```
MODULE  RdcViewInModel;                                  (*  Date    :  19  October  2011  *)
                                                         (*  Author :  Robert D Campbell  *)


IMPORT  Controls, Controllers, Models, Out, Ports, Properties, StdClocks, Stores, Views;


CONST
  minVersion  =  0;
  maxVersion  =  0;

  pt    =  Ports.point;
  mm    =  Ports.mm;
  wid   =  90 * mm;
  hgt   =  35 * mm;

  lightBlue   =  15921766;


TYPE
  View     =  POINTER  TO  RECORD (Views.View)                  (* container view    *)
                 model  :  Model
              END;

  Model    =  POINTER  TO  RECORD (Models.Model)
                 c      :  Context;
                 focus  :  Views.View
              END;

  Context  =  POINTER  TO  RECORD (Models.Context)
                 next                  :  Context;
                 v                     :  Views.View;          (* inner view    *)
                 xPos, yPos, wid, hgt  :  INTEGER              (* of inner view *)
              END;

VAR
  str*  :  ARRAY  16  OF  CHAR;

(**  *****  *****  *****  *****  Context  *****  *****  *****  *****  **)

PROCEDURE (c : Context) ThisModel () : Models.Model;
  BEGIN
    RETURN  NIL
  END  ThisModel;


PROCEDURE (c : Context) GetSize (OUT w, h : INTEGER);
  BEGIN
    w  :=  c.wid; h  := c.hgt
  END  GetSize;


PROCEDURE (c : Context) Normalize () : BOOLEAN;
  BEGIN
    RETURN  FALSE
  END  Normalize;


PROCEDURE (c : Context) Within (x, y : INTEGER) : BOOLEAN, NEW;
  BEGIN
    RETURN  (c.xPos < x)  &  (x < c.xPos + c.wid)
          &  (c.yPos < y)  &  (y < c.yPos + c.hgt)
  END  Within;

(**  *****  *****  *****  *****  View  *****  *****  *****  *****  **)

PROCEDURE (view : View) CopyFromModelView (source : Views.View; model : Models.Model);
  BEGIN
    view.model  :=  model(Model)
  END  CopyFromModelView;
```

```
PROCEDURE (view : View) Externalize (VAR wr : Stores.Writer);
  BEGIN
    wr.WriteVersion (maxVersion);
    wr.WriteStore   (view.model)
  END  Externalize;


PROCEDURE (view : View) HandlePropMsg (VAR msg : Properties.Message);
  BEGIN
    WITH  msg : Properties.SizePref    DO  msg.w      :=  wid; msg.h  :=  hgt
    |     msg : Properties.ResizePref  DO  msg.fixed  :=  TRUE
    ELSE
    END
  END  HandlePropMsg;


PROCEDURE (view : View) HandleCtrlMsg (f : Views.Frame; VAR msg : Views.CtrlMessage;
                                                        VAR focus : Views.View);
  VAR
    m  :  Model;
    c  :  Context;
  BEGIN
    m  :=  view.model; c  :=  m.c;
    WITH  msg : Controllers.PollCursorMsg  DO
       m.focus  :=  view;
       WHILE  c # NIL  DO
         IF  c.Within (msg.x, msg.y)  THEN  m.focus  :=  c.v; RETURN  END;
         c  :=  c.next
       END;
       msg.cursor  :=  Ports.graphicsCursor
    ELSE
    END;
    focus  :=  m.focus
  END  HandleCtrlMsg;


PROCEDURE (view : View) Internalize (VAR rd : Stores.Reader);
  VAR
    thisV  :  INTEGER;
    st     :  Stores.Store;
  BEGIN
    rd.ReadVersion (minVersion, maxVersion, thisV);
    IF  ~rd.cancelled  THEN  rd.ReadStore (st); view.model  :=  st(Model)  END
  END  Internalize;


PROCEDURE (view : View) ThisModel () : Model;
  BEGIN
    RETURN  view.model
  END  ThisModel;


PROCEDURE (view : View) Restore (f : Views.Frame; l, t, r, b : INTEGER);
  VAR
    m     :  Model;
    g     :  Views.Frame;
    w, h  :  INTEGER;
    v     :  Views.View;                                    (* Inner View *)
    c     :  Context;
  BEGIN
    view.context.GetSize (w, h);
    f.DrawRect (0, 0, w, h, Ports.fill, lightBlue);
    f.DrawRect (0, 0, w, h, 0, Ports.black);

    m  :=  view.model; c  :=  m.c;
    WHILE  c  #  NIL  DO
      v  :=  c.v;
      g  :=  Views.ThisFrame (f, v);
      IF  g = NIL  THEN  Views.InstallFrame (f, v, c.xPos, c.yPos, 0, TRUE)  END;
      c  :=  c.next
    END
```

```
  END  Restore;

(**  *****  *****  *****  *****  Model  *****  *****  *****  *****  **)

PROCEDURE (m : Model) AddView (v : Views.View; xPos, yPos, wid, hgt : INTEGER), NEW;
  VAR
    c   :  Context;
  BEGIN
    NEW (c); c.next  :=  m.c; m.c  :=  c;
    c.v     :=  v;
    c.xPos  :=  xPos; c.wid  :=  wid;
    c.yPos  :=  yPos; c.hgt  :=  hgt;
    c.v.InitContext (c); Stores.Join (m, c.v)
  END  AddView;


PROCEDURE (m : Model) CopyFrom (source : Stores.Store);
  VAR
    c, cc  :  Context;
  BEGIN
    c  :=  source(Model).c;
    WHILE  c  #  NIL  DO
      NEW (cc);
      cc^   :=  c;
      cc.v  :=  Views.CopyOf (c.v, Views.deep);
      IF  cc.v.context = NIL  THEN  cc.v.InitContext (cc)  END;
      cc.next  :=  m.c; m.c  :=  cc;
      c  :=  c.next
    END
  END  CopyFrom;


PROCEDURE (m : Model) Externalize  (VAR wr : Stores.Writer);
  VAR
    c  :  Context;
  BEGIN
    wr.WriteVersion (maxVersion);
    c  :=  m.c;
    WHILE  c  #  NIL  DO
      Views.WriteView (wr, c.v);
      wr.WriteInt (c.xPos); wr.WriteInt (c.wid);
      wr.WriteInt (c.yPos); wr.WriteInt (c.hgt);
      c  :=  c.next
    END;
    wr.WriteStore (NIL)
  END  Externalize;


PROCEDURE (m : Model) Internalize  (VAR rd : Stores.Reader);
  VAR
    thisV  :  INTEGER;
    c      :  Context;
    v      :  Views.View;
  BEGIN
    rd.ReadVersion (minVersion, maxVersion, thisV);
    IF  ~rd.cancelled  THEN
      Views.ReadView (rd, v);
      WHILE  v  #  NIL  DO
        NEW (c); c.next  :=  m.c; m.c  :=  c; c.v  :=  v;
        v.InitContext (c);
        rd.ReadInt (c.xPos); rd.ReadInt (c.wid);
        rd.ReadInt (c.yPos); rd.ReadInt (c.hgt);
        Views.ReadView (rd, v)
      END
    END
  END  Internalize;

(**  *****  *****  *****  *****  Deposit  *****  *****  *****  *****  **)

PROCEDURE  NewView () : View;
  VAR
```

```
    m     :  Model;
    view  :  View;
    p     :  Controls.Prop;
  BEGIN
    NEW (m); NEW (view); view.model  :=  m; Stores.Join (m, view);

    m.AddView (StdClocks.New (), 5 * mm, 5 * mm, 25 * mm, 25 * mm);

    NEW (p);
    p.link   :=  'Dialog.Beep'; p.label  :=  'Beep';
    m.AddView (Controls.dir.NewPushButton (p), 35 * mm, 5 * mm, 50 * pt, 18 * pt);

    p.link   :=  'RdcViewInModel.Log(1)'; p.label  :=  'Log 1';
    m.AddView (Controls.dir.NewPushButton (p), 40 * mm, 25 * mm, 50 * pt, 18 * pt);

    p.link   :=  'RdcViewInModel.Log(2)'; p.label  :=  'Log 2';
    m.AddView (Controls.dir.NewPushButton (p), 65 * mm, 5 * mm, 50 * pt, 18 * pt);

    p.link   :=  'RdcViewInModel.str'; p.label  :=  '';
    m.AddView (Controls.dir.NewField (p), 40 * mm, 15 * mm, 120 * pt, 18 * pt);

    RETURN  view
  END  NewView;


PROCEDURE  Deposit*;
  BEGIN
    Views.Deposit (NewView ())
  END  Deposit;


PROCEDURE  Log* (n : INTEGER);
  BEGIN
    Out.String ('Command :'); Out.Int (n, 4); Out.Ln
  END  Log;


END  RdcViewInModel.
```

- DevDebug.Unload
- "RdcViewInModel.Deposit; StdCmds.PasteView"