWSocketClient; Error: Word);
var
  Iclient: TcpClient;
begin
  Iclient := TcpClient(Client);
  Iclient.OnReceive := ReceiveAction; {S'il y arrive quelque chose}
  Iclient.Initialize;
  mmoServerMessages.Lines.Add('MS:Client en ligne.');
end;

{Cette méthode est provoqué quand un client se déconnecte du serveur}
procedure TFrmSimulRobot.WSocketServer1ClientDisconnect(Sender: TObject;
  Client: TWSocketClient; Error: Word);
var
  Iclient: TcpClient;
begin
  Iclient := TcpClient(Client);
  Iclient.Finalize;
  mmoServerMessages.Lines.Add('MS:Client hors ligne.');
  //Robot.ArreteTout;
end;

procedure TFrmSimulRobot.WSocketServer1SessionClosed(Sender: TObject;
Error: Word);
begin
  Robot.ArreteTout;
end;

// ------------ Timer Methodes -------------------------------------------
procedure TFrmSimulRobot.TmrGetValeursTimer(Sender: TObject);
var
  RV : String;
begin
  UdpClient.SetButee(Robot.Bouche(StrToInt(UdpClient.PFW1)));
  UdpClient.SetButee(Robot.Bouche(StrToInt(UdpClient.PFW2)));
  UdpClient.SetButee(Robot.Bouche(StrToInt(UdpClient.PFW3)));
  UdpClient.SetButee(Robot.Bouche(StrToInt(UdpClient.PFW4)));
  UdpClient.SetButee(Robot.Bouche(StrToInt(UdpClient.PFW5)));
end;

procedure TFrmSimulRobot.TmrGetValeursButeeTimer(Sender: TObject);
begin

Robot.SetButees(UdpClient.PSV,UdpClient.PEV,UdpClient.PCV,UdpClient.PPV,Udp
Client.PPIV);
  Robot.SetPositionCoude(UdpClient.PCoudePosition);
  Robot.SetPositionPoignet(UdpClient.PPoignetPosition);
end;
```

```
// ------------ Button Methodes ------------------------------------------
{Commencer la simulation avec l'IPC. Pour retrouver les valeurs des butées
 et Angles. Ces valeurs sont présenté dans l'écran FrmPosEcran.}
procedure TFrmSimulRobot.btnValeursClick(Sender: TObject);
begin

  if (TmrGetValeurs.Enabled = False) then
  begin
    FrmPosEcran.Visible := true;                      {L'autre écran}
    UdpClient.CommenceComIPCAR;                       {Dire au UdpClient
de}
    TmrGetValeurs.OnTimer := TmrGetValeursButeeTimer; {commencer sa
communication}
    TmrGetValeurs.Enabled := true;
    btnTcpServeur.Enabled := False;
    btnSimuleIpc.Enabled := False;
    lblMontreValeursOnOff.Color := clgreen;
  end
  else
  begin
    FrmPosEcran.Visible := False;
    UdpClient.ArreteComIPC;
    TmrGetValeurs.Enabled := False;
    btnTcpServeur.Enabled := True;
    btnSimuleIpc.Enabled := True;
    lblMontreValeursOnOff.Color := clred;
  end;

end;

{Commencer la simulation avec l'IPC. Pour retrouver les valeurs des Flag
Words
 et avec eux on simula le mouvement du robot}
procedure TFrmSimulRobot.btnSimuleIpcClick(Sender: TObject);
begin
  if (TmrGetValeurs.Enabled = False) then
  begin
    UdpClient.CommenceComIPCSR;
    TmrGetValeurs.OnTimer := TmrGetValeursTimer;
    TmrGetValeurs.Enabled := true;
    btnTcpServeur.Enabled := False;
    btnValeurs.Enabled := False;
    lblSimuleIpcOnOff.Color := clgreen;
  end
  else
  begin
    UdpClient.ArreteComIPC;
    TmrGetValeurs.Enabled := False;
    btnTcpServeur.Enabled := True;
    btnValeurs.Enabled := True;
    lblSimuleIpcOnOff.Color := clred;
  end;
end;
```

```
{Commencer le serveur, ou arrêter le serveur}
procedure TFrmSimulRobot.btnTcpServeurClick(Sender: TObject);
begin
  if (WSocketServer1.State = WSListening) Or (WSocketServer1.State =
WSConnected) then
  begin
    if (WSocketServer1.ClientCount > 0) then
    begin
      if (WSocketServer1.Client[0].State = WSConnected) then
      begin
        WSocketServer1.Client[0].Close;     {S'il y a des Clients
connecter}
      end;                                  {Les déconnectent avant
d'arrêter}
    end;                                    {le serveur}
    WSocketServer1.Close;
    lblTcpServeurOnOff.Color := clred;
    btnValeurs.Enabled := true;
    btnSimuleIpc.Enabled := true;
  end
  else if WSocketServer1.State = WSClosed then
  begin
    btnValeurs.Enabled := false;
    btnSimuleIpc.Enabled := false;
    WSocketServer1.Port := ServeurPort;
    WSocketServer1.Addr := ServeurAdresse;
    WSocketServer1.ClientClass := TcpClient;
    WSocketServer1.Listen;
    lblTcpServeurOnOff.Color := clgreen;
  end;

end;

{Si un des fichier avec les propiétés du serveur, de l'IPC ou du robot, est
 modifier. On peut avec ce bouton-ci charger ces valeurs.}
procedure TFrmSimulRobot.btnChargeValeursClick(Sender: TObject);
begin
  //Lire les fichier .ini
  LireProprieteServeur;
  LireProprieteIpc;

  UDPClient.Destroy;

  //Les Object des Classes
  RobotViewer :=
TRobotViewer.Create(Socle,Articul1,Articul2,Poignet,Doigt1,Doigt2,GLSceneVi
ewer1);
  Robot := TRobot.Create;
  Robot.setRobotViewer(RobotViewer);
  UdpClient :=
TUdpClient.Create(Self,IpcPort,IpcAdresse,StrToInt(IpcVitesse),FrmPosEcran)
;

  if (WSocketServer1.State = WSListening) then
    WSocketServer1.Close;

  //SocketServeur
  WSocketServer1.Port := ServeurPort;
  WSocketServer1.Addr := ServeurAdresse;
  WSocketServer1.ClientClass := TcpClient;
  WSocketServer1.Listen;
```

```
end;

// ------------ Menu Methodes ------------------------------------------
procedure TFrmSimulRobot.Fermer1Click(Sender: TObject);
begin
  Close;
end;

procedure TFrmSimulRobot.IPC1Click(Sender: TObject);
begin
  FrmOptionIPC.Visible := true;
end;

procedure TFrmSimulRobot.Serveur1Click(Sender: TObject);
begin
  FrmOptionServeur.Visible := true;
end;

procedure TFrmSimulRobot.Robot1Click(Sender: TObject);
begin
  FrmOptionRobot.Visible := true;
end;

// ------------ Public Procedures --------------------------------------
procedure TFrmSimulRobot.ReceiveAction(Sender: TObject; Action: byte);
var
  RV : String;
begin
  if Action <> 1 then   //Kan Foutmelding op terecht komen
  begin
    RV := Robot.Bouche(Action);
  end;
  mmoServerMessages.Lines.Add('MS:Recieved');
end;

procedure TFrmSimulRobot.MiseOptionIPC(P:string;A:string;V:String);
begin
  IpcPort := P;
  IpcAdresse := A;
  IpcVitesse := V;
end;

// ------------ Procedures Privé ---------------------------------------
procedure TFrmSimulRobot.LireProprieteServeur();
var
  iFileHandle:Integer;
  iFileLength:Integer;
  iBytesRead:Integer;
  Buffer:PChar;
  i,place:Integer;
  KindValue,Value:String;
begin
  try

    place := 1; KindValue := ''; Value := '';

    iFileHandle :=FileOpen('PropServ.ini',fmOpenRead);

    iFileLength :=FileSeek(iFileHandle,0,2);
    FileSeek(iFileHandle,0,0);
```

```
    Buffer :=PChar(AllocMem(iFileLength +1));
    iBytesRead :=FileRead(iFileHandle,Buffer^,iFileLength);

    FileClose(iFileHandle);

    for i := 0 to iBytesRead-1 do
    begin
      if ((Buffer[i] <> '=') And (Buffer[i] <> ';') And (place = 1)) then
      begin
        KindValue := KindValue + Buffer[i];
      end;
      if (Buffer[i] = '=') then
      begin
        place := 2;
      end;
      if ((Buffer[i] <> '=') And (Buffer[i] <> ';') And (place = 2)) then
      begin
        Value := Value + Buffer[i];
      end;
      if (Buffer[i] = ';') then
      begin
        place := 3;

        if (KindValue = 'ip') then
          ServeurAdresse := Value;
        if (KindValue = 'port') then
          ServeurPort := Value;

        KindValue := '';
        Value := '';
      end;
      if (Buffer[i] = #$A) then
        place := 1;

    end;
  finally
    FreeMem(Buffer);
  end;
end;

procedure TFrmSimulRobot.LireProprieteIpc();
var
  iFileHandle:Integer;
  iFileLength:Integer;
  iBytesRead:Integer;
  Buffer:PChar;
  i,place:Integer;
  KindValue,Value:String;
begin
  try

    place := 1; KindValue := ''; Value := '';

    iFileHandle :=FileOpen('PropIpc.ini',fmOpenRead);

    iFileLength :=FileSeek(iFileHandle,0,2);
    FileSeek(iFileHandle,0,0);

    Buffer :=PChar(AllocMem(iFileLength +1));
    iBytesRead :=FileRead(iFileHandle,Buffer^,iFileLength);
```

```
    FileClose(iFileHandle);

    for i := 0 to iBytesRead-1 do
    begin
      if ((Buffer[i] <> '=') And (Buffer[i] <> ';') And (place = 1)) then
      begin
        KindValue := KindValue + Buffer[i];
      end;
      if (Buffer[i] = '=') then
      begin
        place := 2;
      end;
      if ((Buffer[i] <> '=') And (Buffer[i] <> ';') And (place = 2)) then
      begin
        Value := Value + Buffer[i];
      end;
      if (Buffer[i] = ';') then
      begin
        place := 3;

        if (KindValue = 'ip') then
          IpcAdresse := Value;
        if (KindValue = 'port') then
          IpcPort := Value;
        if (KindValue = 'Vitesse') then
          IpcVitesse := Value;

        KindValue := '';
        Value := '';
      end;
      if (Buffer[i] = #$A) then
        place := 1;

    end;
  finally
    FreeMem(Buffer);
  end;

end;

end.
```

frmOptionIpc.pas

```
unit frmOptionIpc;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls;

type

// ------------ Mise en forme de la classe -------------------------------
  TfrmOptionIpc = class(TForm)

    // -- Les Choses Visuel --
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    txtPortIpc: TEdit;
    txtAdresseIpc: TEdit;
    txtVitesseIpc: TEdit;
    cmdSauvegarder: TButton;
    cmdAnnuler: TButton;
    // -- Form méthodes --
    procedure OnCreate(Sender: TObject);
    procedure cmdAnnulerClick(Sender: TObject);
    procedure cmdSauvegarderClick(Sender: TObject);

  private

    // -- Les Variables --
    // -- Les Methodes --
    procedure metValeursDansTextbox();

  public

  end;
// ------------ Fin de la mise en forme ----------------------------------

implementation

{$R *.dfm}

// ------------ Procedures Privé-------------------------------------------
procedure TfrmOptionIpc.metValeursDansTextbox();
var
  iFileHandle:Integer;
  iFileLength:Integer;
  iBytesRead:Integer;
  Buffer:PChar;
  i,place:Integer;
  KindValue,Value:String;
begin
  try

    place := 1; KindValue := ''; Value := '';

    iFileHandle :=FileOpen('PropIpc.ini',fmOpenRead);
```

```
    iFileLength :=FileSeek(iFileHandle,0,2);
    FileSeek(iFileHandle,0,0);

    Buffer :=PChar(AllocMem(iFileLength +1));
    iBytesRead :=FileRead(iFileHandle,Buffer^,iFileLength);

    FileClose(iFileHandle);

    for i := 0 to iBytesRead-1 do
    begin
      if ((Buffer[i] <> '=') And (Buffer[i] <> ';') And (place = 1)) then
      begin
        KindValue := KindValue + Buffer[i];
      end;
      if (Buffer[i] = '=') then
      begin
        place := 2;
      end;
      if ((Buffer[i] <> '=') And (Buffer[i] <> ';') And (place = 2)) then
      begin
        Value := Value + Buffer[i];
      end;
      if (Buffer[i] = ';') then
      begin
        place := 3;

        if (KindValue = 'ip') then
          txtAdresseIpc.Text := Value;
        if (KindValue = 'port') then
          txtPortIpc.Text := Value;
        if (KindValue = 'Vitesse') then
          txtVitesseIpc.Text := Value;

        KindValue := '';
        Value := '';
      end;
      if (Buffer[i] = #$A) then
        place := 1;

    end;
  finally
    FreeMem(Buffer,iBytesRead);
  end;
end;

// ------------ Procedures Publique---------------------------------------
procedure TfrmOptionIpc.OnCreate(Sender: TObject);
begin
  metValeursDansTextBox;
end;

procedure TfrmOptionIpc.cmdAnnulerClick(Sender: TObject);
begin
  MetValeursDansTextBox;
  Self.Visible := False;
end;

procedure TfrmOptionIpc.cmdSauvegarderClick(Sender: TObject);
var
  Fichier:TextFile;
  testtekst:TextFile;
```

```
begin
  assignfile(Fichier,'PropIpc.ini');
  try
    Rewrite(Fichier);
    writeln(Fichier,'port=' + txtPortIpc.Text + ';');
    writeln(Fichier,'ip=' + txtAdresseIpc.Text + ';');
    writeln(Fichier,'Vitesse=' + txtVitesseIpc.Text + ';');
  finally
    CloseFile(Fichier);
  end;
  Self.Visible := False;
end;
end.
```

frmOptionRobot.pas

```
unit frmOptionRobot;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls;

// ------------ Mise en forme de la classe -------------------------------
type
  TfrmOptionRobot = class(TForm)

    // -- Les Choses Visuelles --
    gb_socle: TGroupBox;
    lbl_socle_vitesse: TLabel;
    lbl_socle_amax: TLabel;
    lbl_socle_amin: TLabel;
    lbl_socle_position: TLabel;
    edit_socle_vitesse: TEdit;
    edit_socle_amin: TEdit;
    edit_socle_position: TEdit;
    gb_epaule: TGroupBox;
    lbl_epaule_vitesse: TLabel;
    lbl_epaule_amax: TLabel;
    lbl_epaule_amin: TLabel;
    lbl_epaule_position: TLabel;
    edit_epaule_vitesse: TEdit;
    edit_epaule_amax: TEdit;
    edit_epaule_amin: TEdit;
    gb_coude: TGroupBox;
    gb_poignet: TGroupBox;
    gb_pince: TGroupBox;
    btnSauvegarder: TButton;
    btnAnnuler: TButton;
    lbl_coude_vitesse: TLabel;
    lbl_coude_amax: TLabel;
    lbl_coude_amin: TLabel;
    lbl_coude_position: TLabel;
    edit_coude_vitesse: TEdit;
    edit_coude_amax: TEdit;
    edit_coude_amin: TEdit;
    edit_coude_position: TEdit;
    lbl_poignet_vitesse: TLabel;
```

```
    lbl_poignet_amax: TLabel;
    lbl_poinget_amin: TLabel;
    lbl_poignet_position: TLabel;
    edit_poignet_vitesse: TEdit;
    edit_poignet_amax: TEdit;
    edit_poignet_amin: TEdit;
    edit_poignet_position: TEdit;
    lbl_pince_vitesse: TLabel;
    lbl_pince_amax: TLabel;
    lbl_pince_amin: TLabel;
    lbl_pince_position: TLabel;
    edit_pince_vitesse: TEdit;
    edit_pince_amax: TEdit;
    edit_pince_amin: TEdit;
    edit_pince_position: TEdit;
    edit_socle_amax: TEdit;
    edit_epaule_position: TEdit;

    // -- Form méthodes --
    procedure FormCreate(Sender: TObject);
    procedure OnClick(Sender: TObject);
    procedure btnAnnulerClick(Sender: TObject);

  private
    // -- Les Méthodes --
    procedure remplir_valeurs();
  public

  end;
// ------------ Fin de la mise en forme ----------------------------------

implementation

{$R *.dfm}
// ------------ Procedures Privé ------------------------------------------
procedure TfrmOptionRobot.remplir_valeurs();
  var
    vitesse, anglemax, anglemin, position : String;
    nom : string;

    iFileHandle, iFileLength, iBytesRead:Integer;
    Buffer:PChar;
    i,place:Integer;
    KindValue,Value:String;
  begin
  try
    place := 1; KindValue := ''; Value := '';

    iFileHandle :=FileOpen('PropRobot.ini',fmOpenRead);

    iFileLength :=FileSeek(iFileHandle,0,2);
    FileSeek(iFileHandle,0,0);

    Buffer :=PChar(AllocMem(iFileLength +1));
    iBytesRead :=FileRead(iFileHandle,Buffer^,iFileLength);

    FileClose(iFileHandle);

    for i := 0 to iBytesRead-1 do
    begin
      if ((Buffer[i] <> '=') And (Buffer[i] <> ';') And (place = 1)) then
```

```
begin
  KindValue := KindValue + Buffer[i];
end;
if (Buffer[i] = '=') then
begin
  place := 2;
end;
if ((Buffer[i] <> '=') And (Buffer[i] <> ';') And (place = 2)) then
begin
  Value := Value + Buffer[i];
end;
if (Buffer[i] = ';') then
begin
  place := 3;

  if (KindValue = 'nom') then
    nom := Value;
  if (KindValue = 'vitesse') then
    vitesse := Value;
  if (KindValue = 'angleMax') then
    anglemax := Value;
  if (KindValue = 'angleMin') then
    anglemin := Value;
  if (KindValue = 'position') then
  begin
    position := Value;
    case nom[1] of
      'S'{ocle'} :  begin
                      edit_socle_vitesse.text := vitesse;
                      edit_socle_amax.text := anglemax;
                      edit_socle_amin.text := anglemin;
                      edit_socle_position.text := position;
                    end;
      'E'{paule'} : begin
                      edit_epaule_vitesse.text := vitesse;
                      edit_epaule_amax.text := anglemax;
                      edit_epaule_amin.text := anglemin;
                      edit_epaule_position.text := position;
                    end;
      'C'{oude'} :  begin
                      edit_coude_vitesse.text := vitesse;
                      edit_coude_amax.text := anglemax;
                      edit_coude_amin.text := anglemin;
                      edit_coude_position.text := position;
                    end;
      'P'{oignet'} : begin
                      edit_poignet_vitesse.text := vitesse;
                      edit_poignet_amax.text := anglemax;
                      edit_poignet_amin.text := anglemin;
                      edit_poignet_position.text := position;
                    end;
      'N'{pince'} : begin
                      edit_pince_vitesse.text := vitesse;
                      edit_pince_amax.text := anglemax;
                      edit_pince_amin.text := anglemin;
                      edit_pince_position.text := position;
                    end;
    end;
  end;

  KindValue := '';
```

```
        Value := '';
      end;
      if (Buffer[i] = #$A) then
        place := 1;

    end;
  finally
    FreeMem(Buffer);
  end;
end;

// ------------ Procedures Publique ---------------------------------------
procedure TfrmOptionRobot.FormCreate(Sender: TObject);
begin
  remplir_valeurs();
end;

procedure TfrmOptionRobot.OnClick(Sender: TObject);
var
  Fichier:TextFile;
  testtekst:TextFile;
begin
  assignfile(Fichier,'PropRobot.ini');
  try
    Rewrite(Fichier);
    //Socle
    writeln(Fichier,'nom=Socle;');
    writeln(Fichier,'vitesse=' + edit_socle_vitesse.Text + ';');
    writeln(Fichier,'angleMax=' + edit_socle_amax.Text + ';');
    writeln(Fichier,'angleMin=' + edit_socle_amin.Text + ';');
    writeln(Fichier,'position=' + edit_socle_position.Text + ';');
    //Epaule
    writeln(Fichier,'nom=Epaule;');
    writeln(Fichier,'vitesse=' + Edit_epaule_vitesse.Text + ';');
    writeln(Fichier,'angleMax=' + Edit_epaule_amax.Text + ';');
    writeln(Fichier,'angleMin=' + Edit_epaule_amin.Text + ';');
    writeln(Fichier,'position=' + Edit_epaule_position.Text + ';');
    //Coude
    writeln(Fichier,'nom=Coude;');
    writeln(Fichier,'vitesse=' + Edit_coude_vitesse.Text + ';');
    writeln(Fichier,'angleMax=' + Edit_coude_amax.Text + ';');
    writeln(Fichier,'angleMin=' + Edit_coude_amin.Text + ';');
    writeln(Fichier,'position=' + Edit_coude_position.Text + ';');
    //Poignet
    writeln(Fichier,'nom=Poignet;');
    writeln(Fichier,'vitesse=' + Edit_Poignet_vitesse.Text + ';');
    writeln(Fichier,'angleMax=' + Edit_Poignet_amax.Text + ';');
    writeln(Fichier,'angleMin=' + Edit_Poignet_amin.Text + ';');
    writeln(Fichier,'position=' + Edit_Poignet_position.Text + ';');
    //Pince
    writeln(Fichier,'nom=NPince;');
    writeln(Fichier,'vitesse=' + Edit_pince_vitesse.Text + ';');
    writeln(Fichier,'angleMax=' + Edit_pince_amax.Text + ';');
    writeln(Fichier,'angleMin=' + Edit_pince_amin.Text + ';');
    writeln(Fichier,'position=' + Edit_pince_position.Text + ';');

  finally
    CloseFile(Fichier);
  end;
  Self.Visible := False;
end;
```

```
procedure TfrmOptionRobot.btnAnnulerClick(Sender: TObject);
begin
  remplir_valeurs();
  self.Visible := False;
end;

end.
```

frmOptionServeur.pas

```
unit frmOptionServeur;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls;

type
// ------------ Mise en forme de la classe --------------------------------
  TfrmOptionServeur = class(TForm)

    // -- Les Choses Visuel --
    Label1: TLabel;
    Label2: TLabel;
    txtPortServeur: TEdit;
    txtAdresseServeur: TEdit;
    btnSauvegarder: TButton;
    btnAnnuler: TButton;
    procedure OnCreate(Sender: TObject);
    procedure OnClick(Sender: TObject);
    procedure btnSauvegarderClick(Sender: TObject);

    // -- Form methodes --
  private

    // -- Les Variables --
    // -- Les Methodes --
    procedure metValeursDansTextBox();
  public

  end;
// ------------ Fin de la mise en forme ----------------------------------

implementation

{$R *.dfm}

// ------------ Procedures Privé ------------------------------------------
procedure TFrmOptionServeur.metValeursDansTextBox();
var
  iFileHandle:Integer;
  iFileLength:Integer;
  iBytesRead:Integer;
  Buffer:PChar;
  i,place:Integer;
  KindValue,Value:String;
begin
```

```
  try

    place := 1; KindValue := ''; Value := '';

    iFileHandle :=FileOpen('PropServ.ini',fmOpenRead);

    iFileLength :=FileSeek(iFileHandle,0,2);
    FileSeek(iFileHandle,0,0);

    Buffer :=PChar(AllocMem(iFileLength +1));
    iBytesRead :=FileRead(iFileHandle,Buffer^,iFileLength);

    FileClose(iFileHandle);

    for i := 0 to iBytesRead-1 do
    begin
      if ((Buffer[i] <> '=') And (Buffer[i] <> ';') And (place = 1)) then
      begin
        KindValue := KindValue + Buffer[i];
      end;
      if (Buffer[i] = '=') then
      begin
        place := 2;
      end;
      if ((Buffer[i] <> '=') And (Buffer[i] <> ';') And (place = 2)) then
      begin
        Value := Value + Buffer[i];
      end;
      if (Buffer[i] = ';') then
      begin
        place := 3;

        if (KindValue = 'ip') then
          txtAdresseServeur.Text := Value;
        if (KindValue = 'port') then
          txtPortServeur.Text := Value;

        KindValue := '';
        Value := '';
      end;
      if (Buffer[i] = #$A) then
        place := 1;

    end;
  finally
    FreeMem(Buffer);
  end;
end;

// ------------ Procedures Publique ---------------------------------------
procedure TfrmOptionServeur.OnCreate(Sender: TObject);
begin
  metValeursDansTextBox;
end;

procedure TfrmOptionServeur.OnClick(Sender: TObject);
begin
  metValeursDansTextBox;
  Self.Visible := false;
end;
```

```
procedure TfrmOptionServeur.btnSauvegarderClick(Sender: TObject);
var
  Fichier:TextFile;
  testtekst:TextFile;
begin
  assignfile(Fichier,'PropServ.ini');
  try
    Rewrite(Fichier);
    writeln(Fichier,'ip=' + txtAdresseServeur.Text + ';');
    writeln(Fichier,'port=' + txtPortServeur.Text + ';');
  finally
    CloseFile(Fichier);
  end;
  Self.Visible := False;
end;

end.
```

Positions.pas

```
unit POSITIONS;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls;

type

// ------------ Mise en forme de la classe --------------------------------
  TFrmPosEcran = class(TForm)

    // -- Les Choses Visuelles --
    box_positions: TGroupBox;
    box_degree: TGroupBox;
    box_on_off: TGroupBox;
    box_moteur: TGroupBox;
    lbl_epaule: TLabel;
    lbl_socle: TLabel;
    lbl_coude: TLabel;
    lbl_poignet: TLabel;
    lbl_pince: TLabel;
    lbl_socle_degree: TLabel;
    lbl_epaule_degree: TLabel;
    lbl_coude_degree: TLabel;
    lbl_poignet_degree: TLabel;
    lbl_pince_degree: TLabel;
    lbl_vgauche_socle: TLabel;
    lbl_vbas_epaule: TLabel;
    lbl_vbas_coude: TLabel;
    lbl_vgauche_poignet: TLabel;
    lbl_vfermee_pince: TLabel;
    lbl_socle_gauche: TLabel;
    lbl_epaule_bas: TLabel;
    lbl_coude_bas: TLabel;
    lbl_poignet_gauche: TLabel;
    lbl_pince_ferme: TLabel;
    lbl_vdroite_socle: TLabel;
    lbl_vhaut_epaule: TLabel;
```

```
    lbl_vhaut_coude: TLabel;
    lbl_vdroite_poignet: TLabel;
    lbl_vouvert_pince: TLabel;
    lbl_socle_droite: TLabel;
    lbl_epaule_haut: TLabel;
    lbl_coude_haut: TLabel;
    lbl_poignet_droite: TLabel;
    lbl_pince_ouvert: TLabel;
  private



  public
    // -- Les Méthodes --
    procedure
MiseLesValeurs(SV:integer;EV:integer;CV:integer;PV:integer;PIV:integer);
    procedure MiseDegreeCoude(Degree:integer);
    procedure MiseDegreePoignet(Degree:integer);

  end;

{var
  Form1: TFrmPosEcran;}

implementation

// ------------ Procedures Publique ---------------------------------------
procedure
TFrmPosEcran.MiseLesValeurs(SV:integer;EV:integer;CV:integer;PV:integer;PIV
:integer);
begin

  case SV of
    1:begin lbl_vgauche_socle.Color := clRed; lbl_vdroite_socle.Color :=
clgreen; end;
    2:begin lbl_vgauche_socle.Color := clgreen; lbl_vdroite_socle.Color :=
clgreen; end;
    3:begin lbl_vgauche_socle.Color := clgreen; lbl_vdroite_socle.Color :=
clRed; end;
  end;

  case EV of
    1:begin lbl_vbas_epaule.Color := clred; lbl_vhaut_epaule.Color :=
clgreen; end;
    2:begin lbl_vbas_epaule.Color := clgreen; lbl_vhaut_epaule.Color :=
clgreen; end;
    3:begin lbl_vbas_epaule.Color := clgreen; lbl_vhaut_epaule.Color :=
clred; end;
  end;

  case CV of
    1:begin lbl_vbas_coude.Color := clred; lbl_vhaut_coude.Color :=
clgreen; end;
    2:begin lbl_vbas_coude.Color := clgreen; lbl_vhaut_coude.Color :=
clgreen; end;
    3:begin lbl_vbas_coude.Color := clgreen; lbl_vhaut_coude.Color :=
clred; end;
  end;
```

```
  case PV of
    1:begin lbl_vgauche_poignet.Color := clred; lbl_vdroite_poignet.Color
:= clgreen; end;
    2:begin lbl_vgauche_poignet.Color := clgreen; lbl_vdroite_poignet.Color
:= clgreen; end;
    3:begin lbl_vgauche_poignet.Color := clgreen; lbl_vdroite_poignet.Color
:= clred; end;
  end;

  case PIV of
    1:begin lbl_vfermee_pince.Color := clred; lbl_vouvert_pince.Color :=
clgreen; end;
    2:begin lbl_vfermee_pince.Color := clgreen; lbl_vouvert_pince.Color :=
clgreen; end;
    3:begin lbl_vfermee_pince.Color := clgreen; lbl_vouvert_pince.Color :=
clred; end;
  end;

end;

procedure TFrmPosEcran.MiseDegreeCoude(Degree:integer);
begin
  lbl_coude_degree.Caption := IntToStr(Degree);
end;

procedure TFrmPosEcran.MiseDegreePoignet(Degree:integer);
begin
  lbl_poignet_degree.Caption := IntToStr(Degree);
end;

{$R *.dfm}

end.
```

SauveVRML.pas

```
unit sauveVRML;

interface

uses  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
      StdCtrls, GLScene, GLObjects, ComCtrls, GLTree{, unitsauv,
ObjetsSup}, GLGeomObjects;

type

//---------------------------------------------------------------------------
// Le class TSauveVRML
//---------------------------------------------------------------------------
TSauveVRML = class

  constructor Create;
  Destructor Destroy;

private

  Nom:String;
  Fichier:TextFile;
  GLScene:TGLScene;
```

```
  constante:integer;


  precision:integer;
  precision1a:integer;
  precision1b:integer;
  aenfant:integer;


  Function RempVirg (s: string) : string;
  Procedure CubeVRML (cube: TGLCube);
  Procedure CylindreVRML (cyl: TGLCylinder);
  Procedure ConeVRML (cone: TGLCone);
  Procedure SphereVRML (sphere: TGLSphere);
  {Procedure PyramideVRML (Pyramide: TGLPyramide);}
  {Procedure ToitVRML (Toit: TGLToit);}
  {Procedure DemiSphereVRML (Demi: TGLDemiSphere);}
  Procedure TorusVRML (Torus: TGLTorus);
  Procedure CameraVRML (camera: TGLCamera);
  Procedure LightVRML (light: TGLLightSource);
  procedure analyseTGL (ob : TGLBaseSceneObject);
  Procedure ImplementeObjets(GLScene:TGLScene);

public

  Procedure SauvegardeVRML(N:String;G:TGLScene);

end;
//---------------------------------------------------------------------------


implementation

constructor TSauveVRML.Create();
begin

  precision := 24;
  precision1a := 10;
  precision1b := 20;
  aenfant := 0;

end;

Destructor TSauveVRML.Destroy();
begin
end;

function TSauveVRML.RempVirg(s:String):String;
var
   I:Integer;
Begin
     For I:=1 to length(s) do
     Begin
        If s[I]=',' then
           S[I]:='.';
     end;
     RempVirg:=S;
end;
```

```
procedure TSauveVRML.CubeVRML(cube: TGLCube);
begin

     writeln(Fichier,'      Transform {');
     writeln(Fichier,'          translation  '+
RempVirg(FloatToStr(Cube.Position.X))+' '+
RempVirg(FloatToStr(Cube.Position.Y))+' '+
RempVirg(FloatToStr(Cube.Position.Z)));
     writeln(Fichier,'          children [');
     writeln(Fichier,'            Shape {');
     writeln(Fichier,'              appearance       Appearance');
     writeln(Fichier,'              {');
     writeln(Fichier,'                    material    Material');
     writeln(Fichier,'                      {');
     writeln(Fichier,'                      }');
     writeln(Fichier,'              }');
     writeln(Fichier,'               geometry       Box {');
     writeln(Fichier,'                 size ' +
RempVirg(FloatToStr(Cube.CubeWidth)) + ' ' +
RempVirg(FloatToStr(Cube.CubeHeight)) + ' ' +
RempVirg(FloatToStr(Cube.CubeDepth)));
     writeln(Fichier,'               }');
     writeln(Fichier,'            }');
     aenfant := aenfant + 1;

end;

Procedure TSauveVRML.CylindreVRML(cyl: TGLCylinder);
Begin
     writeln(Fichier,'      Transform {');
     writeln(Fichier,'          translation  '+
RempVirg(FloatToStr(cyl.Position.X))+' '+
RempVirg(FloatToStr(cyl.Position.Y))+' '+
RempVirg(FloatToStr(cyl.Position.Z)));
     writeln(Fichier,'          children [');
     writeln(Fichier,'            Shape {');
     writeln(Fichier,'              appearance       Appearance');
     writeln(Fichier,'              {');
     writeln(Fichier,'                    material    Material');
     writeln(Fichier,'                      {');
     writeln(Fichier,'                      }');
     writeln(Fichier,'              }');
     writeln(Fichier,'               geometry       Cylinder {');
     writeln(Fichier,'                 height '+
RempVirg(FloatToStr(Cyl.Height)));
     writeln(Fichier,'                 radius '+
RempVirg(FloatToStr(Cyl.TopRadius)));
     writeln(Fichier,'               }');
     writeln(Fichier,'            }');
     aenfant := aenfant + 1;

End;

Procedure TSauveVRML.ConeVRML(cone: TGLCone);
Begin

End;
```

```
Procedure TSauveVRML.SphereVRML(Sphere: TGLSphere);
Begin

End;

Procedure TSauveVRML.TorusVRML(Torus: TGLTorus);
Begin

End;

Procedure TSauveVRML.CameraVRML(camera: TGLCamera);
Begin
     writeln(Fichier,'PerspectiveCamera {');
     writeln(Fichier,'  position '+ RempVirg(FloatToStr(Camera.Position.X
{.X})) + ' '
                                              +
RempVirg(FloatToStr(Camera.Position.Y {.Y})) + ' '
                                              +
RempVirg(FloatToStr(Camera.Position.Z {.Z})));
     writeln(Fichier,'    orientation 0 0 0 0');
     writeln(Fichier,'}');
End;

Procedure TSauveVRML.LightVRML(light: TGLLightSource);
Begin
     writeln(Fichier,'PointLight {');
     writeln(Fichier,'  on  TRUE');
     writeln(Fichier,'  intensity ' + RempVirg(FloatToStr(1-
Light.LinearAttenuation)));
     writeln(Fichier,'  color ' + RempVirg(FloatToStr(Light.Diffuse.Red)) +
' '
                                             +
RempVirg(FloatToStr(Light.Diffuse.Green)) + ' '
                                             +
RempVirg(FloatToStr(Light.Diffuse.Blue)));
     writeln(Fichier,'  location ' + RempVirg(FloatToStr(Light.Position.X
{.X})) + ' '
                                               +
RempVirg(FloatToStr(Light.Position.Y {.Y})) + ' '
                                               +
RempVirg(FloatToStr(Light.Position.Z {.Z})));
     writeln(Fichier,'}');
End;

Procedure TSauveVRML.ImplementeObjets(GLScene:TGLScene);
Var
   i,j,k:Integer;
   ExPyramide,ExToit,ExTorus,ExDemiSphere:Boolean;
   {Demi:TDemiSphere;}
Begin
  ExPyramide:=False;
  ExTorus:=False;
  ExToit:=False;
  ExDemiSphere:=False;
  For i:=0 to (GLScene.Objects.Count-1) do
   begin
    {If GLScene.Objects.Children[i] is TPyramide then
       ExPyramide:=True;}
    {If GLScene.Objects.Children[i] is TToit then
       ExToit:=True;}
    {If GLScene.Objects.Children[i] is TDemiSphere then
```

```
      ExDemiSphere:=True;}
    If GLScene.Objects.Children[i] is TGLTorus then
      ExTorus:=True;
  end;
If ExPyramide=true then
   Begin
     writeln(Fichier,'  Material {');
     writeln(Fichier,'     transparency 1');
     writeln(Fichier,'  }');
     writeln(Fichier,'  DEF Pyramide Group {');
     writeln(Fichier,'      Coordinate3 {point[');
     writeln(Fichier,'          0 0 1,');
     writeln(Fichier,'          1 0 1,');
     writeln(Fichier,'          1 0 0,');
     writeln(Fichier,'          0 0 0,');
     writeln(Fichier,'          0.5 1 0.5]');
     writeln(Fichier,'      } #8 Vertices of the pyramide');
     writeln(Fichier,'      IndexedFaceSet { coordIndex [');
     writeln(Fichier,'          0,1,4,-1]}');
     writeln(Fichier,'      IndexedFaceSet { coordIndex [');
     writeln(Fichier,'          0,1,2,3,-1]}');
     writeln(Fichier,'      IndexedFaceSet { coordIndex [');
     writeln(Fichier,'          1,2,4,-1]}');
     writeln(Fichier,'      IndexedFaceSet { coordIndex [');
     writeln(Fichier,'          2,4,3,-1]}');
     writeln(Fichier,'      IndexedFaceSet { coordIndex [');
     writeln(Fichier,'          3,0,4,-1]}');
     writeln(Fichier,'      }');
     writeln(Fichier,'  }');
   End;
   If ExToit=true then
   Begin
     writeln(Fichier,'  Material {');
     writeln(Fichier,'     transparency 1');
     writeln(Fichier,'  }');
     writeln(Fichier,'  DEF Toit Group {');
     writeln(Fichier,'      Coordinate3 {point[');
     writeln(Fichier,'          0 0 1,');
     writeln(Fichier,'          1 0 1,');
     writeln(Fichier,'          1 0 0,');
     writeln(Fichier,'          0 0 0,');
     writeln(Fichier,'          0.25 1 0.5,');
     writeln(Fichier,'          0.75 1 0.5]');
     writeln(Fichier,'      } #8 Vertices of the toit');
     writeln(Fichier,'      IndexedFaceSet { coordIndex [');
     writeln(Fichier,'          0,1,5,4,-1]}');
     writeln(Fichier,'      IndexedFaceSet { coordIndex [');
     writeln(Fichier,'          0,1,2,3,-1]}');
     writeln(Fichier,'      IndexedFaceSet { coordIndex [');
     writeln(Fichier,'          1,2,5,-1]}');
     writeln(Fichier,'      IndexedFaceSet { coordIndex [');
     writeln(Fichier,'          2,5,4,3,-1]}');
     writeln(Fichier,'      IndexedFaceSet { coordIndex [');
     writeln(Fichier,'          3,0,4,-1]}');
     writeln(Fichier,'      }');
     writeln(Fichier,'  }');
   End;
```

```
  If ExDemiSphere=true then
     Begin
       writeln(Fichier,'  Material {');
       writeln(Fichier,'     transparency 1');
       writeln(Fichier,'  }');
       writeln(Fichier,'  DEF DemiSphere Group {');
       writeln(Fichier,'      Coordinate3 {point[');

       for j:=1 to Round(precision/4) do
         for i:=1 to (precision+1) do
           writeln(Fichier,'          '+RempVirg(FloatToStr(0.5*cos((j-
1)/precision*2*pi)*cos((i-1)/precision*2*pi)+0.5))
                                                  +'
'+RempVirg(FloatToStr(0.5*sin((j-1)/precision*2*pi)))
                                                  +'
'+RempVirg(FloatToStr(0.5*cos((j-1)/precision*2*pi)*sin((i-
1)/precision*2*pi)+0.5))+',');

       writeln(Fichier,'      ]} #8 Vertices of the DemiSphere');

       writeln(Fichier,'      IndexedFaceSet { coordIndex [');
       for i:=0 to (precision) do
          writeln(Fichier,'              '+RempVirg(FloatToStr(i))+', ');
          writeln(Fichier,'              '+RempVirg(FloatToStr(precision-
1))+', -1]}');

       constante:=1;
       for i:=0 to Round((precision+1)*precision/4-precision-2) do
         begin
            if i<>(constante*precision+constante-1) then
                begin
                    writeln(Fichier,'      IndexedFaceSet { coordIndex [');
                    writeln(Fichier,'
'+RempVirg(FloatToStr(i))+', '

+RempVirg(FloatToStr(i+precision+1))+', '

+RempVirg(FloatToStr(i+precision+2))+', '

+RempVirg(FloatToStr(i+1))+', -1]}');
                end
            else inc(constante);
          end;


       writeln(Fichier,'      IndexedFaceSet { coordIndex [');
       writeln(Fichier,'              ');
       for j:=Round((precision+1)*precision/4-precision) to
Round((precision+1)*precision/4-2) do
          writeln(Fichier,'              '+RempVirg(FloatToStr(j))+', ');
          writeln(Fichier,RempVirg(FloatToStr(j+1))+', -1]}');


       writeln(Fichier,'      }');
       writeln(Fichier,'  }');
     End;


   If exTorus=true then
     Begin
       writeln(Fichier,'  Material {');
```

```
       writeln(Fichier,'     transparency 1');
       writeln(Fichier,'  }');
       writeln(Fichier,'  DEF Torus Group {');
       writeln(Fichier,'      Coordinate3 {point[');

       for j:=0 to Round(precision1a) do
         for i:=0 to (precision1b) do
           writeln(Fichier,'
'+RempVirg(FloatToStr(0.5+(0.4+0.1*cos(j/precision1a*2*pi))*cos(i/precision
1b*2*pi)))
                                                    +'
'+RempVirg(FloatToStr(0.5+(0.4+0.1*cos((precision1a-
j)/precision1a*2*pi))*sin(i/precision1b*2*pi)))
                                                    +'
'+RempVirg(FloatToStr(0.1*sin(j/precision1a*2*pi)+0.5))+',');

       writeln(Fichier,'      ]} #8 Vertices of the Torus');

       constante:=1;
       for i:=0 to ((precision1a+1)*precision1b-1) do
         begin
            if i<>(constante*precision1b+constante-1) then
                begin
                    writeln(Fichier,'      IndexedFaceSet { coordIndex [');
                    writeln(Fichier,'
'+RempVirg(FloatToStr(i))+', '

+RempVirg(FloatToStr(i+1))+', '

+RempVirg(FloatToStr(i+precision1b+2))+', '

+RempVirg(FloatToStr(i+precision1b+1))+', -1]}');
                end
            else inc(constante);
          end;


       writeln(Fichier,'      }');
       writeln(Fichier,'  }');
     End;


End;

procedure TSauveVRML.analyseTGL(ob :
TGLBaseSceneObject{;X:Integer;Y:Integer;Z:Integer});
var
  i:integer;
begin
     IF ob is TGLCube then
        Begin                                          //Pour les objets
          CubeVRML(ob as TGLCube);
        End;

     IF ob is TGLCylinder then
        Begin
          CylindreVRML(ob as TGLCylinder);
        End;

     IF ob is TGLCone then
        Begin
```

```
          ConeVRML(ob as TGLCone);
        End;

     IF ob is TGLSphere then
        Begin
          SphereVRML(ob as TGLSphere);
        End;

     IF ob is TGLTorus then
        Begin
          TorusVRML(ob as TGLTorus);
        End;

     for i:=0 to (ob.count-1) do
        Begin
          analyseTGL (ob.Children[i]);
        end;

     if aenfant <> 0 then
     begin
       writeln(Fichier,'          ]');
       writeln(Fichier,'      }');
       aenfant := aenfant - 1;
     end;

end;

procedure TSauveVRML.SauvegardeVRML(N:String;G:TGLScene);
var

  i:integer;
  test:TGLLightsource;

begin

  nom := n;
  GLScene := G;

  // Openen van de file
  assignfile(Fichier,nom);
  Rewrite(Fichier);

  // Begin van de vrml defenitie in de file
  {writeln(Fichier,'#VRML V1.0 ascii');
  writeln(Fichier,'');
  writeln(Fichier,'DEF Viewer Info { string "LACAL 3D" ');
  writeln(Fichier,'DEF Title Info { string "' + nom + '" ');
  writeln(Fichier,'Separator {');}


  //ImplementeObjets(GLScene);

  For i:=0 to (GLScene.Lights.Count-5) do {.LightSources.Count}
     Begin
            If (GLScene.Lights.Items[i] as TGLLightsource).Shining=true
then {.LightSources.Children[i]}           //Pour les Lightsources
              Begin
                LightVRML(GLScene.Lights.Items[i] as TGLLightSource);
{.LightSources.Children[i]}
              End;
     End;
```

```
  For i:=0 to (GLScene.Cameras.Count-1) do
     Begin
             {If (GLScene.Cameras.Children[i] as TGLCamera) =true then
//Pour les Cameras
               Begin}
                  //CameraVRML(GLScene.Cameras.Children[i] as TGLCamera);
               {End;}
     End;


  {writeln(Fichier,'Separator ');}

  For i:=0 to (GLScene.Objects.Count-1) do
    begin
      analyseTGL (GLScene.Objects.Children[i]);
    end;

  {writeln(Fichier,' ');}
  {writeln(Fichier,'');}

  CloseFile(Fichier);

end;

End.
```