```
unit NapoHaut;
{
   Example program for playing Napoleon Solitaire
   (Napoleonin hauta in finnish).

   Author:  Vesa Lappalainen
   Date:    17.1.1998
   Changes: 3.9.2001/vl
    + VCL/CLX compaling

}

interface

uses
  SysUtils, Classes,
{$ifdef CLX}
  QGraphics, QControls, QForms, QDialogs, QMenus, QExtCtrls, QTypes, QStdCtrls,
{$else}
  Graphics, Controls, Forms, Dialogs, Menus, ExtCtrls, Types, StdCtrls,
{$endif}
  Kortti, Pakka, Poyta, APakka, Raahausp, SaantoP;

type
  TFormNapo = class(TForm)
    Poyta: TPoyta;
    MainMenu1: TMainMenu;
    File1: TMenuItem;
    Exit1: TMenuItem;
    New1: TMenuItem;
    PanelMuutPakat: TPanel;
    KasiPakka: TSaantoPakka;
    Pakka6: TSaantoPakka;
    Pakka7_3: TSaantoPakka;
    PakkaApu_1: TSaantoPakka;
    Pakka7_1: TSaantoPakka;
    PakkaApu_2: TSaantoPakka;
    PakkaKeski: TSaantoPakka;
    PakkaApu_4: TSaantoPakka;
    Pakka7_4: TSaantoPakka;
    PakkaApu_3: TSaantoPakka;
    Pakka7_2: TSaantoPakka;
    Help1: TMenuItem;
    About1: TMenuItem;
    JakoPakka: TAJakoPakka;
    procedure JakoPakkaClick(Sender: TObject);
    procedure About1Click(Sender: TObject);
    procedure New1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure JakoPakkaFull(sender: TPakka; i: Integer; k: TKortti);
    procedure JakoPakkaInsert(sender: TPakka; i: Integer; k: TKortti);
    procedure PakkaFull(sender: TPakka; i: Integer; k: TKortti);
    procedure Exit1Click(Sender: TObject);
  private
    { Private declarations }
    procedure siivoa_pakka(p:TPakka);
  public
    { Public declarations }
    procedure UusiPeli;
  end;
```



```
var
  FormNapo: TFormNapo;

implementation

uses aboutnap;

{$R *.dfm}

procedure TFormNapo.FormCreate(Sender: TObject);
begin
  Randomize;
  JakoPakka.Sekoita;
{$ifdef CLX}
  Caption := Caption + ', CLX';
{$else}
  Caption := Caption + ', VCL';
{$endif}

end;

procedure TFormNapo.siivoa_pakka(p:TPakka);
begin
//  JakoPakka.siirra(p);  // Hyvä mutta lähtevät liian äkkiä
  if ( p.kortteja <> 0 ) then
    JakoPakka.lisaa_kortti(p.anna_kortti);
end;

procedure TFormNapo.UusiPeli;
begin
  Poyta.ForEveryPakka(siivoa_pakka);
  JakoPakka.siirra(Pakka6);
  JakoPakka.siirra(KasiPakka);
end;


procedure TFormNapo.JakoPakkaClick(Sender: TObject);
begin
  KasiPakka.lisaa_kortti(JakoPakka.anna_kortti);
end;

procedure TFormNapo.About1Click(Sender: TObject);
begin
   AboutBox.ShowModal;
end;

procedure TFormNapo.New1Click(Sender: TObject);
begin
  UusiPeli;
end;

procedure TFormNapo.JakoPakkaFull(sender: TPakka; i: Integer; k: TKortti);
begin
  JakoPakka.Sekoita;
end;

procedure TFormNapo.JakoPakkaInsert(sender: TPakka; i: Integer;
  k: TKortti);
begin
  if JakoPakka.Kortteja <> 52 then
```

```
    UusiPeli;
end;

procedure TFormNapo.PakkaFull(sender: TPakka; i: Integer; k: TKortti);
begin
  sender.Puoli := selka;
end;

procedure TFormNapo.Exit1Click(Sender: TObject);
begin
  Close;
end;



end.
```



```
unit Kortti;
(*
  Kortti-komponentti:
  - Create:sta kaksi eri versiota, huomaa niiden välinen ero.
  - FOwnerin muuttaminen MIELELLÄÄN harkitusti.

  5.11.1997/ Tanja Partanen ja Niina Riikonen

  Muutoksia 8.11.1997/vl
   + Kortin Owneriin liittyvät asiat pois, koska Ownewria ei nähtävästi
     voi muuttaa
   + Create2 => Create3 ja muutettu param. järjestys AOwner ensimmäiseksi
   + TKortti, josta peritään TKuvaKortti, josta peritään TPeliKortti
   + TKuvaKorttin geneeriset tiedostonimet card0n ja cardnn
   + Jos TKortti peritään, kirjoitetaan lataa ja Paint uudestaan
   + SortValue = arvo, jonka mukaan lajitellaan
   + OnFlip - event
  Muutoksia 11.11.1997/vl
   + testit, ettei päivitystä tehdä jollei ole muutoksia
  Muutoksia 01.12.1997/vl
  + Kuvakortin lataaminen tiedostosta, JOS nimi päättyy .bmp.
    Muuten kortti ladataan resursseista (ks. LoadBmp-metodi).
    Resurssit (*.res) voidaan tehdä esim. BC 5.02:lla *.rc tiedosto, jossa
    lukee:
      c01 BITMAP "c01.bmp"
      c02 BITMAP "c02.bmp"
      jne...
    Sitten .rc käännetään.  Täytyy pitää huoli, että kohteena on 32-bit!
    Tämä onnistuu esim. tekemällä projekti, jossa on vain .rc.
    Käännöksen tuloksena syntyy .res, joka liitetään Pascal-tiedostoon
    esim:
    {$R kortit.res}

  Tekemättä
  - kannattaisi jakaa kahdeksi tiedostoksi, jossa toisessa
    olisi kortti ja kuvakortti ja toisessa pelikortti + valmiit
    pelikortin kuvat.
  - samoin pakan kohdalla JakoPakka kannattaisi siirtää omaan
    tiedostoonsa.
*)

interface

uses
{$ifdef CLX}
  SysUtils,Classes,QGraphics,QControls,
  QForms,QDialogs,QExtCtrls,QImgList,Types;
{$else}
  SysUtils,Classes,Graphics,Controls,
  Forms,Dialogs,ExtCtrls,ImgList,Types;
{$endif}

type
  TNakyy  = ( Piilossa, Selka, Kuva );

  TKorttiMeth = procedure (sender:TObject;nakyy:TNakyy) of object;

  TKortti = class(TPaintBox)
  private
    FArvo  : integer;
```

```
    FNakyy : TNakyy;
    FOnFlip : TKorttiMeth;
//    FOwner : TComponent;
  protected
    procedure Paint; override;
  public
    constructor Create(AOwner: TComponent); override;
    constructor Create2(AOwner:TComponent;i:integer); virtual;
    procedure   SetArvo(i:integer);         virtual;
    procedure   SetNakyy(n:TNakyy);         virtual;
    procedure   Lataa;                      virtual;
//    procedure   SetOwner(c:TComponent);     virtual;
//    property    Owner:TComponent read FOwner write SetOwner;
    function SortValue : integer;           virtual;
  published
    property Arvo:integer read FArvo write SetArvo;
    property Nakyy:TNakyy read FNakyy write SetNakyy;
    property OnFlip : TKorttiMeth read FOnFlip write FOnFlip;
  end;

  TKuvaKortti = class(TKortti)
  private
    FTausta : string;
    FImageList : TImageList;
    procedure SetTausta(const Value: string);
  protected
    procedure Paint; override;
  public
    constructor Create2(AOwner:TComponent;i:integer); override;
    procedure   Lataa;                      override;
    procedure   LoadBmp(bmp:TBitMap;name:string); virtual;
    function    Tiedosto:string;            virtual;
  published
    property Tausta:string read FTausta write SetTausta;
  end;

  TMaa    = ( Ruutu, Hertta, Risti, Pata );

  TPeliKortti = class(TKuvaKortti)
  private
    FMaa   : TMaa;
  protected
  public
    constructor Create(AOwner: TComponent); override;
    constructor Create3(AOwner:TComponent;i:integer;maa:TMaa); virtual;
    procedure   SetArvo(i:integer);         override;
    procedure   SetMaa(m:TMaa);             virtual;
    function    Tiedosto:string;            override;
    function    SortValue : integer;        override;
  published
    property Maa:TMaa read FMaa write SetMaa;
  end;

procedure Register;

implementation

//=============================================================================
// TKortti
//=============================================================================
constructor TKortti.Create(AOwner: TComponent);
```



```
begin
  Create2(AOwner,1);
end;

constructor TKortti.Create2(AOwner:TComponent;i:integer);
begin
  inherited Create(AOwner);
  FNakyy  := Piilossa;  // Huomaa, että kaikki kortit syntyvät piilossa oleviksi
  Visible := False;
  FArvo   := i;
  Width   := 50;
  Height  := 100;
//  AutoSize := True;
  lataa;
end;

procedure TKortti.Paint;
var s:string; ts : TSize;
begin
  inherited;
  s := IntToStr(Arvo);
  ts := Canvas.TextExtent(s);
  Canvas.Brush.Color := Color;
  Canvas.Pen.Style := psSolid;
  Canvas.Rectangle(0,0,Width,Height);
  if ( Nakyy = Kuva ) then
    Canvas.TextOut((Width-ts.cx) div 2,(Height-ts.cy) div 2,s);
end;

function TKortti.SortValue : integer;
begin
  Result := arvo;
end;

procedure TKortti.SetArvo(i:integer);
begin
  if ( FArvo = i ) then exit;
  FArvo := i;
  Lataa;
end;

procedure TKortti.SetNakyy(n:TNakyy);
begin
  if ( FNakyy = n ) then exit;
  FNakyy := n;
  Visible := n <> Piilossa;
  Invalidate;
  if Assigned(OnFlip) then OnFlip(self,Nakyy);
//  Lataa;
end;
{
procedure TKortti.SetOwner(c:TComponent);
begin
  FOwner := c;
end;
}

procedure TKortti.Lataa;
begin
  Invalidate;
end;
```

```
//============================================================================
// TKuvaKortti
//============================================================================
constructor TKuvaKortti.Create2(AOwner:TComponent;i:integer);
begin
  if ( FTausta = '' ) then FTausta := 'ktausta.bmp';
  FImageList := TImageList.Create(self);
  FImageList.Masked := false;
  inherited Create2(AOwner,i);
end;

procedure TKuvaKortti.LoadBmp(bmp:TBitMap;name:string);
begin
  if ( Pos('.BMP',UpperCase(name)) > 0 ) then
    bmp.LoadFromFile(name)
  else
    bmp.LoadFromResourceName(hInstance,UpperCase(name));
end;

procedure TKuvaKortti.Lataa;
var bmp  : TBitmap; // Apubittikartta
begin
  FImageList.Clear;                          // Pyyhitään vanhat kuvat pois
  bmp := TBitmap.Create;                     // Luodaan tilap. bittikartta
  LoadBmp(bmp,Tausta);                       // Luetaan tausta
  bmp.Monochrome := false;
  Width := bmp.Width;                        // Koko otetaan taustakuvasta
  Height := bmp.Height;
  FImageList.Width  := Width;                // Samoin kuvalistan
  FImageList.Height := Height;
  FImageList.Add(bmp,nil);                   // Kuvalistan paik.0 tausta

  LoadBmp(bmp,Tiedosto);
  bmp.Monochrome := false;
  FImageList.Add(bmp,nil);                   // Kuvalistan paik.1 kuva
  bmp.Free;
  Invalidate;
end;

procedure TKuvaKortti.Paint;
var index : integer;
begin
  index := 0; if ( Nakyy = Kuva ) then index := 1; // Kumpiko kuva pitää piirtää
  FImageList.Draw(Canvas,0,0,index);
end;

function TKuvaKortti.Tiedosto:string;
var k:string;
begin
  k := 'card';
  if ( Arvo < 10 ) then k := k + '0';
  k := k + IntToStr(Arvo) + '.bmp';
  result := k;
end;

procedure TKuvaKortti.SetTausta(const Value: string);
begin
  FTausta := Value;
  Lataa;
end;
```



```
//============================================================================
// TPeliKortti
//============================================================================
{$R pelikort.res pelikort.rc}

constructor TPeliKortti.Create(AOwner: TComponent);
begin
  Create3(AOwner,1,Ruutu);
end;

constructor TPeliKortti.Create3(AOwner:TComponent;i:integer;maa:TMaa);
begin
  FMaa := maa;
  FTausta := 'ktausta';                          // Ladataan resursseista
  inherited Create2(AOwner,i);
end;

procedure TPeliKortti.SetMaa(m:TMaa);
begin
  if ( FMaa = m ) then exit;
  FMaa := m;
  Lataa;
end;

procedure TPeliKortti.SetArvo(i:integer);
begin
  if ( FArvo = i ) then exit;
  case i of
    1 .. 13 : FArvo := i;
  else exit;
  end;
  Lataa;
end;

function TPeliKortti.Tiedosto:string;
var k:string;
begin
  case Maa of
    Ruutu  : k := 'd';
    Hertta : k := 'h';
    Risti  : k := 'c';
    Pata   : k := 's';
  end;
  if ( Arvo < 10 ) then k := k + '0';
  Result := k + IntToStr(Arvo); // + '.bmp';

//  Result := Format('%s%02d',[k,Arvo]);
end;

function TPeliKortti.SortValue : integer;
begin
  Result := arvo-1 + Ord(maa)*13;
end;

procedure Register;
begin
  RegisterComponents('GKO', [TKortti,TKuvaKortti,TPeliKortti]);
end;

end.
```

```
unit Pakka;
{
  Pakka-komponentti:
  - Omaa yleisen pakan ominaisuudet
  - Huomaa erityisesti:
   * Anna_kortti-metodi palauttaa pyydetyn kortin ja samalla POISTAA
     sen pakasta
   * Pakkaan voi siirtää kaikki toisen pakan kortit käyttämällä
     Siirra-metodia.
   * Jos halutaan klikattava pakka ilman lisäkoodausta, niin päälle voidaan
     laittaa TImage, johon ei laiteta mitään kuvaa.
     Itse pakan Enable asetetaan tällöin
     epätodeksi ja TImagen klikkauksella tehdään mitä tarvitsee.

  JakoPakka-komponentti:
   - sisältää kaikki kortit jo syntyessään

  5.11.1997/ Tanja Partanen ja Niina Riikonen

  Muutoksia 8.11.1997/vl
   + pakan tapahtumat OnTake, OnEmpty, OnInsert, OnFull
   + MaxKortteja ( = hälytysraja pakan täyttymiselle )
   + Kortin Owneriin liittyvät asiat pois, kosa Ownewria ei nähtävästi
     voi muuttaa
   + Lisätty jakopakkaan metodi Uusi, jotta korttiluokan vaihtaminen olisi
     helpompaa.
   + jarjesta
  Muutoksia 8.11.1997/vl
   + ParentColor := true

   + Lisättävä kortti piilotetaan vain jos se menee keskelle pakkaa
  Muutoksia 17.1.1998/vl
   + SetPuoli metodiin lisätty  SetVisible jotta kääntää samalla
     pakan vastaamaan nykyistä tilannetta
  Muutoksia 12.11.1999/mk ja lp
  + Lisaa_paikkaan-metodi jaettu kahteen osaan: lisaa_paikkaan ja aseta.
    Jos perittäessä halutaan asetella pakan kortit muuten kuin päällekkäin
    pinoon, kirjoitetaan aseta-metodi uudestaan.
  Muutoksia 10.12.1999/mk ja lp
  + tee_tarkistukset kortin poistuessa jarjetsaa pakan taas kuntoon
  + sotke on sekoittamista varten oleva aliohjelma, jonka aikana ei
    laheteta eventteja poistamisesta eikä lisäämisestä
  + FDoEvent protected (jotta voi tarvittaessa käyttää)
  Muutoksia 27.8.2001/vl
  + hiiren tapahtumat pakalta siirtyvät kortille, jotta korttia
    pakkaa voi "napsauttaa"
    => tarvittaessa override metodit PakkaClick jne...
  + Kelpaako ja SaakoOttaa siirretty jo pakkaan.
  Muutoksia 2.9.2001/vl
  + ehdollinen kääntäminen CLX/VCL (pitää määritellä vakio CLX jos
    halutaan CLX-kirjaston mukainen käännös
}

interface

uses
{$ifdef CLX}
  SysUtils, Classes, QGraphics, QControls, QForms, QDialogs,
  QExtCtrls,QStdCtrls,
{$else}
  SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls,StdCtrls,
{$endif}
  Kortti;

type
  TPakka = class;
  TPakkaChangeMeth = procedure (sender:TPakka;i:integer;k:TKortti) of object;

  TPakka = class(TPanel)
  private
    Fkortit: TList; // array of TKortti
    FPuoli : TNakyy; //Tämä ominaisuus maaraa mitenpain pakka on.
    FMaxKortteja : integer;                                          // vl
    FOnInsert : TPakkaChangeMeth;                                    // vl
    FOnTake : TPakkaChangeMeth;                                      // vl
    FOnEmpty : TPakkaChangeMeth;                                     // vl
    FOnFull : TPakkaChangeMeth;
  protected
    FDoEvent : boolean;                                              // vl
    procedure PakkaMouseDown(Sender: TObject; Button: TMouseButton;
```



```
      Shift: TShiftState; X, Y: Integer);  virtual;
    procedure PakkaMouseMove(Sender: TObject; Shift: TShiftState; X,
      Y: Integer);  virtual;
    procedure PakkaMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);  virtual;
    procedure PakkaClick(Sender: TObject); virtual;
    procedure PakkaDblClick(Sender: TObject); virtual;
  public
    constructor Create(AOwner:TComponent);              override;
    destructor  Destroy;                                override;
    function  kortteja:integer;                         virtual;
    procedure lisaa_loppuun(kortti:TKortti);            virtual;
    procedure lisaa_paikkaan(i:integer;kortti:TKortti); virtual;
    procedure lisaa_kortti(kortti:TKortti);             virtual;
    function  anna_kortti_i(i:integer):TKortti;         virtual;
    function  anna_kortti:TKortti;                      virtual;
    function  anna(k:TKortti):TKortti;                  virtual;
    function  anna_arvo:integer;                        virtual;
    function  anna_arvo_i(i:integer):integer;           virtual;
    procedure siirra(p:TPakka);                         virtual;
    procedure sekoita;                                  virtual;
    procedure jarjesta;                                 virtual;          // vl
    procedure SetPuoli(p:TNakyy);                       virtual;
    procedure SetVisible;                               virtual;

    procedure aseta(i:integer; kortti:TKortti);         virtual;
    procedure tee_tarkistukset(k: TKortti; i: integer); virtual;    // vl&mk&lp
    procedure sotke;                                    virtual;    // vl&mk&lp
    function  Kelpaako(k:TKortti):boolean;              virtual;
    function  SaakoOttaa(k:TKortti):boolean;            virtual;

    property  kortit:TList read Fkortit;
  published
    property  Puoli:TNakyy read FPuoli write SetPuoli;
    property  MaxKortteja : integer read FMaxKortteja write FMaxKortteja default 52;
    property  OnInsert : TPakkaChangeMeth read FOnInsert write FOnInsert;
    property  OnEmpty  : TPakkaChangeMeth read FOnEmpty write FOnEmpty;
    property  OnFull : TPakkaChangeMeth read FOnFull write FOnFull;
    property  OnTake : TPakkaChangeMeth read FOnTake write FOnTake;
  end;

type
  TJakoPakka = class(TPakka)
  private
  protected
  public
    function  Uusi(i:integer; maa:TMaa):TKortti;        virtual;
    procedure lisaa_kortit;                             virtual;
    procedure lisaa_maa(maa:TMaa);                      virtual;
    constructor Create(AOwner:TComponent);              override;
  published
  end;

procedure Register;

implementation
uses
  Liikkuva;

constructor TPakka.Create(AOwner:TComponent);
// Jos mielii pakkaa, joka olisi JO suunnitteluaikana kuvapuoli ylöspäin
// konstruktori täytyy FPuolen osalta kirjoittaa uusiksi.
begin
  Inherited;
  Caption      := ' ';
  BevelOuter   := bvNone;
  ParentColor  := true;
  Fkortit      := TList.Create;
  FPuoli       := Selka;
  FMaxKortteja := 52;
  FOnEmpty     := nil;
  FOnTake      := nil;
  FOnInsert    := nil;
  FOnFull      := nil;
  FDoEvent     := true;
end;

destructor  TPakka.Destroy;
begin
  kortit.Clear;  // Turha
  kortit.Free;
```

```
  inherited;
end;

function  TPakka.kortteja:integer;
begin
  result := kortit.Count;
end;

procedure TPakka.lisaa_loppuun(kortti:TKortti);
begin
  lisaa_paikkaan(kortteja,kortti);
end;

procedure TPakka.lisaa_kortti(kortti:TKortti);
begin
  lisaa_paikkaan(0,kortti);
end;

procedure TPakka.aseta(i: integer; kortti: TKortti);
var apu:TKortti;
begin
  if ( i=0 ) then begin                 //lisätään pakan päällimmäiseksi
    if ( kortteja > 0 ) then begin
      apu := kortit[0];
      apu.Nakyy := Piilossa;            //edellinen kortti piilotetaan
    end;
    kortti.Nakyy := FPuoli;
  end
  else kortti.Nakyy  := Piilossa;      //vain päällimmäinen kortti näkyvissä,
                                       //muut piilossa
end;

procedure TPakka.PakkaClick(Sender: TObject);
begin
  if ( Assigned(OnClick) ) then OnClick(Sender);
end;

procedure TPakka.PakkaDblClick(Sender: TObject);
begin
  if ( Assigned(OnDblClick) ) then OnDblClick(Sender);
end;

procedure TPakka.PakkaMouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
  if ( Assigned(OnMouseDown) ) then OnMouseDown(Sender,Button,Shift,X,Y);
end;

procedure TPakka.PakkaMouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
  if ( Assigned(OnMouseUp) ) then OnMouseUp(Sender,Button,Shift,X,Y);
end;

procedure TPakka.PakkaMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
begin
  if ( Assigned(OnMouseMove) ) then OnMouseMove(Sender,Shift,X,Y);
end;

procedure TPakka.lisaa_paikkaan(i:integer;kortti:TKortti);
begin
  if ( (i < 0) or (i > kortteja) ) then i := kortteja;
  if ( kortti = nil ) then exit;
  if ( kortteja = 0 ) then Show;
  kortti.Parent := self;
//  kortti.Owner  := self;
  aseta(i, kortti);
  kortit.insert(i,kortti);
  if ( Assigned(FOnInsert) and FDoEvent ) then FOnInsert(self,i,kortti);
  if ( kortteja >= MaxKortteja ) and ( Assigned(FOnFull) and FDoEvent ) then
    FOnFull(self,i,kortti);
  kortti.OnClick := PakkaClick;
  kortti.OnDblClick := PakkaDblClick;
  kortti.OnMouseDown := PakkaMouseDown;
  kortti.OnMouseUp := PakkaMouseUp;
  kortti.OnMouseMove := PakkaMouseMove;

{$ifdef CLX}
//  MouseCapture := false;  // Qt vaatii tämän jostakin syystä
{$else}
//  MouseCapture := false;  // Qt vaatii tämän jostakin syystä
{$endif}
end;
```



```
procedure TPakka.tee_tarkistukset(k:TKortti;i:integer);
begin
  if ( kortit.Count = 0 ) then //Hide
  else
    if ( i = 0 ) then TKortti(kortit.Items[0]).Nakyy := puoli;
end;

function  TPakka.anna_kortti_i(i:integer):TKortti;
var apu:TKortti;
begin
  if( kortit.Count = 0 )then begin
    result := nil; exit;
  end;
  if( (i < 0) or (i > kortteja-1) ) then i := kortteja - 1;
  SetCaptureControl(nil);
  apu :=kortit[i];  // kortit[i].OnClick := nil;
//  apu.Owner := NIL;
//  apu.Nakyy := Piilossa;
//  anna_kortti_i := apu;                                          // vl
  Result := apu;  // ei ihan sama kuin c:n return apu;
  apu.OnClick := nil;
  apu.OnDblClick := nil;
  apu.OnMouseDown := nil;
  apu.OnMouseUp := nil;
  apu.OnMouseMove := nil;
  kortit.Remove(apu);
  tee_tarkistukset(apu,i);

  if ( Assigned(FOnTake) and FDoEvent ) then FOnTake(self,0,Result);
  if ( Assigned(FOnEmpty) and FDoEvent and ( Kortteja = 0 ) ) then
    FOnEmpty(self,0,Result);
end;

function  TPakka.anna(k:TKortti):TKortti;
var i:integer;
begin
  Result := nil;
  for i:=0 to kortteja-1 do
    if kortit.Items[i] = k then begin
      Result := anna_kortti_i(i);
      Exit;
    end;
end;

function  TPakka.anna_kortti:TKortti;
begin
  result := anna_kortti_i(0); // Annetaan siis pakan ensimmäinen kortti.
end;

function  TPakka.anna_arvo:integer;
begin
  result := anna_arvo_i(0);
end;

function  TPakka.anna_arvo_i(i:integer):integer;
var apu:TKortti;
begin
  if(kortit.Count = 0) then begin result := 0; exit; end;
  if( (i < 0) or (i > kortteja-1) ) then i := kortteja-1;
  apu := kortit.Items[i];
  result := apu.Arvo;
end;

procedure TPakka.siirra(p:TPakka);
begin
  if( p = self ) then exit;
  while (p.kortteja > 0) do
    lisaa_kortti(p.anna_kortti);
end;

procedure TPakka.sotke;
var i:integer;
begin
  for i := kortit.Count downto 1 do
    lisaa_loppuun(anna_kortti_i(random(i)));
end;

procedure TPakka.sekoita;
begin
  FDoEvent := false;
```

```
  sotke;
  FDoEvent := true;
end;

procedure TPakka.SetPuoli(p:TNakyy);
begin
  FPuoli := p;
  SetVisible;
end;

function compare(item1,item2:Pointer) : integer;
var k1,k2:TKortti;
begin
  k1 := item1;
  k2 := item2;
  Result := k1.SortValue - k2.SortValue;
end;

procedure TPakka.SetVisible;
var i:integer; k:TKortti;
begin
  for i:=0 to Kortit.Count-1 do begin
    k := Kortit[i];
    if ( i = 0 ) then k.Nakyy := Puoli
                 else k.Nakyy := Piilossa;
  end;
end;

procedure TPakka.jarjesta;
begin
  Fkortit.Sort(compare);
  SetVisible;
end;

function TPakka.Kelpaako(k:TKortti):boolean;
begin
  Result := kortteja < MaxKortteja;
end;

function TPakka.SaakoOttaa(k:TKortti):boolean;
begin
  Result := true;
end;

//---------------------------------------------------------------------
// Jakopakan omat metodit:

constructor TJakoPakka.Create(AOwner:TComponent);
begin
  inherited;
  lisaa_kortit;
end;

function TJakoPakka.Uusi(i:integer; maa:TMaa) : TKortti;
begin
  Result := TPeliKortti.Create3(self,i,maa);
end;

procedure TJakoPakka.lisaa_maa(maa:TMaa);
var i:integer;
var apu:TKortti;
begin
  for i:= 1 to 13 do begin
    apu := uusi(i,maa);
    lisaa_kortti(apu);
  end;
end;

procedure TJakoPakka.lisaa_kortit;
var maa:TMaa;
begin
//  for maa:=Ruutu to Ruutu do lisaa_maa(maa);
  for maa:=Ruutu to Pata do lisaa_maa(maa);
end;

procedure Register;
begin
  RegisterComponents('GKO', [TPakka,TJakoPakka]);
end;

end.
```



```
unit APakka;
{
  Animoitu pakka-komponentti:
  - Omaa yleisen pakan ominaisuudet
  - Siirtää kortin "lentämällä"
  AJakoPakka-komponentti:
   - sisältää kaikki kortit jo syntyessään

  8.11.1997/ Vesa Lappalainen
  Muutoksia:

  11.11.1997 / vl
   + lisätty testi siitä, ettei nil-kortti aiheuta ongelmia
}

interface

uses
  SysUtils, Classes,
{$ifdef CLX}
  QGraphics, QControls, QForms, QDialogs, QExtCtrls,QStdCtrls,
{$else}
  Graphics, Controls,   Forms,  Dialogs,  ExtCtrls, StdCtrls,
{$endif}
  Kortti,Pakka;

type
  TAPakka = class(TPakka)
  private
    FSpeed : integer;
  protected
  public
    procedure Arrive(o:TControl; i:integer);            virtual;
    procedure lisaa_paikkaan(i:integer;kortti:TKortti); override;
  published
    property Speed : integer read FSpeed write FSpeed default 0;
  end;

type
  TAJakoPakka = class(TJakoPakka)
  private
    FSpeed : integer;
  protected
  public
    procedure Arrive(o:TControl; i:integer);            virtual;
    procedure lisaa_paikkaan(i:integer;kortti:TKortti); override;
  published
    property Speed : integer read FSpeed write FSpeed default 0;
  end;

procedure Register;

implementation
uses Liikkuva;


procedure TAPakka.Arrive(o:TControl; i:integer);
begin
  if not ( o is TKortti ) then exit;
  inherited lisaa_paikkaan(i,o as TKortti);
```

```
end;

procedure TAPakka.lisaa_paikkaan(i:integer;kortti:TKortti);
begin
  if ( kortti = nil ) then exit;
  if ( kortti.Parent = nil ) or ( kortti.Parent = self ) then begin
    inherited lisaa_paikkaan(i,kortti);
    exit;
  end;
  kortti.Visible := true;
  Kuljeta(kortti,self,Arrive,i,Speed);
end;


//---------------------------------------------------------------------
// Jakopakan omat metodit:

procedure TAJakoPakka.Arrive(o:TControl; i:integer);
begin
  if not ( o is TKortti ) then exit;
  inherited lisaa_paikkaan(i,o as TKortti);
end;

procedure TAJakoPakka.lisaa_paikkaan(i:integer;kortti:TKortti);
begin
  if ( kortti = nil ) then exit;
  if ( kortti.Parent = nil ) or ( kortti.Parent = self ) then begin
    inherited lisaa_paikkaan(i,kortti);
    exit;
  end;
  kortti.Visible := true;
  Kuljeta(kortti,self,Arrive,i,Speed);
end;


procedure Register;
begin
  RegisterComponents('GKO', [TAPakka,TAJakoPakka]);
end;

end.
```



```
unit Liikkuva;
{
  Olio, joka on tehty kuljettamaan muita olioita paikasta toiseen.
  Vesa Lappalainen 7.11.1997

  Liikuttaa olion olion nykyisetä paikasta joko annetun WinControllin
  paikkaan pt (KuljetaPt) tai annettun kontrollin päälle (Kuljeta).
  Jos käytetään funktiota tai metodia KuljetaPT, pitää sekä lähtöpaikka
  että saapumispaikka olla saman kontrollin alueella.
  Saadaan tapahtuma jokaisesta liikahduksesta (OnMove) sekä perille
  saapumisesta (OnArrive).
  Matka tehdään joko ajassa Time tai nopeudella SpeedPts (pxl/s)
  (käytetään sitä joka <> 0). Matkalla otetaan Steps askelta
  tai siirtojen väli on Interval (käytetään sitä, joka <>0).

  Käyttöä helpottamaan on tehty kaksi funktiota, jotka luovat liikuvan
  olio, siirtävät siirrettävän olien liikkuvan sisälle ja ilmoittavat
  lopuksi kun siirto on valmis.

  function KuljetaPt(OWhat : TControl; f:TWinControl;ToPt:TPoint;
                     arrive:TLiikkuvaMeth;data : integer; v:integer) : boolean;
  function Kuljeta(OWhat,OTo : TControl; arrive:TLiikkuvaMeth;
                     data:integer; v:integer) : boolean;
  Parametrit:
    OWhat    - siirrettävä olio
    f        - ikkuna jonka sisällä siirto tehdään
    ToPt     - koordinaatti, johon siirto loppuu
    arrive   - metodi, joka käsittelee valmiin siirron
    data     - vapaasti käytettävä sana, joka saadaan käyttöön siirron loputtua
    v        - jos >0 siirron nopeus pxl/s
               jos <0 siirron kesto ms (abs(v))
    OTo      - olio johon siirto päättyy

  Muutoksia 11.11.1997/vl
   + olion Parent := nil ennen Arrive-metodin kutsumista
  Muutoksia 26.11.1998/vl
   + KuljetaCrtlPt joka kuljettaa kontrollin tiettyyn pisteeseen.
     Itse asiassa Kuljeta on nyt tämän erikoistapaus, jossa kuljetetaan
     ctrl:in yläkulmaan.
  Muutoksia 03.09.2001/vl
   + VCL/CLX-käännös (määrittele vakio CLX)
}

interface

uses
  SysUtils, Classes,
{$ifdef CLX}
  QGraphics, QControls, QForms, QDialogs,QExtCtrls, Qt,
{$else}
  Graphics, Controls, Forms, Dialogs, ExtCtrls,Messages,Windows,
{$endif}
  Types;

type
  TLiikkuvaMeth = procedure (o:TControl;data:integer) of object;
  TLiikkuvaMoveMeth = procedure (o:TControl;pt:TPoint;data:integer) of object;

  TLiikkuva = class(TWinControl)
  private
    FTimer : TTimer;
    FOnArrive : TLiikkuvaMeth;
    FOnMove   : TLiikkuvaMoveMeth;
    FObject   : TControl;
    FDpt      : TPoint;
    FTo       : TPoint;
    FFrom     : TPoint;
    FData     : Integer;
```

```
    FStep     : integer;
    FSteps    : integer;
    FTime     : integer;
    FSpeedPts : integer;
    FInterval : integer;

    FUseSteps : integer;

    FAutoDestroy : boolean;
    procedure FTimerTimer(Sender: TObject);
  protected
  public                                                override;
    constructor Create(AOwner:TComponent);              override;
    destructor  Destroy;
    function KuljetaPt(OWhat : TControl; f:TWinControl;ToPt:TPoint;
                       data : integer) : boolean; virtual;
    function KuljetaCtrlPt(OWhat, OTo: TControl; ToPt: TPoint;
      data: integer): boolean;
    function Kuljeta(OWhat,OTo : TControl; data:integer):boolean; virtual;
    function Moving : boolean;                          virtual;
{$ifdef CLX}
{$else}
//    procedure PaintWindow(DC: HDC); override;
    procedure WmClose(var Message: TWMClose); message WM_CLOSE;
{$endif}
  published
    property OnArrive : TLiikkuvaMeth read FOnArrive write FOnArrive;
    property OnMove   : TLiikkuvaMoveMeth read FOnMove write FOnMove;
    property Steps : integer read FSteps write FSteps default 20;
    property TimeMs : integer read FTime write FTime default 1000;
    property SpeedPts : integer read FSpeedPts write FSpeedPts default 0;
    property Interval : integer read FInterval write FInterval default 0;
  end;

function KuljetaPt(OWhat : TControl; f:TWinControl;ToPt:TPoint;
                   arrive:TLiikkuvaMeth;data : integer; v:integer) : boolean;
function Kuljeta(OWhat,OTo : TControl; arrive:TLiikkuvaMeth;
                 data:integer; v:integer) : boolean;
function KuljetaCtrlPt(OWhat,OTo : TControl; ToPt:TPoint;
                   arrive:TLiikkuvaMeth;data : integer; v:integer) : boolean;

procedure Register;

implementation

var FOrigo : TPoint =(x:0;y:0);



constructor TLiikkuva.Create(AOwner:TComponent);
begin
  inherited;
//  BevelInner := bvNone;
//  BevelOuter := bvNone;
//  ParentColor := true;
  FTimer := TTimer.Create(self);
  FTimer.Enabled := false;
  FTimer.OnTimer := FTimerTimer;
  FAutoDestroy := false;
  FOnMove := nil;
  FOnArrive := nil;
  FOrigo.x := 0; FOrigo.y := 0;
  FInterval := 0;
  FSteps    := 20;
  FSpeedPts := 0;
  FTime     := 1000;
end;

destructor  TLiikkuva.Destroy;
```



```
begin
  FTimer.Free;
  inherited;
end;

function TLiikkuva.Moving : boolean;
begin
  Result := FTimer.Enabled;
end;

function TLiikkuva.KuljetaPt(OWhat : TControl; f:TWinControl;ToPt:TPoint;
                             data : integer) : boolean;
var dist,t,dt : integer;
begin
  Result := false;
  if ( Moving ) then exit;
  FObject := OWhat;
  FFrom := f.ScreenToClient(FObject.ClientToScreen(FOrigo));
//  FObject.Owner := self;
  FObject.Parent := self;
  FObject.Top := 0;
  FObject.Left := 0;
  FData := data;
  Width := FObject.Width;
  Height := FObject.Height;
  Parent := f;

  FTo := ToPt;
  Left := FFrom.x;
  Top := FFrom.y;

  FUseSteps := 0;  if ( Steps <> 0 ) then FUseSteps := Steps;

  FDpt.x := (FTo.x - FFrom.x);
  FDpt.y := (FTo.y - FFrom.y);
  Dist := round(sqrt(1.0*FDpt.x*FDpt.x + 1.0*FDpt.y*FDpt.y));

  t := TimeMs;
  if ( SpeedPts <> 0 ) then  // v = s/t
    t := Round(1000*(dist/SpeedPts));
  if ( t = 0 ) then t := 1000;

  dt := 100;
  if ( Interval > 0 ) then dt := Interval;
  if ( FUseSteps <> 0 ) then dt := t div FUseSteps
  else FUseSteps := t div dt;

  if ( dt < 30 ) then begin
    dt := 30;
    FUseSteps := t div dt;
  end;

  If FUseSteps = 0 then FUseSteps := 1;
  FStep := 0;
  FTimer.Interval := dt;
  FTimer.Enabled := true;
  Visible := true;
end;

function TLiikkuva.KuljetaCtrlPt(OWhat,OTo : TControl; ToPt:TPoint;
                             data : integer) : boolean;
var p : TObject; f:TForm; pt:TPoint;
begin
  Result := false;
  if ( Moving ) then exit;
  p := OTo;
  while ( p<> nil ) and (not ( p is TForm )) do
    p := (p as TControl).Parent;
  if ( p = nil ) then exit;
```

```
  f := p as TForm;

  pt := f.ScreenToClient(OTo.ClientToScreen(ToPt));

  Result := KuljetaPt(OWhat,f,pt,data)
end;

function TLiikkuva.Kuljeta(OWhat, OTo: TControl; data: integer): boolean;
begin
  Result := KuljetaCtrlPt(OWhat,OTo,FOrigo,data)
end;

procedure TLiikkuva.FTimerTimer(Sender: TObject);
var pt:TPoint;
begin
  Left := FFrom.x + Round(FStep*(FDpt.x/FUseSteps));
  Top  := FFrom.y + Round(FStep*(FDpt.y/FUseSteps));
  pt.x := Left;
  pt.y := Top;
  inc(FStep);
  if ( Assigned(FOnMove) ) then OnMove(FObject,pt,FData);
  if ( FStep <= FUseSteps ) then exit;
  FTimer.Enabled := false;
  FObject.Parent := nil;
  if ( Assigned(FOnArrive) ) then OnArrive(FObject,FData);
//  if ( FAutoDestroy ) then Free;
{$ifdef CLX}
  if ( FAutoDestroy ) then QApplication_postEvent(Application.Handle, QCustomEvent_create(QE
{$else}
  if ( FAutoDestroy ) then PostMessage(Handle,WM_CLOSE,0,0);
{$endif}
end;

{$ifdef CLX}
{$else}
procedure TLiikkuva.WmClose(var Message: TWMClose);
begin
  Free;
end;
{$endif}


function InitLiikkuva(var l:TLiikkuva;f:TComponent;
                      arrive:TLiikkuvaMeth; v:integer) : boolean;
begin
  Result := false;
  l := TLiikkuva.Create(f);
  if ( l = nil ) then exit;
  l.Visible := false;
  l.FAutoDestroy := true;
  l.OnArrive := arrive;
  if ( v > 0 ) then l.SpeedPts := v
               else l.TimeMs := abs(v);
  Result := true;
end;


function KuljetaPt(OWhat : TControl; f:TWinControl;ToPt:TPoint;
                   arrive:TLiikkuvaMeth;data : integer; v:integer) : boolean;
var liikkuva : TLiikkuva;
begin
  Result := false;
  if not InitLiikkuva(Liikkuva,f,arrive,v) then exit;
  Result := liikkuva.KuljetaPt(OWhat,f,ToPt,data);
end;

function Kuljeta(OWhat,OTo : TControl; arrive:TLiikkuvaMeth; data:integer; v:integer) : bool
begin
  Result := KuljetaCtrlPt(OWhat,OTo,FOrigo,arrive,data,v);
```



```
end;

function KuljetaCtrlPt(OWhat,OTo : TControl; ToPt:TPoint;
                       arrive:TLiikkuvaMeth;data : integer; v:integer) : boolean;
var liikkuva : TLiikkuva;
begin
  Result := false;
  if not InitLiikkuva(Liikkuva,OTo.Owner,arrive,v) then exit;
  Result := liikkuva.KuljetaCtrlPt(OWhat,OTo,ToPt,data);
end;

procedure Register;
begin
  RegisterComponents('GKO', [TLiikkuva]);
end;

end.
```

```
unit raahattava;
{
  Olio, joka on tehty raahaamaan muita olioita paikasta toiseen.
  Vesa Lappalainen 17.01.1998
  Muutoksia:
  7.10.1999/vl
   + oma tappaminen viestillä WM_CLOSE Delphi 5:sta varten
  3.9.2001/vl
   + VCL/CLX-kääntäminen (määrittele vakio CLX)

  Liikuttaa olion olion nykyisestä paikasta hiiren liikkeiden
  mukaan.
  Saadaan tapahtuma jokaisesta liikahduksesta (OnMove) sekä perille
  saapumisesta (OnArrive).

  Jos irtipäästöpaikalla on kontrolli, jossa on OnDragGdrop-tapahtuman
  käsittelijä, niin tätä kutsutaan Arrive-tapahtuman jälkeen.
  Jos joko Arrive tai OnDragDrop muuttaa olion Parenttia, niin
  raahattava ei enää koske olioon.  Jos olion Parenttia ei muuteta,
  laittaa raahattava olion parentiksi sen WinControllin, jonka päälle
  olio pudotettiin, tai jos WinControllia ei löydy, niin olio palautetaan
  alkuperäiseen paikkaansa ja alkuperäinen Parent asetetaan takaisin.

  Arrrive-tapahtuman parametrit:
    TRaahaavaMeth = procedure (o,wnd:TControl;data:integer) of object;
      o      - olio, joka saapui raahattavan mukana
      wnd    - kontrolli, jonka päälle raahattiin
      data   - vapaamuotoinen kutsussa annettu data

  Käyttöä helpottamaan on tehty kolme funktiota, jotka luovat raahattavan
  olio, siirtävät siirrettävän olion raahattavan sisälle ja ilmoittavat
  lopuksi kun siirto on valmis.

   function Raahaa(OWhat : TControl; arrive:TRaahaavaMeth;
                 data:integer) : boolean;
   function RaahaaXY(OWhat : TControl; x,y:integer; arrive:TRaahaavaMeth;
                 data:integer) : boolean;
   function RaahaaPt(OWhat : TControl; dp:TPoint; arrive:TRaahaavaMeth;
                 data:integer) : boolean;
  Parametrit:
    OWhat    - siirrettävä olio
    f        - ikkuna jonka sisällä siirto tehdään
    ToPt     - raahattavan sisällä, yleensä tähän kannattaa laittaa
               MouseDown-tapahtumasta saatu hiiren paikka
    arrive   - metodi, joka käsittelee valmiin siirron
    data     - vapaasti käytettävä sana, joka saadaan käyttöön siirron loputtua

}

interface

uses
  SysUtils, Classes,
{$ifdef CLX}
  Qt,              QGraphics, QControls, QForms, QDialogs,QExtCtrls,
{$else}
  Messages,Windows,Graphics,   Controls,  Forms,  Dialogs, ExtCtrls,
{$endif}
  Types;
```



```
type
  TRaahaavaMeth = procedure (o,wnd:TControl;data:integer) of object;
  TRaahaavaMoveMeth = procedure (o:TControl;pt:TPoint;data:integer) of object;

  TDragControl = class(TControl)  // Hämäystä jotta saadaan OnMouse??? käyttöön
  published
  end;

  TRaahaava = class(TWinControl)
  private
    FOnArrive : TRaahaavaMeth;
    FOnMove   : TRaahaavaMoveMeth;
    FObject   : TDragControl;
    FDp       : TPoint;
    FFrom     : TPoint;
    FData     : Integer;
    FOrigo    : TPoint;

    FOldMove  : TMouseMoveEvent;
    FOldUp    : TMouseEvent;
    FOldParent: TWinControl;
    FMoving   : Boolean;
    FAreaOver : TDragControl;

    FAutoDestroy : boolean;
    procedure ObjMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer);
    procedure ObjMouseUp(Sender: TObject; Button: TMouseButton;
              Shift: TShiftState; X, Y: Integer);
    procedure PaintBox1Paint(Sender: TObject);
    procedure CreateDot;
  protected
  public
    constructor Create(AOwner:TComponent);                  override;
    destructor  Destroy;                                    override;
    function KuljetaPt(OWhat : TControl; f:TWinControl;ToPt:TPoint;
                       data : integer) : boolean; virtual;
    function Kuljeta(OWhat : TControl; dp:TPoint; data:integer):boolean; virtual
    function Moving : boolean;                              virtual;
{$ifdef CLX}
{$else}
//    procedure PaintWindow(DC: HDC); override;
    procedure WmClose(var Message: TWMClose); message WM_CLOSE;
{$endif}
  published
    property OnArrive : TRaahaavaMeth read FOnArrive write FOnArrive;
    property OnMove   : TRaahaavaMoveMeth read FOnMove write FOnMove;
  end;



function Raahaa(OWhat : TControl; arrive:TRaahaavaMeth;
               data:integer) : boolean;
function RaahaaXY(OWhat : TControl; x,y:integer; arrive:TRaahaavaMeth;
               data:integer) : boolean;
function RaahaaPt(OWhat : TControl; dp:TPoint; arrive:TRaahaavaMeth;
               data:integer) : boolean;

procedure Register;

implementation
```

```
constructor TRaahaava.Create(AOwner:TComponent);
begin
  inherited;
//  BevelInner := bvNone;
//  BevelOuter := bvNone;
//  ParentColor := true;
  FAutoDestroy := false;
  FOnMove := nil;
  FOnArrive := nil;
  FOrigo.x := 0; FOrigo.y := 0;
//  ControlStyle := ControlStyle + [csCaptureMouse];
end;

destructor  TRaahaava.Destroy;
begin
  inherited;
end;

function TRaahaava.Moving : boolean;
begin
  Result := FMoving;
end;


procedure TRaahaava.ObjMouseMove(Sender: TObject; Shift: TShiftState; X, Y: Inte
begin
  SetBounds(x - FDp.x,y - Fdp.y,Width,Height);
  if ( sender <> FObject ) then exit;
//  Left := Left + x - FDp.x;
//  Top :=  Top  + y - Fdp.y;
end;

function FindControlAtPos(w:TWinControl;const Pos: TPoint;
                          AllowDisabled: Boolean; nc:TControl): TControl;
var
  I: Integer;
  P: TPoint;
begin
  with w do
    for I := ControlCount - 1 downto 0 do
    begin
      Result := Controls[I];
      if ( Result <> nc ) then with Result do
      begin
        P := Point(Pos.X - Left, Pos.Y - Top);
        if PtInRect(ClientRect, P) and
          ((csDesigning in ComponentState) and (Visible or
          not (csNoDesignVisible in ControlStyle)) or
          (Visible and (Enabled or AllowDisabled)
{$ifdef CLX}
{$else}
  and  (Perform(CM_HITTEST, 0, Longint(PointToSmallPoint(P))) <> 0)
{$endif}
          )) then
          Exit;
      end;
    end;
  Result := nil;
end;

procedure TRaahaava.ObjMouseUp(Sender: TObject; Button: TMouseButton;
```



```
                    Shift: TShiftState; X, Y: Integer);
var wnd,ctl : TControl; dwnd : TDragControl;
    parw : TWinControl;
    dpt,pt : TPoint;
begin
//  FObject.Parent := nil;
  FAreaOver.OnMouseMove := FOldMove;
  FAreaOver.OnMouseUp := FOldUp;

  SetCaptureControl(nil);

  pt := Parent.ScreenToClient(ClientToScreen(FOrigo));
  inc(pt.x,x); inc(pt.y,y);

  pt.x := X; pt.y := y;
  dpt := pt;

  wnd := FindControlAtPos(Parent,pt,true,self);
  ctl := wnd;
  parw := nil;
  while ( ctl is TWinControl ) do begin
    wnd  := ctl;
    parw := ctl as TWinControl;
    pt := parw.ScreenToClient(FAreaOver.ClientToScreen(dpt));
    ctl  := FindControlAtPos(parw,pt,true,self);
  end;

  if ( ctl <> nil ) then wnd := ctl;

  if ( Assigned(FOnArrive) ) then OnArrive(FObject,wnd,FData);

  if ( wnd <> nil ) then begin
    dwnd := TDragControl(wnd);
    if ( assigned(dwnd.OnDragDrop) ) then
      dwnd.OnDragDrop(wnd,FObject,pt.x,pt.y);
  end;

  if FObject.Parent = self then // Jos vastuuta ei ole otettu
    if parw <> nil then begin // Jos löytyi uusi, joka kelpaa vanhemmaksi
      pt := parw.ScreenToClient(ClientToScreen(FOrigo));
      FObject.Parent := parw;
      FObject.Left   := pt.x;
      FObject.Top    := pt.y;
    end
    else begin               // Muuten alkuperäinen paikka ja isäntä
      FObject.Parent := FOldParent;
      FObject.Left   := FFrom.x;
      FObject.Top    := FFrom.y;
    end;

{$ifdef CLX}
  if ( FAutoDestroy ) then QApplication_postEvent(Application.Handle, QCustomEve
{$else}
  if ( FAutoDestroy ) then PostMessage(Handle,WM_CLOSE,0,0);
{$endif}
end;

procedure TRaahaava.PaintBox1Paint(Sender: TObject);
begin
  (Sender as TPaintBox).Canvas.Brush.Color := clRed;
  (Sender as TPaintBox).Canvas.Ellipse(0,0,4,4);
```

```
end;

procedure TRaahaava.CreateDot;
var t: TPaintBox;
begin
  t := TPaintBox.Create(self);
  t.Parent := self;
  t.OnPaint := PaintBox1Paint;
  t.BringToFront;
  t.Width := 20;
  t.Height := 20;
//  t.Align := alClient;
end;

function TRaahaava.KuljetaPt(OWhat : TControl; f:TWinControl;ToPt:TPoint;
                             data : integer) : boolean;
var pt : TPoint;
begin
  Result := false;
  if ( OWhat = nil ) then exit;
  FAreaOver := TDragControl(f);

  FObject := TDragControl(OWhat);
  if ( Moving ) then exit;
  FDp := ToPt;
  Parent := f;
  Visible := true;
  Color := FObject.Color;
  FData := data;

  FOldMove   := FAreaOver.OnMouseMove;
  FOldUp     := FAreaOver.OnMouseUp;
  FAreaOver.OnMouseMove := ObjMouseMove;
  FAreaOver.OnMouseUp   := ObjMouseUp;

  FFrom.x   := FObject.Left; FFrom.y := FObject.Top;
  FOldParent := FObject.Parent;
//  FObject.OnMouseMove := ObjMouseMove;
//  FObject.OnMouseUp := ObjMouseUp;
  pt := Parent.ScreenToClient(FObject.ClientToScreen(FOrigo));
  FObject.Parent := self;
  FObject.Top := 0;
  FObject.Left := 0;
  FMoving := true;
  Width := FObject.Width;
  Height := FObject.Height;
  Left := pt.x;
  Top := pt.y;

//  OnMouseMove := ObjMouseMove;
//  OnMouseUp := ObjMouseUp;
//  FObject.Visible := false;
//  if ( OWhat is TWinControl ) then
//    ReleaseCapture //((OWhat as TWinControl).Handle);
//  Mouse.Capture := TControl(Owner);
//  SetCaptureControl(TControl(Self));

  CreateDot;
  SetCaptureControl(FAreaOver);

//  SetCaptureControl(FObject);
```



```
end;

function TRaahaava.Kuljeta(OWhat : TControl; dp:TPoint; data : integer) : boolean;
var p : TObject; f:TForm; pt:TPoint;
begin
  Result := false;
  if ( OWhat = nil ) then exit;
  if ( Moving ) then exit;
  p := OWhat;
  while ( p<> nil ) and (not ( p is TForm )) do
    p := (p as TControl).Parent;
  if ( p = nil ) then exit;
  f := p as TForm;

  pt := f.ScreenToClient(OWhat.ClientToScreen(FOrigo));

  Result := KuljetaPt(OWhat,f,dp,data)
end;

function InitRaahaava(var l:TRaahaava;f:TComponent;
                      arrive:TRaahaavaMeth) : boolean;
begin
  Result := false;
  l := TRaahaava.Create(f);
  if ( l = nil ) then exit;
  l.Visible := false;
  l.FAutoDestroy := true;
  l.OnArrive := arrive;
  Result := true;
end;


function RaahaaPt(OWhat : TControl; dp:TPoint; arrive:TRaahaavaMeth;
              data:integer) : boolean;
var Raahaava : TRaahaava;
begin
  Result := false;
  if ( OWhat = nil ) then exit;
  if not InitRaahaava(Raahaava,OWhat.Owner,arrive) then exit;
  Result := Raahaava.Kuljeta(OWhat,dp,data);
end;

function RaahaaXY(OWhat : TControl; x,y:integer; arrive:TRaahaavaMeth;
              data:integer) : boolean;
begin
  Result := RaahaaPt(OWhat,Point(x,y),arrive,data);
end;

function Raahaa(OWhat : TControl; arrive:TRaahaavaMeth;
              data:integer) : boolean;
begin
  Result := RaahaaXY(OWhat,0,0,arrive,data);
end;

procedure Register;
begin
  RegisterComponents('GKO', [TRaahaava]);
end;

(* KYLIX
procedure TRaahaava.PaintWindow(DC: HDC);
```

```
begin
  //
end;
*)

{$ifdef CLX}
{$else}
procedure TRaahaava.WmClose(var Message: TWMClose);
begin
  Free;
end;
{$endif}

end.
```



```
unit Raahausp;
{
  Raahauspakka-komponentti:
  - Pakka, josta voidaan raahata kortteja

  Tekijä:  Vesa Lappalainen
  Milloin: 16.1.1998

  Muutoksia
    +
}

interface

uses
  SysUtils, Classes,
{$ifdef CLX}
  QGraphics, QControls, QForms, QDialogs, QExtCtrls, QStdCtrls,
{$else}
  Graphics, Controls, Forms, Dialogs, ExtCtrls, StdCtrls,
{$endif}
  Kortti,Pakka,APakka;

type
  TRaahausPakka = class(TAPakka)
  private
    procedure DragArrive(o,w:TControl;data:integer);
  protected
  public
    constructor Create(AOwner:TComponent);        override;
    destructor  Destroy;                          override;
  published
    procedure PakkaMouseDown(Sender: TObject; Button: TMouseButton;
            Shift: TShiftState; X, Y: Integer); override;
end;

procedure Register;

implementation
uses raahattava;


constructor TRaahausPakka.Create(AOwner:TComponent);
begin
  Inherited;
  Height   := 100;
  Width    := 73;
end;

destructor  TRaahausPakka.Destroy;
begin
  inherited;
end;


procedure TRaahausPakka.DragArrive(o,w:TControl;data:integer);
var p:TRaahausPakka; k:TKortti;
begin
  if not ( o is TKortti ) then exit;
  k := ( o as TKortti );
  p := nil;
  if ( w is TKortti ) and ( w.Parent is TRaahausPakka )  then p := w.Parent as TRaahausPakka
  if ( w is TRaahausPakka ) then p := ( w as TRaahausPakka );
  if ( p = nil ) then p := self;
  if ( not p.Kelpaako(k) ) then p := self;
  p.Puoli := k.Nakyy;
  k.left := 0; k.top := 0;
  p.lisaa_kortti(k);
end;
```

```
procedure TRaahausPakka.PakkaMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var k:TKortti;
begin
  inherited;
  if ( Button <> mbLeft ) then exit;
  if not ( Sender is TKortti ) then exit;
  k := Sender as TKortti;
  if ( not SaakoOttaa(k) ) then exit;
  RaahaaXY(anna(k),x,y,DragArrive,0);
end;



procedure Register;
begin
  RegisterComponents('GKO', [TRaahausPakka]);
end;

end.
```



```
unit SaantoP;
{ Pakka johon voidaan antaa sääntöjä raahattavan kortin hyväksymiselle:

   Author:  Vesa Lappalainen
   Date:    17.1.1998
   Changes:

   Säännöt
      int eka;         - Jos pakka tyhjä, minkä kortin voi jättää
      int ero;         - Luku, joka lisätään päällimmäisenä olevaan, että
                         saadaan seuraavan sopivan kortin arvo
      int lopetus;     - Pakkaan viimeiseksi sopiva kortti
      int jatko;       - Viimeisen päälle käyvä kortti }

interface

uses
{$ifdef CLX}
  SysUtils, Classes, QGraphics, QControls, QForms, QDialogs,
  QExtCtrls, QStdCtrls,Kortti,Pakka,APakka,Raahausp;
{$else}
  SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, StdCtrls,Kortti,Pakka,APakka,Raahausp;
{$endif}

type
  TSaannot = class(TPersistent)
  private
    FEka       : integer;
    FEro       : integer;
    FLopetus   : integer;
    FJatko     : integer;
  published
    property Eka       : integer read FEka     write FEka;
    property Ero       : integer read FEro     write FEro;
    property Lopetus   : integer read FLopetus write FLopetus;
    property Jatko     : integer read FJatko   write FJatko;
  end;

  TSaantoPakka = class(TRaahausPakka)
    FSaannot : TSaannot;
  public
    constructor Create(AOwner:TComponent);     override;
    destructor  Destroy;                       override;
    function    Kelpaako(k:TKortti):boolean;   override;
  published
    property Saannot : TSaannot read FSaannot write FSaannot;
  end;

const spMikaVaan = -20; spEiMikaan = -30;

procedure Register;

implementation

constructor TSaantoPakka.Create(AOwner:TComponent);
begin
  FSaannot := TSaannot.Create;
  inherited;
end;
```

```
destructor  TSaantoPakka.Destroy;
begin
  FSaannot.Free;
  inherited;
end;

function  TSaantoPakka.Kelpaako(k:TKortti):boolean;
begin
  Result := false;
  if ( Kortteja >= MaxKortteja ) then exit;
  Result := true;
  if ( Kortteja = 0 ) then begin                      // Jos tyhjä pakka
    if ( Saannot.Eka = spMikaVaan ) then exit;        // Ja mikä vaan ekaksi, OK
    Result := k.Arvo = Saannot.Eka;                   // Muuten Eka suhde kortin
    Exit;                                             // arvoon ratkaisee
  end;

  if Saannot.Ero = spMikaVaan then exit;              // On kortteja joten OK
  Result := false;
  if Saannot.Ero = spEiMikaan then exit;              // Ei saa laittaa muita!
  if ( anna_arvo = Saannot.Lopetus ) then begin       // Jos päällim.on lopetus
    if ( k.Arvo <> Saannot.Jatko ) then exit;         // Niin ei kelpaa
    Result := true;                                   // Jollei ole jatko-kortti
    Exit;
  end;
  if ( k.Arvo - anna_arvo <> Saannot.Ero ) then exit; // Jos väärä ero, niin ei
  Result := true;
end;

procedure Register;
begin
  RegisterComponents('GKO', [TSaantoPakka]);
end;

end.
```



```
unit Poyta;
{ Pöytä-komponentti, jonka päällä on pöytäpakkoja. Pöytä osaa selata
   pakkojaan.

  5.11.1997/ Tanja Partanen ja Niina Riikonen

  Muutoksia 7.11.1997/vl
   + poista_merkkinnat nimetty poista_merkinta }

interface

uses
{$ifdef CLX}
  SysUtils, Classes, QGraphics, QControls, QForms, QDialogs,
  QExtCtrls,Kortti,Pakka,Poytap;
{$else}
  SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls,Kortti,Pakka,Poytap;
{$endif}

type TMethPakka  = procedure (p:TPakka) of object;

type
  TPoyta = class(TPanel)
  private
  protected
  public
    constructor Create(AOwner:TComponent);             override;
    procedure ForEveryPakka(proc:TMethPakka);         virtual;
  published
  end;

procedure Register;

implementation

constructor TPoyta.Create(AOwner:TComponent);
begin
  inherited;
  Align    := alRight;
  Caption  := ' ';
end;


procedure TPoyta.ForEveryPakka(proc:TMethPakka);
var i:integer; apu:TControl;
begin
  for i:= 0 to ControlCount-1 do begin
    apu := Controls[i];
    if not( apu is TPakka ) then continue;
    proc(apu as TPakka);
  end;
end;

procedure Register;
begin
  RegisterComponents('GKO', [TPoyta]);
end;

end.
```

```
unit Poytap;
{
  Pöytäpakka-komponentti:
  - Pakka, jolla on myös CheckBox, joten se voidaan merkitä.

  5.11.1997/ Tanja Partanen ja Niina Riikonen

  Muutoksia 7.11.1997/vl
    + anna_kortin OnClickin muuttaminen inherited jälkeen
  Muutoksia 27.8.2001/vl
    + otettu huomioon pakan muutokset, eli jo pakkaa
      voi klikata
    + peritty suoraan raahauspakasta, jolloin ei tarvitse kirjoittaa
      niin paljoa koodia
}

interface

uses
{$ifdef CLX}
  SysUtils, Classes, QGraphics, QControls, QForms, QDialogs,
  QExtCtrls, QStdCtrls,Kortti,Pakka,APakka,Raahausp;
{$else}
  SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, StdCtrls,Kortti,Pakka,APakka,Raahausp;
{$endif}

type
  TPoytap = class(TRaahausPakka)
  private
    FActive      : boolean;
    FCheckBox    : TCheckBox;
  protected
  public
    constructor Create(AOwner:TComponent);    override;
    destructor  Destroy;                      override;
    function    GetChecked:Boolean;             virtual;
    procedure   SetChecked(check:Boolean);      virtual;
    procedure   SetActive(a:Boolean);         virtual;
    function    Kelpaako(k:TKortti):boolean;  override;
    procedure PakkaClick(Sender: TObject); override;
  published
    property  Checked:Boolean read GetChecked write SetChecked;
    property  Active:Boolean  read FActive write SetActive;
end;

procedure Register;

implementation


constructor TPoytaP.Create(AOwner:TComponent);
begin
  Inherited;
  Height   := 100;
  Width    := 73;
  FCheckBox := TCheckBox.Create(self);
  if ( FCheckBox = NIL ) then exit;
  FCheckBox.Parent    := self;
//KYLIX  FCheckBox.Alignment := taRightJustify;
```



```
  FCheckBox.Left      := 4;
  FCheckBox.Top       := Height;
  FCheckBox.Checked   := False;
  Checked             := False;
  Active              := True;
//  OnDragDrop          :
end;

destructor  TPoytap.Destroy;
begin
  FCheckBox.Free;
  inherited;
end;


procedure TPoytap.SetActive(a:Boolean);
begin
  FActive := a;
//  Visible := a and (kortteja > 0);                                          // 
end;

procedure TPoytap.SetChecked(check:Boolean);
begin
  FCheckBox.Checked := check;
end;

function TPoytap.GetChecked:Boolean;
begin
  result := FCheckbox.Checked;
end;

procedure TPoytap.PakkaClick(Sender: TObject);
begin
  inherited;
  Checked := not Checked;
end;

function TPoytap.Kelpaako(k:TKortti):boolean;
begin
  Result := kortteja = 0;
end;



procedure Register;
begin
  RegisterComponents('GKO', [TPoytap]);
end;

end.
```