


TOPOL

GEO-INFORMATIONSSYSTEME

# TopoLeXLite

## Schnelleinstieg - programmieren mit Delphi

© 2002 by TopoL Software Ltd.

Alle Rechte vorbehalten. Ohne ausdrückliche, schriftliche
Genehmigung ist es nicht gestattet,
die Dokumentation oder Teile daraus in irgendeiner Form durch
Fotokopie, Mikrofilm oder ein anderes Verfahren zu
vervielfältigen oder zu verbreiten. Dasselbe gilt für das
Recht der öffentlichen Wiedergabe.

Der Herausgeber macht darauf aufmerksam, daß die genannten
Firmen- und Markenzeichen sowie Produktbezeichnungen
in der Regel marken-, patent-, oder warenzeichenrechtlichem
Schutz unterliegen.

Die Herausgeber übernehmen keine Gewähr für die Funktionsfähigkeit
beschriebener Verfahren, Programme oder Algorithmen.

# Inhalt


**INHALT** ..... **2**

**ÜBER TOPOLEXLITE** ..... **3**

**ÜBER „SCHNELLEINSTIEG – PROGRAMMIEREN MIT DELPHI“** ..... **3**

**TOPOL NT** ..... **4**

MODULARE ARCHITEKTUR ..... 4
STANDARDDATENZUGRIFF ..... 5
OFFENER ENTWICKLUNGSRAHMEN (OPEN DEVELOPMENT FRAMEWORK) ..... 5

**GRUNDLAGEN UND BEGRIFFE** ..... **7**

**SCHEMATISCHE DARSTELLUNG DES GEODATENZUGRIFFS MIT TOPOLEXLITE** ..... **8**

**GEODATENQUELLEN –ZUGRIFF UND ANZEIGE** ..... **9**

BENÖTIGTE UND VERWENDETE UNITS IM INTERFACE-ABSCHNITT ..... 9
TYPDEKLARATION WICHTIGER EIGENSCHAFTEN UND METHODEN ..... 9
BENÖTIGTE UND VERWENDETE UNITS IM IMPLEMENTATION-ABSCHNITT ..... 9
INITIALISIEREN VON WICHTIGEN EINSTELLUNGEN BEI PROGRAMMSTART ..... 9
FREIGABE VON OBJEKTEN BEI PROGRAMMENDE ..... 10
ÖFFNEN EINES TOPOL BLOCKS ..... 10
DATENLAYER IM KARTENFENSTER ANZEIGEN ..... 11
ÖFFNEN EINER RASTERKARTE ..... 12
DATENLAYER AUS KARTENFENSTER ENTFERNEN ..... 13
GEODATENQUELLE SCHLIEßEN ..... 13

**ANZEIGE DES KARTENAUSGABEFENSTERS KONFIGURIEREN** ..... **14**

LAYER EINBLENDEN UND AUSBLENDEN ..... 14
REIHENFOLGE DER LAYER VERÄNDERN ..... 14
KARTENMAßSTAB EINSTELLEN ..... 15
ALLES ANZEIGEN ..... 15
VERGRÖßERN UND VERKLEINERN ..... 15
AUSSCHNITT VERÄNDERN ..... 15
FARBE UND SYMBOL VON LAYERN EINSTELLEN ..... 16

**EREIGNIS-VERARBEITUNG** ..... **17**

**GEO-OBJEKTE LOKALISIEREN** ..... **18**

**GEO-OBJEKTE MARKIEREN** ..... **19**

# Über TopoLeXLite

**TopoLeXLite** ist eine ActiveX – Komponentenbibliothek für die Entwicklung von GIS-Applikationen. TopoLeXLite basiert auf **TopoL NT** (New Technology), der neuen Generation der bewährten TopoL GIS Technologie. Die offene Archtiktur der neuen TopoL-Generation beruht auf einem Komponenten – basierenden Design. **Die TopoL NT- Komponenten** bilden die Basis der neuen TopoL-Produkte und Nachfolger der bewährten Produktfamilie.

Die TopoLeXLite-Komponenten sind speziell für die schnelle und einfache Entwicklung von GIS-Applikationen unter modernen RAD-Systemen wie Delphi konzipiert. TopoLeXLite kapselt die komplexe und mächtige Funktionalität der TopoL NT-Komponenten. Über definierte Schnittstellen der Komponenten kann der Entwickler mit wenig Aufwand auf Geodaten zugreifen und diese visualisieren oder abfragen.

# Über „Schnelleinstieg – programmieren mit Delphi“

Diese Anleitung wendet sich hauptsächlich an Delphi-Programmierer. Der Anleitung liegen entsprechende Delphi-Quelltexte und Beispiele anbei. Vorrausgesetzt wird hier zur Zeit Delphi 5 incl. ADO-Unterstützung (ADO-Express oder die Enterpise–Version von Delphi).

# TopoL NT

Neben dem offenen und erweiterbaren Design stellen OpenGIS-Philosophie und die OGC-(*OpenGIS Consortium*) Spezifikationen die wesentliche Technologiebasis von TopoL NT. Die Spezifikationen des OpenGIS Consortiums für die Entwicklung von offenen und interoperablen raumbezogenen Systemen und Werkzeugen ist Grundlage und Bedingung von TopoL New Technology. Die Vorteile dieser revolutoniären GIS-Technolgie ermöglichen kleine und grosse unternehmensweite raumbezogene Informationssysteme mit unterschiedlichen Datenquellen und verschiedenstenen angespassten Werkzeugen, Komponenten und skalierbaren Benutzerschnittstellen. Zukunftsweisende Unabhängigkeit von properitären Herstellerformaten und Wiederverwendbarkeit und Kombination von unterschiedlichsten Basiskomponenten für die schnelle Entwicklung und Anpassung von GIS-Lösungen sind einige der Vorteile der neuen TopoL-Technologie.

Selbstverständlich verfügt TopoL NT nach wie vor auch über Kompatibilität zum TopoL-eigenen Vektor- und Rasterformat. Darüber hinaus bieten sich aber zukünftig die fantastischen Möglichkeiten der OpenGIS – konformen objektbezogenen blattschnittfreien Datenhaltung von Geometrie und Attributdaten in Datenbanksystemen. Über standardisierte Schnittstellen können TopoL NT Produkte direkt auf die Datenquellen ohne umständliche Konvertierung zugreifen. Die Interoperabilitäts-Kriterien der OGC gewährleisten hierbei einen reibungslosen Datentransport über die Systemgrenzen hinweg. Durch die konsequente sehr tiefgreifende Umsetzung der OGC-Spezifikationen in der Entwicklung der TopoL – Komponenten gehören die TopoL – Produkte des neuen Jahrtausend zur international führenden GeoInformationstechnologie.

Dieser Überblick der neuen TopoL NT – Technologie soll dem interessierten TopoL-Anwender und TopoL-Partner sowie Entwicklern einen prinzipiellen Überblick der neuen Technologie und seiner Bestandteile und Möglichkeiten geben. Neben der Erörterung der Architektur von TopoL NT sind auch einige grundlegende Ausführungen über die Basistechnologien Bestandteil dieser Dokumentation. Insbesondere ein kurzer Überblick der OLE-Technolgie. Auf eine Wiedergabe von Basiskonzepten und Spezifikationen des Open GIS Consortium wurde jedoch weitgehend verzichtet.

TopoL NT basiert auf 3 grundlegenden Konzeptionen :

## Modulare Architektur

Die folgenden Bedingungen wurden für das Design von TopoL NT umgesetzt und erfüllt :

- Die Architektur basiert auf dem OGC *Pluggable Computing Model.*
- Modularer Technologieansatz, mit gemeinsamen Komponenten für Zugriff und Manipulation von raumbezogener Information.
- Das Komponenten-Design basiert auf dem Standard Windows Objekt-Modell und seinen Technolgien (OLE, COM , ActiveX)
- Erweiterbarer offener Rahmen, welcher von Entwicklern mit neuen Komponenten erweitert und angepasst werden kann.

## Standarddatenzugriff

Raumbezogene Daten sind verfügbar über Schnittstellen für Standarddatenzugriff :

Vektordatenzugriff basiert auf Datenbankprinzipien, d.h. räumliche Vektordaten sind im Kontext von Datenbankmanagement – Technologien zu betrachten :

- Auf verschiedene Datenquellen wird mit Hilfe von Schnittstellen zugegriffen, der OpenGIS Spezifikation für OLE/COM entsprechend, die z.B. die Integration von räumlichen Daten in den Kontext von grösseren unternehmensweiten Informationssystemen ermöglichen.
- Durch die Standard OLE DB Technologie wird der Zugriff zu unterschiedlichen Informationsquellen ohne Datenumwandlung gewährleistet.
- Dieser Ansatz unterstützt Speicherung und Verwaltung von räumlichen Daten in Standard Datenbank Managenment Systemen (relational oder objektorientiert), die mit Hilfe von Standardanfragesprachen (z.B. SQL92) zugänglich sind.

Rasterdaten sind zur Zeit mit Hilfe von spezifischen Schnittstellen, Rasterdaten - Provider für verschiedene Formate verfügbar :

- Auf verschiedene Rasterdaten wird mit Hilfe von Schnittstellen zugegriffen, die der TopoL Rasterschnittstellen - Spezifikation entsprechen, welches auch die Integration von anderen Formaten ermöglicht.

TopoL NT beinhaltet Standarddaten – Provider für unterschiedliche Datenformate :

- Der TopoL Vektordaten – Provider macht das properitäre TopoL-Format verfügbar über die Standard OLE DB – Schnittstellen. ArcView Shapefiles und ODBC – konforme Datenbanken werden ebenfalls unterstützt. Weitere Datenprovider werden getestet und nachgerüstet.

Die Komponente für Rasterdaten – Provider unterstützt zur Zeit TIFF, BMP, CIT und das TopoL RAS – Format.

*Hinweis :* Nach der Veröffentlichung der OpenGIS Spezifikation für Rasterdatenzugriff soll diese Technolgie ebenfalls von TopoL NT unterstützt werden (OpenGIS Grid Coverages Specification).

## Offener Entwicklungsrahmen (Open development framework)

TopoL NT bietet für die Entwicklung und Anpassung von GIS – Lösungen umfangreiche Komponenten. Diese Komponenten können in allen modernen Entwicklungsumgebungen welche die OLE – Automatisierung unterstützen verwendet werden : Delphi, Borland C++ Builder, Visual Basic, Visual Basic for Applications (VBA), Visual C++, PowerBuilder etc.
Die TopoL NT – Komponenten können in folgende Gruppen/Bereiche oder Schichten unterteilt werden :

- ***Provider für raumbezogenen Datenzugriff (Spatial Data access providers)***

  Daten-Provider ermöglichen anderen Komponenten/Schichten, auf räumliche Daten über Standardschnittstellen zuzugreifen. Diese Schicht kapselt und verbirgt die Komplexität des Zugriffs auf unterschiedliche Formate und properitäre Datenbanken für die Service Module und Applikationen.

- ***GIS Service-Module (GIS service modules)***

  Diese Module verarbeiten die von den Providern gelieferten Daten und liefern GIS-Dienste (z.B. Transformation, Statistiken, Abfragen, räumliche Operationen). Diese Schicht kann leicht mit Unterstützung für andere Dienste oder spezielle Aufgaben erweitert werden.

- ***Präsentations-Module (Data presentation modules)***

  Visualisierungsmodule ermöglichen, die von den Providern gelieferten oder von den Service - Modulen transformierten oder manipulierten Daten (z.B. ausgewählte Daten) zu präsentieren oder zu drucken. Diese Schicht schliesst auch Methoden für die Definition von Darstellungsmerkmalen einschließlich Farbdefinition und Symbolbibliotheken ein. Die offene Natur der Visualisierungs-Komponenten erlaubt das Hinzuzufügen von neuen Merkmalen und Definitionsmethoden und Service-Modulen.

- ***Benutzerschnittstelle (User Inteface controls)***

  Die Benutzerschnittstelle ermöglicht die Präsentation der Daten (mit Nutzung der Präsentations-Module) und bietet eine Schnittstelle zur Eingabe und Änderung von Parametern (z.B. Asuwahl Koordinatensystem, Farbkorrektur, Druckparameter etc.).

- ***Applikationsrahmen-Module (Application framework modules)***

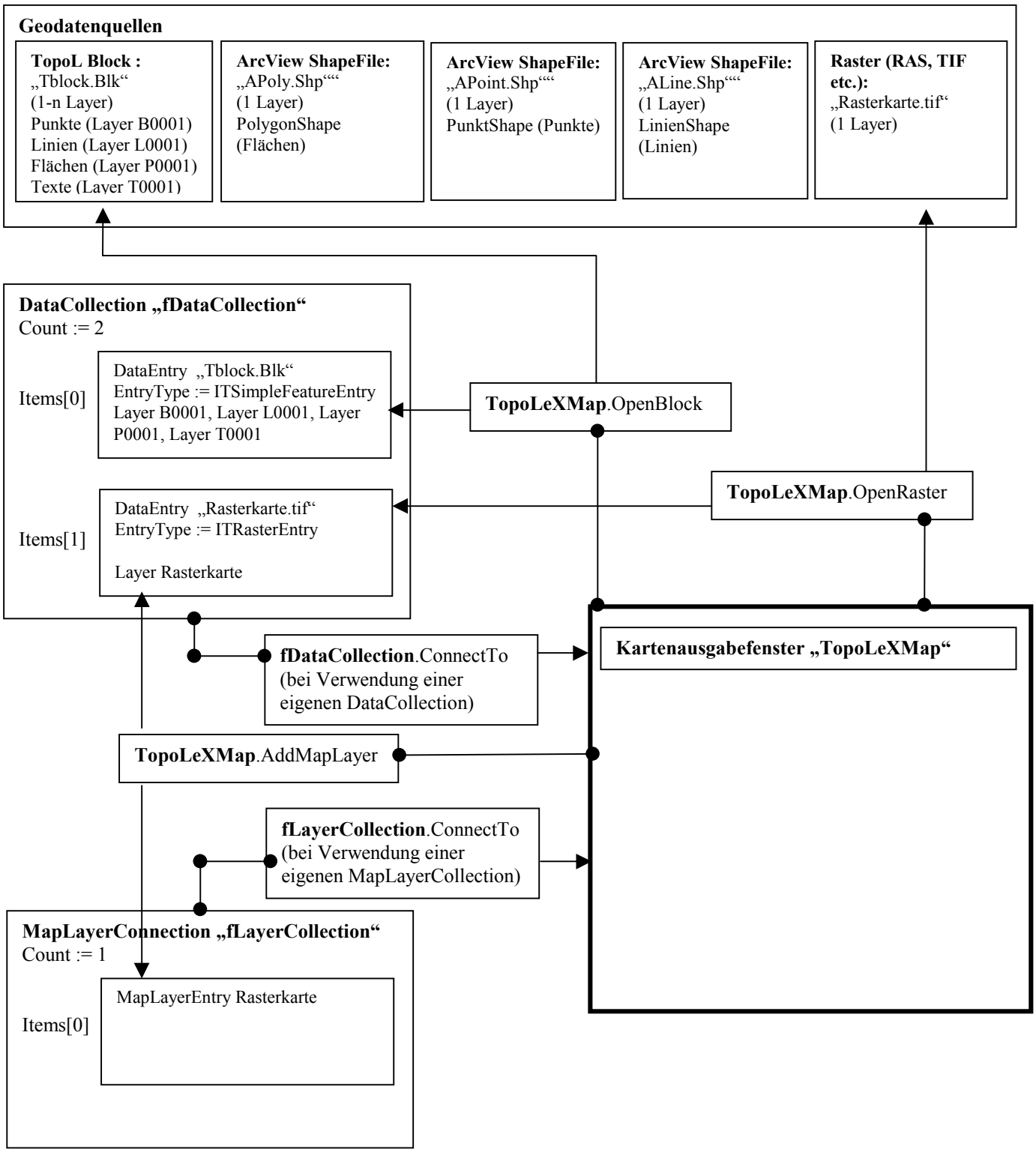
  Diese Schicht beinhaltet das Design und den Aufbau der grafischen Benutzeroberfläche : Menüs, Toolbar – Definition, Kommandodefinition.

Die offene Natur der TopoL NT - Architektur erlaubt, seine Fähigkeiten mit Hilfe von externen Komponenten zu ergänzen, die seine Bestandteile und Methoden entweder erweitern oder ersetzen. Es erlaubt auch Mechanismen für die benutzerdefinierte Anpassung von bereits fertig entwickelten Produkten.

# Grundlagen und Begriffe

| | |
|---|---|
| **Vektordaten**<br><br>Beispiele:<br>• Block (BLK) in TopoL<br>• ShapeFile (SHP) in ArcView<br><br>*z.B: C:\Daten\Flurstücke.Blk* | Geodatenquellen aus verschiedenen Geo-Informationssystemen. Vektordaten beinhalten Punkte, linienförmige Verbindungen dieser Punkte, sowie Flächen, die aus umgebenden Polygonzügen bestehen. Ein Flurstück besteht somit aus den Eckpunktkoordinaten des umgebenden Polygons. Diese Eckpunkte werden in TopoL als Scheitelpunkte bezeichnet. |
| **Rasterdaten**<br><br>Beispiele:<br>• Gescannte TopKarte (TIF oder RAS)<br>• Gescanntes Luftbild (TIF oder RAS)<br><br>*z.B: C:\Daten\TopKarte.Tif* | Rasterdaten oder Pixeldaten haben eine zeilen- und spaltenweise Matrix aus einzelnen Bildpunkten (= *Pixel*n) zur Grundlage. In TopoL sind Rasterdaten zum größten Teil gescannte Karten und Bilder. In Rasterdaten gibt es keine Unterscheidung zwischen Punkten, Linien oder Flächen, es gibt lediglich verschiedene Werte der einzelnen Bildpunkte (verschiedene Grautöne oder Farbwerte, bzw. die Information schwarz oder weiß bei Binärrastern). |
| **Sachdaten** | Unter Sachdaten werden alle alphanumerischen Informationen verstanden, die nichtgeometrische Elemente wie Texte, Zahlen, Namen und Eigenschaften repräsentieren. Sachdaten werden in TopoL als Datenbank- Attribute oder auch als beschreibende Attribute bezeichnet. Diese können in TopoL mit den verschiedenen Vektorobjekten (Punkt, Linie, Fläche) verknüpft werden. |
| **Geodatenquelle** | Vektor – oder Rasterdaten aus verschiedenen GeoInformationssystemen. Eine Geodatenquelle kann aus einem oder aber auch aus mehreren Datenebenen (Layern) bestehen. |
| **TopoLeXLite ActiveX**<br><br>**Kartenausgabefenster** | Hierbei handelt es sich um die visuelle ActiveX-Komponente der TopoLeXLite-Entwicklungsumgebung. Diese Komponente dient zur Darstellung und Abfrage der Geodaten. Diese Komponente stellt in Ihrer Form ein Kartenausgabefenster dar. Sie können 1 oder mehrere Kartenausgabefenster in Ihrer Applikation verwenden. In Delphi steht Ihnen diese Komponente auf der Paletten-Seite ActiveX nach der Installation des Delhi Packages oder dem Import der Typbibliotheken zur Verfügung. |
| **DataCollection** | Eine Liste von Dateneinträgen (DataEntry), wobei jeder Eintrag einer Geodatenquelle entspricht. Das TopoLeXLite ActiveX erzeugt automatisch eine eigene DataCollection. Es ist aber möglich eine eigene DataCollection zu erzeugen und diese über die Delphi-Wrapper Methode ConnectTo mit dem TopoLeXLite ActiveX zu verbinden. Dies hat den Vorteil das man auf Events der DataCollection reagieren kann. Man kann für ein Kartenausgabefenster nur eine DataCollection verwalten. |
| **DataEntry** | Eigenschaft und Eintrag von DataCollection, indiziert den Namen und den Typ (Raster, Vektor etc.) eines Dateneintrags in DataCollection. Jeder Eintrag steht für eine Geodatenquelle. |
| **MapLayerCollection** | Eine Liste von Kartenlayer-Einträgen (MapLayerEntry) für ein Kartenausgabefenster. Das TopoLeXLite ActiveX erzeugt automatisch eine eigene MapLayerCollection. Es ist aber möglich eine eigene MapLayerCollection zu erzeugen und diese über die Delphi-Wrapper Methode ConnectTo mit dem TopoLeXLite ActiveX zu verbinden. Dies hat den Vorteil das man auf Events der MapLayerCollection reagieren kann. Man kann für ein Kartenausgabefenster nur eine MapLayerCollection verwalten. |
| **MapLayerEntry** | Eigenschaft und Eintrag von MapLayerCollection, steht für jeweils ein Layer einer Geodatenquelle aus der DataCollection. Wobei z.B. TopoL-Blöcke mehrere Layer beinhalten können, während Raster und ArcView ShapeFiles nur einen Layer darstellen. |

# Schematische Darstellung des Geodatenzugriffs mit TopoLeXLite



# Geodatenquellen –Zugriff und Anzeige

In diesem Beispiel wird die **DataCollection** für die Aufnahme von Geodatenquellen, sowie die **MapLayerCollection** für die Kartenlayer nicht automatisch **vom TopoLeXLite ActiveX -Kartenausgabefenster** erzeugt. Dies wird durch den folgenden Programmcode erledigt und über die Delphi Wrapper-Methode **ConnectTo** mit dem TopoLeXLite ActiveX verbunden. Dies hat den Vorteil das man auf Events der DataCollection und MapLayerCollection reagieren kann (z.B. beim Einfügen von Layern). Man kann jedoch auch darauf verzichten und die Collections automatisch vom Kartenausgabefenster erzeugen und verwalten laßen.

## Benötigte und verwendete Units im Interface-Abschnitt

```
uses Classes, Forms, Controls, StdCtrls, Buttons, CheckLst, ComCtrls,
     OleCtrls, Menus, TLDataMgr2_TLB, TLViewMgr2_TLB, TopoLeXLT_TLB;
```

## Typdeklaration wichtiger Eigenschaften und Methoden

```
type
  TAppForm = class(TForm)
    TopoLeXMap: TTopoLeXLite;                  //Kartenausgabefenster
    lbData: TListBox;                          //User-Listbox für Geodatenquellen
    clbView: TCheckListBox;                    //User-Checklistbox für Kartenlayer
    procedure FormCreate(Sender: TObject);     //Initialisieren von wichtigen Einstellungen
    procedure FormDestroy(Sender: TObject);     //Freigabe von Objekten und Speicher
  private
    DataCounter: integer;                      //Zähler für geöffnete Geodatenquelle
    ViewCounter: integer;                      //Zähler für Layer eines Kartenfensters
    SampleData: string;
    fDataCollection: TTDataCollection;         //Eigene Liste der Geodatenquellen
    fLayerCollection: TTMapLayerCollection;    //Eigene Liste der Karten-Layer
    procedure DataInsert(Sender: TObject; Index: Integer; var Item: OleVariant); // für „Einfügen von Daten“
    procedure DataRemove(Sender: TObject; Index: Integer; var Item: OleVariant); // für „Entfernen von Daten“
    procedure LayerInsert(Sender: TObject; Index: Integer; var Item: OleVariant); // für „Einfügen von Layer“
    procedure LayerRemove(Sender: TObject; Index: Integer; var Item: OleVariant); // für „Entfernen von Layer“
  end;
```

## Benötigte und verwendete Units im Implementation-Abschnitt

```
uses SysUtils, Windows, ActiveX, Dialogs, Printers, FileCtrl, ADOInt;
```

## Initialisieren von wichtigen Einstellungen bei Programmstart

```
procedure TAppForm.FormCreate(Sender: TObject);
begin
  DataCounter := 0;          //Zähler für geöffnete Geodatenquellen auf 0 setzen
  ViewCounter := 0;          //Zähler für Kartenlayer auf 0 setzen

  TopoLeXMap.Cursor := 2;          //Cursor für Kartenausgabefenster festlegen

  fDataCollection := TTDataCollection.Create(nil);                //Datenliste initialisieren
  fDataCollection.ConnectTo(TopoLeXMap.DataCollection);  //Datenliste verbinden mit Kartenfenster
  fDataCollection.OnInsert := DataInsert;                         //Methode für das Ereignis „Einfügen von Daten“
  fDataCollection.OnRemove := DataRemove;                         //Methode für das Ereignis „Entfernen von Daten“
```

```
  fLayerCollection := TTMapLayerCollection.Create(nil);        //Layerliste initialisieren
  fLayerCollection.ConnectTo(TopoLeXMap.MapLayerCollection);      //Layerliste verbinden mit Kartenfenster
  fLayerCollection.OnInsert := LayerInsert;                //Methode für das Ereignis „Layer  einfügen“
  fLayerCollection.OnRemove := LayerRemove;         // Methode für das Ereignis „Layer  entfernen“

end;
```

## Freigabe von Objekten bei Programmende

```
procedure TAppForm.FormDestroy(Sender: TObject);
begin
  TopoLeXMap.ClearHighlightObjects;                //Besonders dargestellte Objekte entfernen
  TopoLeXMap.MapLayerCollection.ClearAll;          //Layerliste löschen
  TopoLeXMap.DataCollection.ClearAll;              //Geodatenliste löschen
  fDataCollection.Free;                 //DataCollection frei geben
  fLayerCollection.Free;                //LayerCollection frei geben
end;
```

## Öffnen eines TopoL Blocks

Das nachfolgende Quelltextbeispiel öffnet einen TopoL-Block. Die Aktion wird ausgelöst durch Das Anklicken eines Buttons.

```
procedure TAppForm.btBlockClick(Sender: TObject);
var Default, Path: string;
    Entry: ITDataEntry; //lokale Variable für den DataEntry-Eintrag
begin
  //Benutzer gibt den Blockpfad an
  Path := InputBox('Open block', 'Block path:', Default);
  if DirectoryExists(Path)
  then begin
    // der Zähler für die Liste der geöffneten Geodatenquellen wird erhöht
    inc(DataCounter);
    // Öffnen des Blocks über die Methode des Kartenausgabefensters
    Entry := TopoLeXMap.OpenBlock('DATA'+IntToStr(DataCounter), Path);
  end;
end;
```

Die Methode **OpenBlock** fügt der **DataCollection** automatisch einen neuen Eintrag **DataEntry** hinzu. Gleichzeitig wird das **OnInsert-Event** ausgelöst für das Einfügen von Geodaten in die **DataCollection**. Die in **FormCreate** zugeordente Methode wird automatisch ausgeführt und hier wird der geöffnete Block in die Listbox eingetragen:

```
procedure TAppForm.DataInsert(Sender: TObject; Index: Integer; var Item: OleVariant);
var Entry: ITDataEntry;   //lokale Variable für den DataEntry-Eintrag
begin
  //Item enthält eine Referenz auf den DataEntry-Eintrag der DataCollection
  Entry := IDispatch(Item) as ITDataEntry;
   //  Die Eigenschaft Beschreibung von DataEntry wird in die Listbox eingetragen
  // zusätzlich wird ein Pointer als Referenz für die Geodatenquelle angelegt
  // für spätere Verwendung
  lbData.ItemIndex := lbData.Items.AddObject(Entry.Description, pointer(Entry));
end;
```

## Datenlayer im Kartenfenster anzeigen

Nachdem die **Geodatenquelle** geöffnet wurde und ein Eintrag in die **DataCollection** und die **Geodaten-Listbox** für den User erfolgt ist, sollen der oder die **Layer** der Datenquelle im **Kartenfenster** angezeigt werden und in der **Checklistbox** für den User dargestellt werden. Die Aktion wird ausgelöst Durch Anklicken eines Buttons. Hierbei wird/werden die/der **Layer** des in der **Geodaten-Listbox** ausgewählten Eintrags zur Darstellung herangezogen.

```
procedure TAppForm.btViewDataClick(Sender: TObject);
var Data, Table: ITDataEntry;
    Layer: ITMapLayerEntry;
    Empty: boolean;
    I: integer;
begin
  // Testen ob in der Listbox ein Eintrag selekiert ist
  if lbData.ItemIndex < 0
  then ShowMessage('Please select data in list') //Wenn nicht dann Meldung
  else begin
    Empty := clbView.Items.Count <= 0; //Feststellen ob bereits ein Layer angezeigt
    // Nun die Referenz auf den Dateneintrag DataEntry aus der Listbox holen
    Data := ITDataEntry(pointer(lbData.Items.Objects[lbData.ItemIndex]));
    // Wenn es ein Raster ist -  dann nur einen Layer der Karte hinzufügen
    if Data.EntryType = etRaster
    then begin
      // Den Zähler erhöhen
      inc(ViewCounter);
      // Raster im Kartenfenster anzeigen
      Layer := TopoLeXMap.AddMapLayer('VIEW'+IntToStr(ViewCounter), Data);
    end
    else begin
      // Wenn es sich um TopoL Blöcke handelt – Alle Layer eines Blocks hinzufügen
      if Data.EntryType = etSFDatabase
      then begin
        // Für jeden Layer-Eintrag
        for I := pred((Data as ITSimpleFeatureEntry).Tables.Count) downto 0
        do begin
          // Den Zähler erhöhen
          inc(ViewCounter);
          // Layer hinzufügen
          Table := (Data as ITSimpleFeatureEntry).Tables[I];
          Layer := TopoLeXMap.AddMapLayer('VIEW'+IntToStr(ViewCounter), Table);
        end;
      end;
    end;
    // Alles anzeigen im Kartenfenster nur bei erstem Layer
    if Empty then TopoLeXMap.FitAll;
    // Anzeige neu aufbauen
  TopoLeXMap.Refresh;
  end;
end;
```

Die Methode **AddMapLayer** fügt der **MapLayerCollection** automatisch einen neuen Eintrag **MapLayerEntry** hinzu. Gleichzeitig wird das **OnInsert-Event** ausgelöst für das Einfügen von Layern in die **MapLayerCollection**. Die in **FormCreate** zugeordente Methode wird automatisch ausgeführt und hier wird der dargestellte Layer in die CheckListbox eingetragen:

```
procedure TAppForm.LayerInsert(Sender: TObject; Index: Integer; var Item: OleVariant);
var Entry: ITMapLayerEntry;
    Descript: string;
    Idx: integer;
begin
  // Layer in die Checklistbox eintragen
  //Item enthält eine Referenz auf den MapLayerEntry-Eintrag der MapLayerCollection
  Entry := IDispatch(Item) as ITMapLayerEntry;
  // Entweder die Beschreibung des Rasterlayers übernehmen
  // oder bei Blöcken den Namen eines der möglichen Layer übernehmen
  if Entry.DataEntry.EntryType = etRaster  //Raster
  then Descript := Entry.DataEntry.Description
  else begin //Blöcke etc.
    if Entry.DataEntry.EntryType = etSFTable
    then Descript := ((Entry.DataEntry as ITSimpleFeatureTable).DataEntry as ITDataEntry).Description+'-
'+Entry.DataEntry.Name
    else Descript := 'unknown';
  end;
  // Die Referenz auf dem MapLayer-Eintrag in der Liste speichern für spätere Verwendung
  Idx := clbView.Items.AddObject(Descript, pointer(Entry));
  // Den Listeneintrag auf Checked setzen da der Layer im Kartenfenster sichtbar
  clbView.Checked[Idx] := true;
end;
```

## Öffnen einer Rasterkarte

Das nachfolgende Quelltextbeispiel öffnet Rasterdatei. Die Aktion wird ausgelöst durch Das Anklicken eines Buttons.

```
procedure TAppForm.btRasterClick(Sender: TObject);
var Default, Path: string;
    Entry: ITDataEntry; //lokale Variable für den DataEntry-Eintrag
begin
  //Benutzer gibt den Rasterpfad an
  Path := InputBox('Open raster', 'Raster file:', Default);
  if FileExists(Path)
  then begin
    // der Zähler für die Liste der geöffneten Geodatenquellen wird erhöht
    inc(DataCounter);
    // Öffnen des Rasters über die Methode des Kartenausgabefensters
    Entry := TopoLeXMap.OpenRaster('DATA'+IntToStr(DataCounter), ExpandFileName(Path));
  end;
end;
```

Die Methode **OpenRaster** fügt der **DataCollection** automatisch einen neuen Eintrag **DataEntry** hinzu. Gleichzeitig wird das **OnInsert-Event** ausgelöst für das Einfügen von Geodaten in die **DataCollection**. Die in **FormCreate** zugeordente Methode wird automatisch ausgeführt und hier wird das geöffnete Raster in die Listbox eingetragen. Dieses Verfahren entspricht dem beim Öffnen von Blöcken (siehe oben). Die Anzeige des Rasters greift auf die gleichen Methoden wie zuvor beschrieben zurück. Daher wird hier auf eine erneute Beschreibung verzichtet.

## Datenlayer aus Kartenfenster entfernen

Die Aktion wird ausgelöst Durch Anklicken eines Buttons. Hierbei wird der in der **Checklistbox** selektierte **Layer** aus dem **Kartenfenster** und der Checklistbox entfernt.

```
procedure TAppForm.btViewRemoveClick(Sender: TObject);
begin
  //Überprüfen ob in der Liste ein Eintrag selektiert ist
  if clbView.ItemIndex < 0
  then ShowMessage('Please select layer in list')
  else begin
    // Schließen des Layers über die in der Liste gespeicherten MapLayerEntry-Referenz
TopoLeXMap.RemoveMapLayer(ITMapLayerEntry(pointer(clbView.Items.Objects[clbView.ItemIndex])).Name
);
    // Anzeige im Kartenfenster neu aufbauen
 TopoLeXMap.Refresh;
  end;
end;
```

Die Methode **RemoveMapLayer** entfernt in der **MapLayerCollection** einen Layer-Eintrag. Gleichzeitig wird das **OnRemove-Event** ausgelöst für das Entfernen von Layern aus der **MapLayerCollection**. Die in **FormCreate** zugeordente Methode wird automatisch ausgeführt und hier wird der Layer aus der CheckListbox entfernt:

```
procedure TAppForm.LayerRemove(Sender: TObject; Index: Integer; var Item: OleVariant);
begin
  // Layer aus Checklistbox entfernen – der Index entspricht dem Index des übergebenen Parameter
  clbView.Items.Delete(Index);
end;
```

## Geodatenquelle schließen

Die Aktion wird ausgelöst Durch Anklicken eines Buttons. Hierbei wird die in der **Listbox** selektierte **Geodatenquelle** geschloßen.

```
procedure TAppForm.btCloseDataClick(Sender: TObject);
begin
  // Testen ob Listeneintrag selektiert
  if lbData.ItemIndex < 0
  then ShowMessage('Please select data in list')
  else begin
    // Schließen der Geodatenquelle über die in der Liste gespeicherten DataEntry-Referenz
TopoLeXMap.CloseData(ITDataEntry(pointer(lbData.Items.Objects[lbData.ItemIndex])).Name);
   // Anzeige im Kartenfenster neu aufbauen
    TopoLeXMap.Refresh;
  end;
end;
```

Die Methode **CloseData** entfernt in der **DataCollection** einen Geodateneintrag-Eintrag sowie automatisch alle zugehörigen und eventuell im Kartenfenster sichtbaren Layer. Gleichzeitig wird das **OnRemove-Event** ausgelöst für das Entfernen von Geodaten aus der **DataCollection**. Die in **FormCreate** zugeordente Methode wird automatisch ausgeführt und hier wird der Eintrag aus der Listbox entfernt:

```
procedure TAppForm.DataRemove(Sender: TObject; Index: Integer; var Item: OleVariant);
begin
    // Geodatenquelle aus Listbox entfernen – der Index entspricht dem Index des übergebenen Parameter
  lbData.Items.Delete(Index);
end;
```

# Anzeige des Kartenausgabefensters konfigurieren

Nach dem Öffnen und der Darstellung von **Geodaten** im vorherigen Kapitel wird nun die Manipulation der Anzeige im **Kartenausgabefenster** behandelt.

## Layer einblenden und ausblenden

Über die **Checklistbox** für geöffnete **Layer**, in unsererm Beispiel, soll die **Sichtbarkeit** der Layer im **Kartenausgabefenster** vom User beinflußbar sein. Mit einem Klick auf die Check-Optionsschalter sollen die Layer ein-oder ausgeschaltet werden. Dazu bedienen wir uns des **OnClickCheck-Events** der Checklistbox:

```
procedure TAppForm.clbViewClickCheck(Sender: TObject);
var I: integer;
begin
  // Sichtbarkeit ein/aus wenn der Listeneintrag angehakt/nicht angehakt ist
  for I := 0 to pred(clbView.Items.Count)
 // Es werden alle Einträge untersucht und die Eigenschaft Visible des Layereintrags in der MapLayerCollection
//wird die Checked-Eigenschaft der Listbox zugeordnet – der Index der LayerCollection ist hierbei identisch mit
//dem Index der Checklistbox
  do ITMapLayerEntry(pointer(clbView.Items.Objects[I])).Visible := clbView.Checked[I];
    // Anzeige im Kartenfenster neu aufbauen
    TopoLeXMap.Refresh;
end;
```

## Reihenfolge der Layer verändern

Die Reihenfolge der **Layer** entspricht der Reihenfolge der Listeneinträge in der Benutzer-**Checklistbox**. Über 2 Buttons für die 2 Richtungen „eins nach oben“ und „eins nach unten“ wird sowohl die Reihenfolge in der Checklistbox als auch in der **MapLayerCollection** verändert (nachfolgend das Beispiel für „eins nach unten“):

```
procedure TAppForm.sbDownClick(Sender: TObject);
begin
  // Testen ob Listeneintrag selektiert
  if clbView.ItemIndex < 0
  then ShowMessage('Please select layer in list')
  else begin
    // Testen ob nicht bereits ganz unten in Liste
    if clbView.ItemIndex = 0
    then ShowMessage('Layer is at bottom of map')
    else begin
      // Layer um eine Position verschieben
TopoLeXMap.MoveMapLayer(ITMapLayerEntry(pointer(clbView.Items.Objects[clbView.ItemIndex])).Name,
clbView.ItemIndex-1);
      // Layer in Listbox verschieben
      clbView.Items.Move(clbView.ItemIndex, clbView.ItemIndex-1);
    // Anzeige im Kartenfenster neu aufbauen
    TopoLeXMap.Refresh;
    end;
  end;
end;
```

## Kartenmaßstab einstellen

Über die Eigenschaft **MapScale** (Integer) des **Kartenausgabefensters** kann der Maßstab abgefragt oder gesetzt werden. Zur Aktivierung des neuen Maßstabs muß die Anzeige neu aufgebaut werden :

```
// Maßstab setzen über die Eigenschaft des Kartenausgabefensters
TopoLeXMap.MapScale := Scale;
// Anzeige im Kartenfenster neu aufbauen
TopoLeXMap.Refresh;
```

## Alles anzeigen

Die Methode **FitAll** des *Kartenausgabefensters* bringt alle aktiven und sichtbaren **Layer** in den aktuellen Ausschnitt des Fensters. Zur Aktivierung muß die Anzeige neu aufgebaut werden :

```
// Alles anzeigen
TopoLeXMap.FitAll;
// Anzeige im Kartenfenster neu aufbauen
TopoLeXMap.Refresh;
```

## Vergrößern und Verkleinern

Über die Methode **Zoom**(*zoomfaktor*) und den Integer-Parameter *zoomfaktor* kann die Darstellung im **Kartenausgabefenster** vergößert oder verkleinert werden. Der Faktor setzt dabei die Größe des neuen Ausgabebereichs in Relation zur aktuellen Größe des Darstellungsbereichs. Ein negativer Faktor verkleinert (-2 ) ein positiver Faktor (2) vergrößert. Zur Aktivierung muß die Anzeige neu aufgebaut werden :

```
// 50% Vergrößern
TopoLeXMap.Zoom(2);
// Anzeige im Kartenfenster neu aufbauen
TopoLeXMap.Refresh;
```

## Ausschnitt verändern

Der sichtbare Bereich des **Kartenausgabefenster** ist definiert über die Eckkoordinaten **MinH**, **MaxV**, **MaxH**, **MinV**. Diese stehen über das Kartenausgabefenster-Objekt als Eigenschaft (read only) zur Verfügung. Mit den Methoden **SetViewport** und **SetViewportCentre** kann der Ausschnitt verändert werden. Für die Methode **SetViewportCentre** steht auch ein *ZoomFactor* als Parameter zur Verfügung, wird hier der Wert 1 benutzt wird lediglich der Auschnitt an eine neue Position verschoben. Hierbei werden die Parameter **H** und **V** zum neuen Mittelpunkt des Ausschnitts gemacht. Über das **OnMouseUp**-Ereignis des Kartenfensters kann man hier zum Beipiel die Position eines Mausklicks ermitteln und dies Position über **SetViewportCentre** zum neuen Mittelpunkt machen. Zur Aktivierung muß die Anzeige neu aufgebaut werden :

```
TopoLeXMap.SetViewportCentre(H, V, 1);
// Anzeige im Kartenfenster neu aufbauen
TopoLeXMap.Refresh;
```

## Farbe und Symbol von Layern einstellen

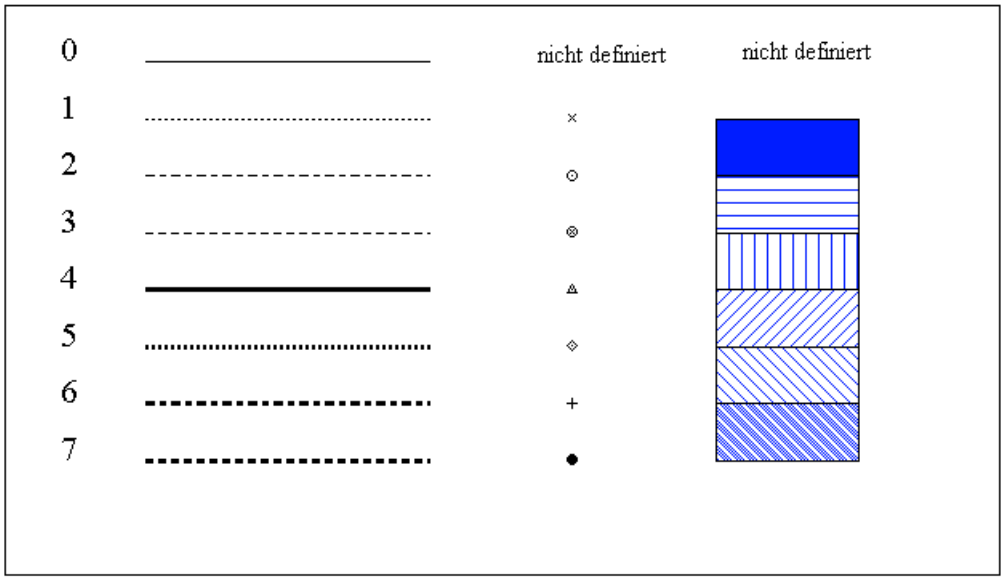
Über die **MapLayerEntry**-Eigenschaften **Color** und **Symbol** kann man allen Objekten (Flächen, Punkte, Linien, Texte oder Pixeln bei Rasterdaten) eine Farbe oder ein Symbol zuordnen. Wichtig zu wissen ist hierbei, das Raster-Layer die Eigenschaft **Symbol** nicht unterstützt. Text-Layer unterstützen zur Zeit nur die Eigenschaft **Color**. Zur Aktivierung muß die Anzeige neu aufgebaut werden :

```
//Ausgewählte Farbe des Standard-Farbdialogs zuordnen
        ITMapLayerEntry(pointer(clbView.Items.Objects[clbView.ItemIndex])).Color := ColorDialog.Color;
//Symbol zuordnen
        ITMapLayerEntry(pointer(clbView.Items.Objects[clbView.ItemIndex])).Symbol := LayerSymbol;
// Anzeige im Kartenfenster neu aufbauen
        TopoLeXMap.Refresh;
```

Die Farb- und Symboleinstellungen werden nicht in den Datenquellen gespeichert. Die internen Farb- und Symboleinstellungen der Datenquellen können jederzeit mit der **MapLayerEntry**-Methode **SetTopoLDefault** hergestellt werden. Zur Aktivierung muß die Anzeige neu aufgebaut werden :

```
//Die gespeicherten Farben und Symbole des Layers wieder herstellen
        ITMapLayerEntry(pointer(clbView.Items.Objects[clbView.ItemIndex])).SetTopoLDefault;
```

Die Symbolnummern für Punkte, Linien und Flächen beziehen sich auf die Symbolnummern der TopoL-Symbolbibliothek. Diese verfügt über eine Reihe von internen Standardsymbolen :




Liniensymbole werden durchnummeriert von 0-n, Punkte von 1-n und Flächen ebenfalls von 1-n. Prinzipiell gibt es auch die Möglichkeit auf benutzerdefinierte Symbolbibliotheken zurückzugreifen. Diese Symbole sind dynamisch, skalierbar und rotierbar.

# Ereignis-Verarbeitung

Das **Kartenausgabefenster** verfügt über eine Reihe von **Ereignissen**, die z.B. auf Bewegung der Maus oder auf Klicken mit der Maus reagieren. Über den Objektinspektor von Delphi kann man hier sehr komfortabel eine Ereignisbehandlung implementieren. Die Ereignis-Routinen des Kartenausgebefensters geben eine Reihe von wichtigen Parameter zurück :

- **X, Y:** Cursor-Position der Maus in Fensterkoordinaten
- **H, V:** Cursor-Position der Maus in geographischen Koordinaten (H:horizontal,V: vertikal)
- **Button, Shift**: Staus der Maustasten und der Shifttaste beim Ereignis.

Folgende Ereignise stehen zur Verfügung :

- **OnMouseMove**(X: Integer; Y: Integer; H: Double; V: Double; Shift: Integer); *Wird ausgelöst wenn die Maus über dem Kartenfenster bewegt wird. Kann verwendet werden für z.B. „Anzeige der Koordinatenposition in einer Statuszeile“.*
- **OnMouseUp**(X: Integer; Y: Integer; H: Double; V: Double; Button: Integer; Shift: Integer); *Wird ausgelöst nachdem die Maustaste gedrückt wurde über dem Kartenfenster.. Kann verwendet werden für z.B. „Abfrage von Objektinformationen“ oder „Verschieben des Ausschnitts“* .
- **OnMouseDown**(X: Integer; Y: Integer; H: Double; V: Double; Button: Integer; Shift: Integer); *Wird ausgelöst wenn die Maustaste gedrückt wird über dem Kartenfenster.*
- **OnPaintProgress**(Status: Integer); *Wird ausgelöst während der Inhalt des Kartenfensters neu gezeichnet wird.*

Das folgende Beispiel zeigt wie der sichtbare Kartenausschnitt nach einem Klick auf das Kartenausgabefenster verschoben wird. Zur Aktivierung muß die Anzeige neu aufgebaut werden :

```
procedure TAppForm.TopoLeXMapMouseUp(Sender: TObject; X, Y: Integer; H,
  V: Double; Button, Shift: Integer);
begin

   //Neuen Mittelpunkt setzen, dieser entspricht den MouseKlick-Koordinaten
   //des OnMouseUp-Events des Kartenausgabefensters
   TopoLeXMap.SetViewportCentre(H, V, 1);
   // Anzeige im Kartenfenster neu aufbauen
   TopoLeXMap.Refresh;
end;
```

# Geo-Objekte lokalisieren

Über die Ereignis-Verarbeitung des **Kartenausgabefensters** können Benutzereingaben der Maus bearbeitet werden. Für die Lokalisierung oder Identifizierung von Geo-Objekten bietet sich dieses Verfahren ebenfalls an. Hierbei klickt der Anwender mit der Maus auf ein Geo-Objekt (Fläche, Linie oder Punkt) und möchte Informationen über das Objekt abfragen. Das Kartenausgabefenster stellt für die Suche nach Objekten die Methode **LocateObjects** zur Verfügung. Diese Methode lokalisiert über ein Koordinatenrechteck Geometrie-Objekte eines **DataEntry**-Eintrags.

```
//Für alle Karten-Layer in der Layer-Checklistbox
  for I := 0 to pred(clbView.Items.Count)
    do begin
    Data := ITMapLayerEntry(pointer(clbView.Items.Objects[I])).DataEntry;
    // Nur wenn der Layer sichtbar ist
    if  ITMapLayerEntry(pointer(clbView.Items.Objects[I])).Visible
    then begin
    // Nur für TopoL-Blöcke
    if Data.EntryType = etSFTable
      then begin
      //Ermitteln des Spaltennamens für das ID-Feld der Geodatentabelle
      RefName := (Data as ITSimpleFeatureTable).RefColumnName;
     // Nur Flächenlayer abfragen
      if (Data as ITSimpleFeatureTable).GeometryType = 3
        then begin
        // Objekte lokalisieren mit der Kartenausgabefenster-Methode
        // Hierbei wird in einem Rechteck um die ermittelte Mausklickposition gesucht
        LocRec := Data.DataSource as _Recordset;
        Located := TopoLeXMap.LocateObjects(Data, LocRec, H-Size, V-Size, H+Size, V+Size);
        // Wenn kein Objekt gefunden wird : VarIsEmpty prüft, ob die angegebene Variante
       //den Wert Unassigned hat
        if not VarIsEmpty(Located)
          then begin
          // Für jedes gefundene Objekt im Array : Es können auch mehrere Flächen sein
          //VarArrayLowBound gibt die Untergrenze einer bestimmten Dimension
         //eines varianten Arrays zurück
          for K := VarArrayLowBound(Located, 1) to VarArrayHighBound(Located, 1)
            do begin
            // Anzeige der Flächen-ID in einer Message-Box
            LocRec.Bookmark := Located[K];
            ShowMessage(RefName+':'+IntToStr(LocRec.Fields[RefName].Value));
            end;
          end;
        end;
      end;
    end;
```

Den vorgestellten Mechanismus kann man auch erweitern um z.B. in einer numerischen nicht-geometrischen Sachdatentabelle einen zugehörigen Datensatz über den ermittelten Objektschlüssel des Geo-Objekts zu lokalisieren :

```
    LocRec.Bookmark := Located[VarArrayLowBound(Located, 1)];
    TableExtern.Locate(DataField, IntToStr(LocRec.Fields[DataField].Value), []);
```

## Geo-Objekte markieren

Im Unterschied zu der Lokalisierung von Geo-Objekten wird die Auswahl und Markierung von geometrischen Objekten des **Kartenausgabefensters** nicht über die Ereignis-Verarbeitung des Kartenfensters eingeleitet. Die Auswahl und Darstellung von Objekten mit benutzerdefinierte Farbe und Symbol wird über die Methode **HighlightObjects** durchgeführt, hierbei werden 1-n Geometrie-Objekte eines **DataEntry**-Eintrags über einen Filter definiert und anschließend mit der spezifizierten Farbe und Symbol dargestellt. Über die Methode **ZoomToHighlightedObjects** kann man anschließend optional das markierte Objekt vergrößert darstellen.

```
// Markieren eines Polygons
  // über die Objekt-ID (CISLO, NUMBER, NUMMER feld in TopoL block)
  // Benutzereingabe der Referenznummer
  RefNumS := InputBox('Objekt markieren', 'ID-Nummer:', '');
  if RefNumS <> ''
  then begin
  // In Zahl konvertieren
  RefNum := StrToInt(RefNumS);
  // Für alle Layer in Checklistbox
  for I := 0 to pred(clbView.Items.Count)
     do begin
     //DataEntry-Eintrag ermitteln über die MapLayerEntry-Referenz der Checklistbox
     Data := ITMapLayerEntry(pointer(clbView.Items.Objects[I])).DataEntry;
     // Nur für TopoL-Blöcke
     if Data.EntryType = etSFTable
     then begin
        // Nur für Flächen-Layer
        if (Data as ITSimpleFeatureTable).GeometryType = 3
        then begin
          Rec := Data.DataSource as _Recordset;
          // // Filter für Geodatentabelle setzen für die ID-Spalte der Flächen
          Rec.Filter := (Data as ITSimpleFeatureTable).RefColumnName + ' = ' + IntToStr(RefNum);
          // Wenn ein Objekt gefunden wurde
          if Rec.RecordCount > 0
          then begin
            // Ein Bookmark-Array erzeugen
            Bookmarks := VarArrayCreate([0, pred(Rec.RecordCount)], varVariant);
            Rec.MoveFirst;
            // Bookmarks im Array speichern
            for K := 0 to pred(Rec.RecordCount)
            do begin
              Bookmarks[K] := Rec.Bookmark;
              Rec.MoveNext;
            end;
            // Objekt markieren
            TopoLeXMap.HighlightObjects(Data, Rec, Bookmarks, $FF, 4);
            // Markiertes Objekt vergrößern, wenn Checkbox markiert
            if cbZoomHighlight.Checked then TopoLeXMap.ZoomToHighlightedObjects(50, 500);
            // Anzeige im Kartenfenster neu aufbauen
            TopoLeXMap.Refresh;
            // Schleife abbrechen nachdem das erste Objekt gefunden wurde
            break;
          end;
        end;
     end;
  end;
 end;
```

Der zuvor abgebildete Code markiert eine Fläche mit einer bestimmten Flächen-ID. Man kann aber auch andere Attribute und Werte benutzen (Filter := 'ATTRIB = ' + YourValue;) oder einen Satz von Attributen (Filter := 'ATTRIB like ' + YourWildcardValue) oder numerische Intervalle. Der Wert des Filters ist wie die WHERE-Klausel in einem SQL SELECT QUERY, wenn auch limitiert. Informationen hierzu gibt es u.a. in der MSDN Library über ADO.

Prinzipiell kann man über die Methode **HighlightObjects** auch alle Objekte eines Layers mit unterschiedlichen Farben und Symbolen versehen.

Die Markierung der Geo-Objekte kann über die Methode **ClearHighlightObjects** wieder rückgängig gemacht werden. Zur Aktivierung muß die Anzeige neu aufgebaut werden :

```
// Markierung löschen
  TopoLeXMap.ClearHighlightObjects;
  // Anzeige im Kartenfenster neu aufbauen
  TopoLeXMap.Refresh;
```