

## Просто и ясно о MapInfo и Delphi I

**Дмитрий Кузан**

дата публикации **27-03-2002 15:46**

Доброе время суток!

Данной статьей я начинаю цикл статей, посвященных возможностям интегрированной картографии MapInfo и возможности встраивания геоинформационной системы в вашу программу.

### **Примечание:**

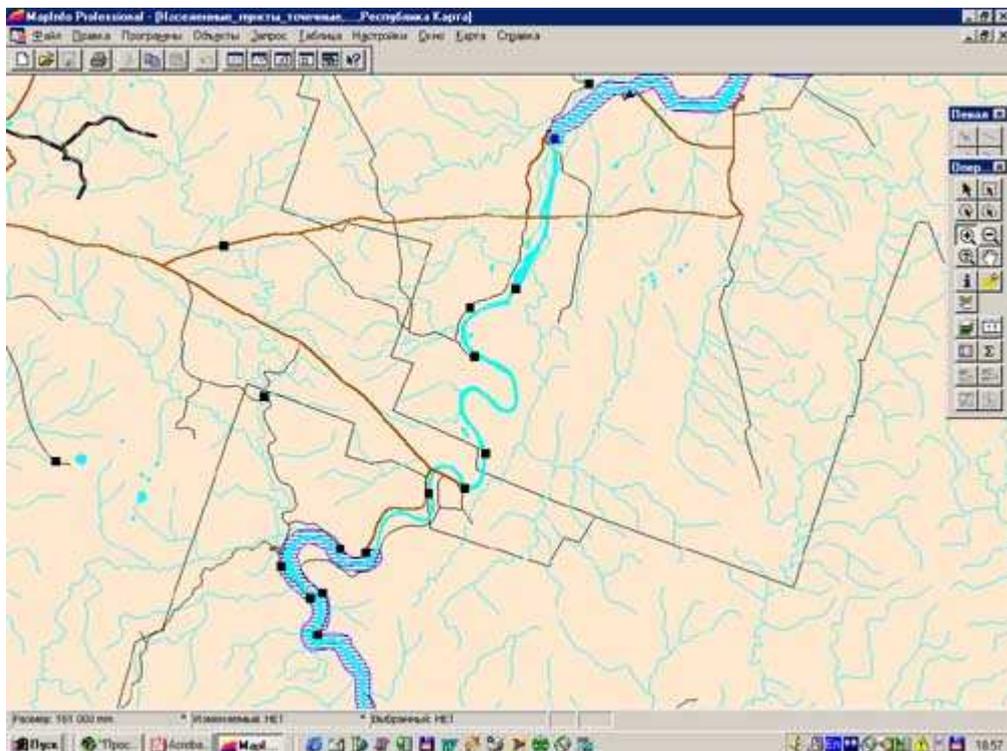
*Все примеры распространяются свободно и разработаны в обучающих целях на Delphi 6. Всю информацию по работе смотрите в прилагаемых исходных кодах.*

Итак, начнем.

### **1. Что такое MapInfo и с чем его едят? Краткое предисловие.**

Сейчас нам доступны огромные объемы информации. Данные хранятся в электронных таблицах, отчетности о торговле и маркетинге. Масса информации о клиентах, магазинах, персонале, оборудовании и ресурсах находится на бумаге и в памяти компьютеров. (...) Почти все эти данные имеют географическую составляющую. По разным оценкам, до 85 процентов всех баз данных содержат какую либо географическую информацию.

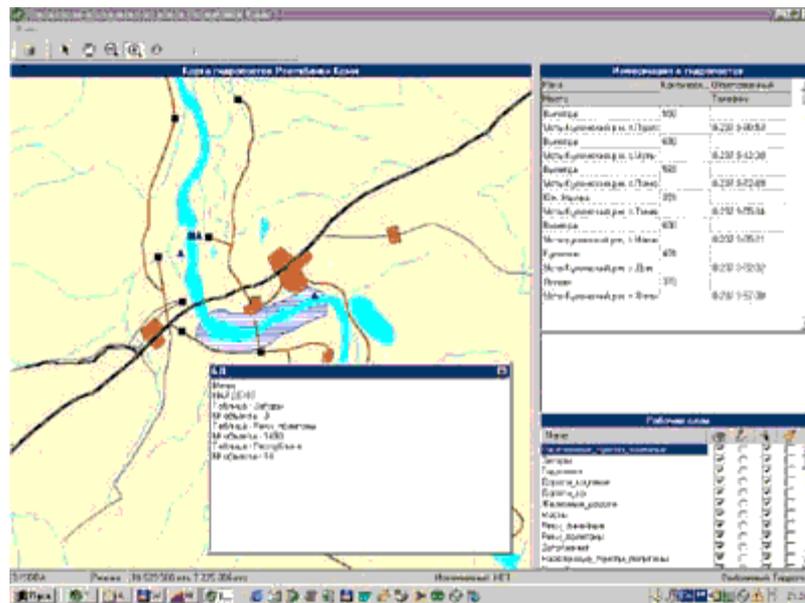
При этой оценке учитывались объемы данных, содержащие адреса, имена городов, названия областей, государств, почтовые индексы и даже номера телефонов, включая коды удаленного доступа и добавочные номера. Настольная картография позволит вашему компьютеру не просто обрабатывать данные, а быстро и наглядно представлять их, используя географические компоненты данных, чтобы вы могли уловить их общий смысл, отражаемый на картах.



Как настольная картография может работать на вас? MapInfo, как средство настольной картографии – это мощное средство анализа данных. Вы можете придать графический вид

статистическим и прочим данным. Вы можете отобразить ваши данные как точки, как тематически выделенные области, круговые и столбчатые графики, районы и т.п. К данным можно применять географические операторы, такие как районирование, комбинация и разрезание объектов и буферизация. Доступ к данным можно оформлять как запросы, в том числе к удаленным базам данных непосредственно из MapInfo. Например: «какой из магазинов ближе к самым крупным клиентам вашей фирмы»? На карте легко увидеть особенности и тенденции, которые практически невозможно выявить в списочно организованных данных. Можно легко вычислить расстояния между клиентами и магазинами; можно увидеть местоположение офиса клиента, потратившего наибольшую сумму за прошлый год; размер символов, отмечающих местоположение магазинов на карте, может зависеть от объема продаж. Все это делает визуализацию ваших данных более наглядной. Итак, краткое предисловие из Руководства Пользователя дает вам общее представление об MapInfo.

## 2. Что такое интегрированная картография и какой нам от нее смысл.



Интегрированная картография позволяет управлять пакетом MapInfo, используя языки программирования, отличные от MapBasic. Например если вам хорошо знакомо программирование на языке Visual Basic или C++ или Delphi (о чем и пойдет речь далее), вы можете включить окно MapInfo в ваше приложение, тем самым обеспечивая интеграцию пакета MapInfo с логикой (бизнес-правилами) вашей программы.

Причем основную работу по поддержанию векторных карт берет на себя MapInfo (MapBasic). Вы можете создавать, назначать обработчики и механизмы взаимодействия, не свойственные MapBasic, а также те механизмы, которые MapBasic не поддерживает напрямую (например, обновление карты по интернету, съем информации с датчиков на территории и обновление на карте и т.п.).

На рис.2 как раз представлен пример интегрированной картографии, встроенной в мой проект на Delphi.

Итак, приступим. В цикле статей будут рассмотрены следующие вещи:

- Соединение и загрузка MapInfo
- Встраивание окна MapInfo и других окон (легенда, информация и т.д.) в программу на Delphi
- Отправка команд MapBasic в пакет MapInfo
- Получение информации от MapInfo посредством функций
- Использование уведомляющих вызовов (CallBack) и подключение их к своей программе
- Создание собственных уведомляющих вызовов
- Переопределение уведомляющих вызовов

- Обработка уведомляющих вызовов
- Создание простейшего компонента (возможно, данная тема будет затронута) для управления MapInfo.
- и многое другое.

### 3. Концепции Интегрированной Картографии

Для создания приложения с Интегрированной Картой вы должны написать программу – но не программу на языке MapBasic. Приложения с Интегрированной Картой могут быть написаны на нескольких языках программирования, среди которых наиболее часто используются C, Visual Basic, Delphi.

В вашей программе должна присутствовать инструкция, запускающая MapInfo в фоновом режиме. Например, в программе вы можете запустить MapInfo вызовом функции CreateObject(). Программа MapInfo запускается в фоновом режиме незаметно для пользователя, не выводя заставку на дисплей. Ваша программа осуществляет управление программой MapInfo, конструируя строки, представляющие операторы языка MapBasic, которые затем передаются в MapInfo посредством механизма управления объектами OLE (OLE Automation) или динамического обмена данными (DDE). MapInfo выполняет эти операторы точно так же, как если бы пользователь вводил их с клавиатуры в окно MapBasic.

#### Примечание:

*Переподчинение окон MapInfo другому приложению не дает программе MapInfo автоматического доступа к данным этого приложения. Для отображения данных приложения в окне MapInfo вы должны предварительно записать эти данные в таблицу MapInfo*

### 4. Системные требования

1. Интегрированная картография требует наличия на компьютере MapInfo версии 4.0 или выше. Вы можете использовать полную версию MapInfo или так называемый исполняемый (Runtime) модуль (усеченная версия MapInfo, поставляемая в качестве основы для специализированных приложений).
2. Вы должны иметь опыт работы с Handle.
3. Ваша программа должна быть способна действовать в качестве контроллера механизма управления объектами OLE (OLE Automation Controller) или клиента динамического обмена данными DDE. Рекомендуется применение OLE контроллера как более быстрого и надежного метода по сравнению с DDE. Его-то мы и будем рассматривать.

### 5. Другие краткие технические замечания

1. Интегрированная картография использует механизм управления OLE, но не использует OLE-внедрение.
2. Интегрированная картография не использует элементы управления VBX или OCX (дело не совсем так – существует OCX модуль MapX для работы с ГИС MapInfo, который не входит в стандартный комплект поставки, но это уже не интегрированная картография и он рассматриваться не будет).
3. Интегрированная картография не предоставляет вам какие-либо заголовочные файлы и библиотеки.
4. Интегрированная картография включает несколько DLL-библиотек, но не предоставляет к ним доступ напрямую.

### 6. Запуск и связывание с сервером MapInfo

Итак, рассмотрим простейший компонент для запуска и управления MapInfo (**TKDMapInfoServer**). Следует заметить, что мной не ставилась цель написания специализированного компонента – я представляю основы.

```
unit KDMapInfoServer;

interface
uses
ComObj, Controls, Variants, ExtCtrls, Windows, Messages, SysUtils, Classes;

const
scMapInfoWindowClass = 'xvt320mditask100';
icWinMapinfo = 1011;
icWinInfoWindowid = 13;

type
TEvalResult = record
AsVariant: OLEVariant;
AsString: String;
AsInteger: Integer;
AsFloat: Extended;
AsBoolean: Boolean;
end;

TKDMapInfoServer = class(TComponent)
private
// Владелец
FOwner : TWinControl;

// OLE сервер
FServer : Variant;
FHandle : THandle;
FActive : Boolean;
FPanel : TPanel;

Connected : Boolean;

MapperID : Cardinal;
MapperNum : Cardinal;

procedure SetActive(const Value: Boolean);
procedure SetPanel(const Value: TPanel);

procedure CreateMapInfoServer;
procedure DestroyMapInfoServer;
{ Private declarations }
protected
{ Protected declarations }
public
{ Public declarations }
constructor Create(AOwner: TComponent); override;
destructor Destroy; override;

// Данная процедура выполняет метод сервера MapInfo - Do
procedure ExecuteCommandMapBasic(Command: String; const Args: array of
const);
// Данная процедура выполняет метод сервера MapInfo - Eval
function Eval(Command: String; const Args: array of const): TEvalResult;
virtual;
procedure WindowMapDef;
procedure OpenMap(Path : String);
published
{ Published declarations }

// Создает соединение с сервером MapInfo
property Active: Boolean read FActive write SetActive;
property PanelMap : TPanel read FPanel write SetPanel;
```

```
end;

procedure Register;

implementation

procedure Register;
begin
RegisterComponents('Kuzan', [TKDMapInfoServer]);
end;

{ TKDMapInfoServer }

constructor TKDMapInfoServer.Create(AOwner: TComponent);
begin
inherited Create(AOwner);
FOwner := AOwner as TWinControl;
FHandle := 0;
FActive := False;
Connected := False;
end;

destructor TKDMapInfoServer.Destroy;
begin
DestroyMapInfoServer;
inherited Destroy;
end;
//-----
procedure TKDMapInfoServer.CreateMapInfoServer;
begin
try
FServer := CreateOleObject('MapInfo.Application');
except
FServer := Unassigned;
end;

// Скрываем панели управления MapInfo
ExecuteCommandMapBasic('Alter ButtonPad ID 4 ToolbarPosition (0, 0) Show
Fixed', []);
ExecuteCommandMapBasic('Alter ButtonPad ID 3 ToolbarPosition (0, 2) Show
Fixed', []);
ExecuteCommandMapBasic('Alter ButtonPad ID 1 ToolbarPosition (1, 0) Show
Fixed', []);
ExecuteCommandMapBasic('Alter ButtonPad ID 2 ToolbarPosition (1, 1) Show
Fixed', []);
// Переопределяем окна
ExecuteCommandMapBasic('Close All', []);
ExecuteCommandMapBasic('Set ProgressBars Off', []);
ExecuteCommandMapBasic('Set Application Window %D', [FOwner.Handle]);
ExecuteCommandMapBasic('Set Window Info Parent %D', [FOwner.Handle]);

FServer.Application.Visible := True;
if IsIconic(FOwner.Handle) then ShowWindow(FOwner.Handle, SW_Restore);
BringWindowToTop(FOwner.Handle);
end;

procedure TKDMapInfoServer.DestroyMapInfoServer;
begin
ExecuteCommandMapBasic('End MapInfo', []);
FServer := Unassigned;
end;

//-----
procedure TKDMapInfoServer.ExecuteCommandMapBasic(Command: String;
```

```
const Args: array of const);
begin
if Connected then
try
FServer.Do(Format(Command, Args));
except
on E: Exception do MessageBox(FOwner.Handle,
PChar(Format('Ошибка выполнения () - %S',
[E.Message])), 'Warning', MB_ICONINFORMATION OR MB_OK);
end;
end;
//-----
function TKDMapInfoServer.Eval(Command: String;
const Args: array of const): TEvalResult;

Function IsInt(Str : String): Boolean;
var
Pos : Integer;
begin
Result := True;
For Pos := 1 To Length(Trim(Str)) do
begin
IF (Str[Pos] <> '0') and (Str[Pos] <> '1') and
(Str[Pos] <> '2') and (Str[Pos] <> '3') and
(Str[Pos] <> '4') and (Str[Pos] <> '5') and
(Str[Pos] <> '6') and (Str[Pos] <> '7') and
(Str[Pos] <> '8') and (Str[Pos] <> '9') and
(Str[Pos] <> '.') Then
Begin
Result := False;
Exit;
end;
end;
end;

var
ds_save: Char;
begin
if Connected then
begin
Result.AsVariant := FServer.Eval(Format(Command, Args));
Result.AsString := Result.AsVariant;
Result.AsBoolean := (Result.AsString = 'T') OR (Result.AsString = 't');

IF IsInt(Result.AsVariant) Then
Begin
try
ds_save := DecimalSeparator;
try
DecimalSeparator := '.';
Result.AsFloat := StrToFloat(Result.AsString); //Result.AsVariant;
finally
DecimalSeparator := ds_save;
end;
except
Result.AsFloat := 0.00;
end;

try
Result.AsInteger := Trunc(Result.AsFloat);
except
Result.AsInteger := 0;
end;
```

```
end
else
Begin
Result.AsInteger := 0;
Result.AsFloat := 0.00;
end;
end;
end;

//-----
procedure TKDMapInfoServer.SetActive(const Value: Boolean);
begin
FActive := Value;

IF FActive then
begin
CreateMapInfoServer;
WindowMapDef;
Connected := True;
end
else
begin
IF Connected then
begin
DestroyMapInfoServer;
Connected := False;
end;
end;
end;

//-----
procedure TKDMapInfoServer.SetPanel(const Value: TPanel);
begin
FPanel := Value;
end;

procedure TKDMapInfoServer.WindowMapDef;
begin
ExecuteCommandMapBasic('Set Next Document Parent %D Style 1',
[FPanel.Handle]);
end;

procedure TKDMapInfoServer.OpenMap(Path: String);
begin
ExecuteCommandMapBasic('Run Application <%S>', [Path]);
MapperID := Eval('WindowInfo(FrontWindow(),%D)', [12]).AsInteger;
with PanelMap do MoveWindow(MapperID, 0, 0, FPanel.ClientWidth,
FPanel.ClientHeight, True);
end;

end.
```

Итак, что мы имеем:

- Мы установили связь с сервером MapInfo.
- Мы узнали, что у сервера MapInfo есть метод Do – он предназначен для отправки команд MapBasic серверу точно так же, как если бы пользователь набирал их в окне MapBasic-а самой программы MapInfo.
- Мы узнали, что у сервера MapInfo есть метод Eval – он предназначен для получения значения функций после отправки команд MapBasic серверу.
- Мы познакомились с командами переопределения направления вывода MapInfo.

Для начала неплохо

## Теперь немного теории.

### Запуск MapInfo

Запуск уникального экземпляра программы MapInfo осуществляется вызовом функции CreateObject() Visual Basic с присваиванием возвращаемого значения объектной переменной. Вы можете декларировать объектную переменную как глобальную; в противном случае объект MapInfo освобождается после выхода из локальной процедуры.

Например:

```
FServer := CreateOleObject('MapInfo.Application');
```

Для подключения к ранее исполнявшемуся экземпляру MapInfo, который не был запущен вызовом функции CreateObject(), используйте функцию GetObject().

```
// Данная реализация оставлена вам, уважаемые читатели, для тренировки  
FServer := GetObject('MapInfo.Application');
```

#### Внимание:

*Если вы работаете с Runtime-версией MapInfo, а не с полной копией, задавайте «MapInfo.Runtime» вместо «MapInfo.Application». Runtime-версия и полная версия могут работать одновременно.*

Функции CreateObject() и GetObject() используют механизм управления объектами OLE (OLE Automation) для связи с MapInfo.

#### Примечание:

*В 32-разрядной версии Windows (Windows95 или Windows NT) можно запускать несколько экземпляров MapInfo. Если вы запустите MapInfo и вслед за этим программу, использующую Интегрированную Картографию и вызывающую CreateObject(), то будут работать два независимых экземпляра MapInfo. Однако в 16-разрядной версии программа, использующая Интегрированную Картографию с запущенным экземпляром MapInfo, работать не сможет.*

### Пересылка команд в программу MapInfo

После запуска программы MapInfo необходимо сконструировать текстовые строки, представляющие операторы языка MapBasic.

Если вы установили связь с MapInfo, используя механизм управления объектами OLE (OLE Automation), передавайте командную строку программе MapInfo методом Do.

Например:

```
FServer.Do('здесь команда MapBasic');
```

#### Примечание:

*В компоненте это реализовано процедурой ExecuteCommandMapBasic, но, в сущности, вызывается FServer.Do.*

При использовании метода Do программа MapInfo исполняет командную строку точно так, как если бы ее ввели в окне команд MapBasic.

#### Примечание:

*Вы можете передать оператор в программу MapInfo, если этот оператор допустим окне MapBasic. Например, вы не можете переслать MapBasic-оператор Dialog, поскольку его использование не разрешено в окне MapBasic.*

Для определения допустимости использования оператора языка MapBasic в окне MapBasic обратитесь к Справочнику MapBasic или откройте Справочную систему: искомая информация находится под заголовком «Предупреждение». Например, в Справке по оператору Dialog дано следующее ограничение: «Вы не можете использовать оператор Dialog в окне исполнения (такие, как For...Next и Goto), не разрешены для исполнения в окне MapBasic».

### Запрос данных от программы Map Info

Для выполнения запроса из вашей программы-клиента значения MapBasic используйте OLE-метод Eval.

Например:

```
MyVar:= FServer.Eval('здесь команда MapBasic');
```

#### Примечание:

*В компоненте это реализовано процедурой Eval, но, в сущности, вызывается FServer.Eval.*

При использовании метода Eval программа MapInfo интерпретирует строку как выражение языка MapBasic, определяет значение выражения и возвращает это значение в виде строки.

#### Замечание:

*Если выражение приводится к логическому значению (тип Logical), MapInfo возвращает односимвольную строку, «Т» или «F» соответственно.*

### Переподчинение окон MapInfo

После запуска MapInfo используйте оператор Set Application Window языка MapBasic для обеспечения перехвата управления вашей программой-клиентом диалоговых окон и сообщений об ошибках программы MapInfo.

Затем, в желаемой точке включения окна MapInfo в ваше приложение передайте MapInfo оператор Set Next Document, за которым следует MapBasic-оператор, создающий окно.

Оператор Set Next Document позволяет вам «переподчинить» окна документов. Синтаксис этого оператора требует указания уникального номера HWND элемента управления в вашей программе. При последующем создании окна-документа MapInfo (с использованием операторов Map, Graph, Browse, Layout или Create Legend) создаваемое окно становится для окна порождающим объектом.

Примеры приведены из компонента, но тоже самое можно выполнить и методом Do непосредственно, но вы это уже, я думаю, поняли:

```
ExecuteCommandMapBasic('Set Application Window %D', [FOwner.Handle]);  
ExecuteCommandMapBasic('Set Window Info Parent %D', [FOwner.Handle]);  
  
ExecuteCommandMapBasic('Set Next Document Parent %D Style 1',  
[FPanel.Handle]);
```

#### Примечание:

*В компоненте это реализовано процедурой WindowMapDef, которая ссылается на панель, заданную свойством PanelMap.*

Для каждого переподчиняемого окна необходимо передать программе MapInfo из вашей программы пару операторов – оператор Set Next Document Parent, а затем оператор, создающий окно. После создания окна вам может понадобиться запросить из MapInfo значение функции WindowID(0) – целочисленный ID-номер окна (Window ID) в MapInfo, так как многие операторы языка MapBasic требуют задания этого номера. Этот запрос выполняется на основе компонента следующим образом:

```
WindowID := Eval('WindowID(%D)',[0]).AsInteger;
```

Заметьте, что даже после переподчинения окна Карты, MapInfo продолжает управлять им. Клиентская программа может не обращать внимания на сообщения о перерисовке, реализацию данной особенности я оставлю на потом.

### Переподчинение окон Легенд, растровых диалогов и других окон MapInfo

Чтобы изменить (переподчинить) данные окна, используется оператор MapBasic Set Window... Parent.

Например, в компоненте переподчинение окна информации реализовано так:

```
ExecuteCommandMapBasic('Set Window Info Parent %D', [FOwner.Handle]);
```

Реализацию переподчинения других окон я оставляю вам, уважаемые читатели.

Заметьте, что способ переподчинения окна Информации другой, чем для окна Карты. В последнем случае не используется предложение Set Next Document. Дело в том, что может существовать несколько окон Карты.

Окна Легенды – особый случай. Обычно существует только одно окно Легенды, так же, как и одно окно Информации. Однако при помощи оператора MapBasic Create Legend вы можете создавать дополнительные окна Легенды.

Для одного окна Легенды используйте оператор MapBasic Window Legend Parent.

Чтобы создать дополнительное окно Легенды, используйте оператор MapBasic Set Next Document и оператор Create Legend. Заметьте, что в этом случае вы создаете Легенду, которая привязана к одному определенному окну Карты или окну Графика. Такое окно Легенды не изменяется, когда другое окно становится активным.

#### Совет:

*Вы можете создать «плавающее» окно Легенды внутри окна Карты. В операторе Set Next Document укажите окно Карты как порождающее окно. Для получения более подробной информации смотрите в документации по MapBasic.*

## Просто и ясно о MapInfo и Delphi II

[Дмитрий Кузан](#)

дата публикации 01-04-2002 13:15

### 1. Использование уведомляющих вызовов (Callbacks) для получения информации из MapInfo – краткий учебный курс.

Вы можете построить ваше приложение так, чтобы MapInfo автоматически посылало информацию вашей клиентской программе. Например, можно сделать так, чтобы всякий раз при открытии и смене диалоговых окон сообщать ID-номер текущего окна.

Такой тип уведомления известен как обратный вызов или уведомление (callback).

Уведомления используются в следующих случаях:

- **Пользователь применяет инструмент в окне.** Например, если пользователь производит перемещение объекта мышкой в окне Карты, MapInfo может вызвать вашу клиентскую программу, чтобы сообщить x- и y- координаты.
- **Пользователь выбирает команду меню.** Например, предположим, что ваше приложение настраивает «быстрое» меню MapInfo (меню, возникающее при нажатии правой кнопки мышки). Когда пользователь выбирает команду из этого меню, MapInfo может вызвать вашу клиентскую программу, чтобы сообщить ей о выборе.
- **Изменяется окно Карты.** Если пользователь изменяет содержание окна Карты (например, добавляя или передвигая слои), MapInfo может послать вашей клиентской программе идентификатор этого окна.
- **Изменяется текст в строке сообщений MapInfo.** Строка состояния MapInfo не появляется автоматически в приложениях Интегрированной Картографии. Если вы хотите, чтобы ваша клиентская программа эмулировала строку состояния MapInfo, то вы должны построить приложение так, чтобы MapInfo сообщало вашей клиентской программе об изменениях текста в строке состояния.

### 2. Требования к функциям уведомления

Программа должна быть способна функционировать, как DDE-сервер или как сервер Автоматизации OLE.

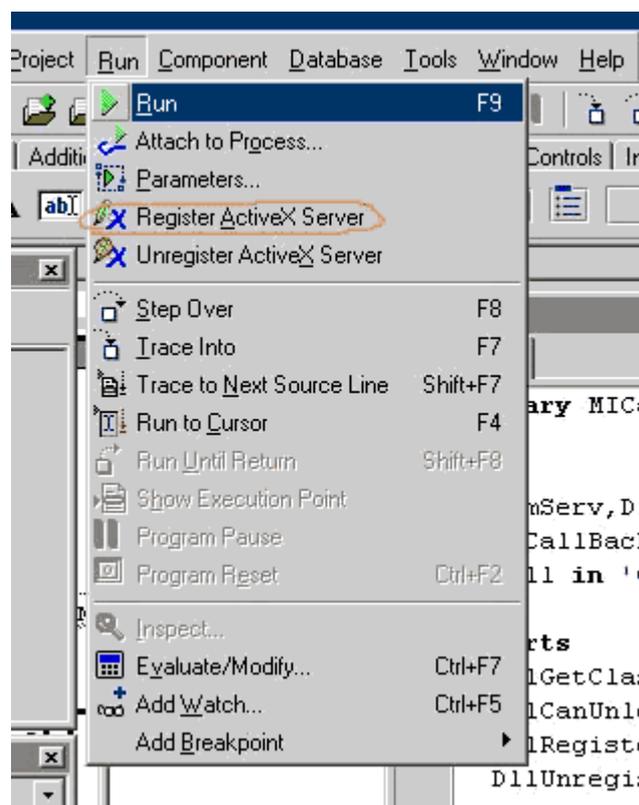
#### Предопределенные процедуры **SetStatusText**, **WindowContentsChanged**.

1. Если вы хотите имитировать строку состояния MapInfo, создайте метод, называемый **SetStatusText**. Определите этот метод так, чтобы у него был один аргумент: строка.
2. метод **WindowContentsChanged**, MapInfo посылает четырехбайтовое целое число (ID окна MapInfo), чтобы указать, какое из окон Карты изменилось. Напишите код, делающий необходимую обработку.

Возможно также и регистрация пользовательских событий, но это отложим пока на третью часть.

### 3. Переинсталляция компонента TKDMapInfoServer.

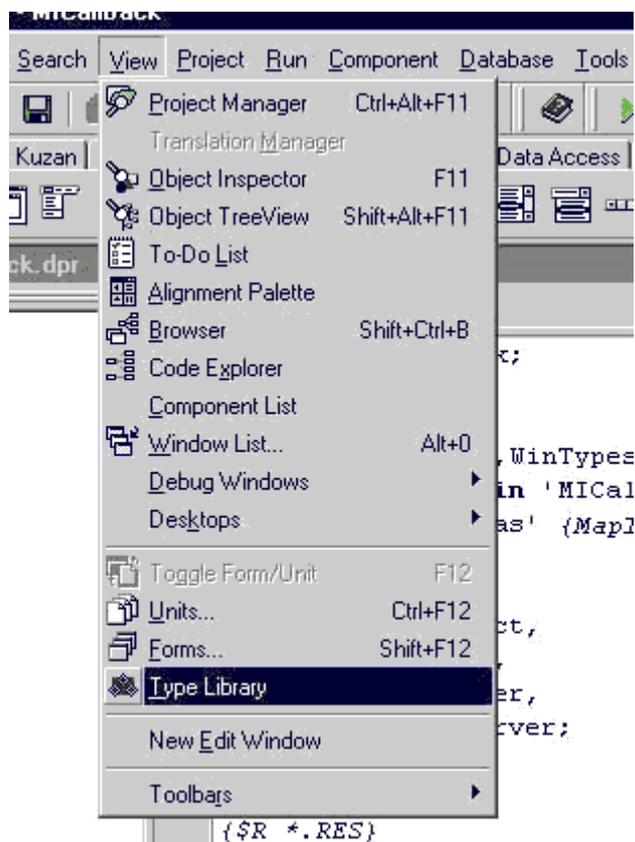
1. Удалите старый компонент.
2. Зарегистрируете в системе библиотеку MICallBack.dll, для этого откройте MICallBack.dpr и в меню Run Delphi выберите Register ActiveX Server. После этого скопируйте саму DLL в каталог Windows.
3. Установите пакет KDPack.dpk в Delphi.



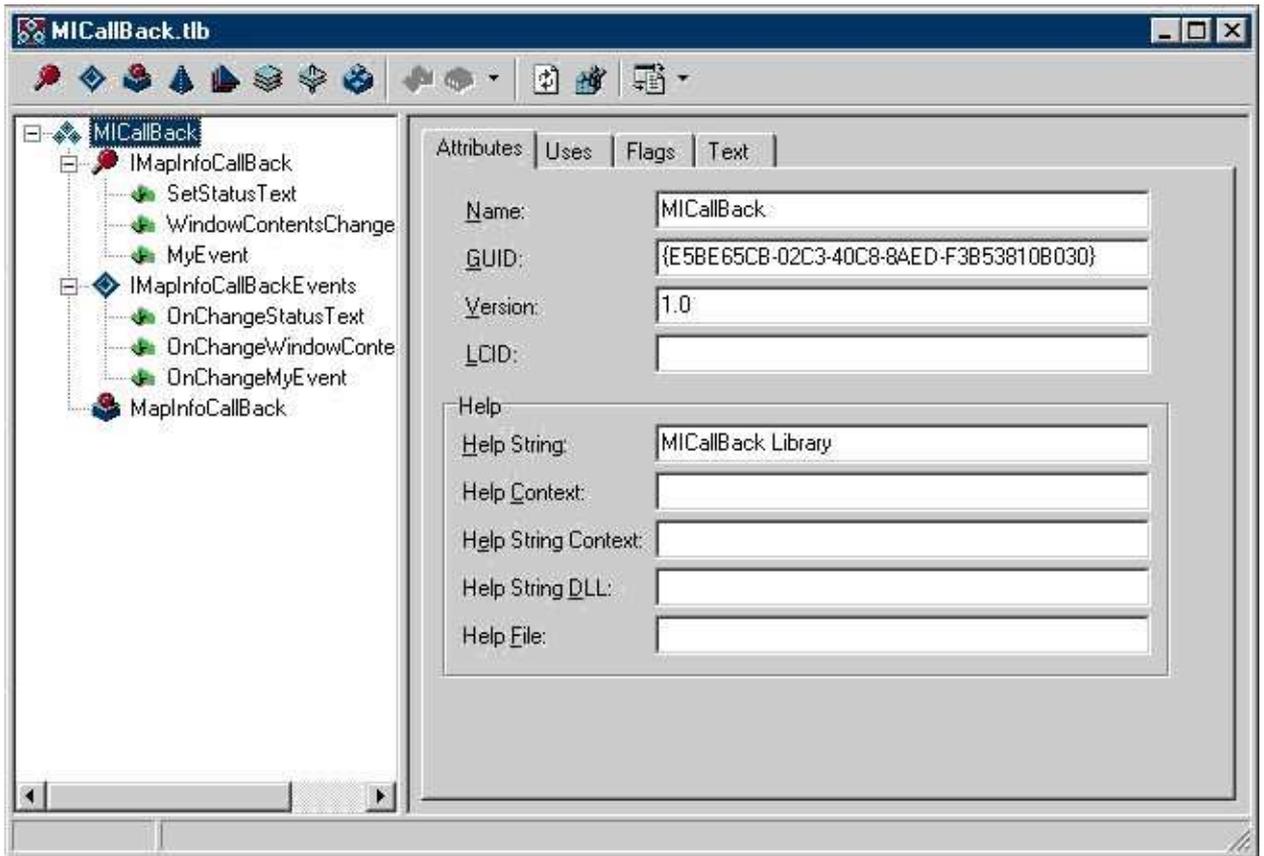
Вот в принципе и все.

#### 4. Сервер автоматизации OLE для обработки CallBack.

Данный сервер я разместил в ActiveX DLL.(данная DLL называется MICallBack.dll) в виде Automation Object.-а.



Чтобы просмотреть методы и свойства данного Automation Object-а, откройте MICallBack.dpr и в меню Run Delphi выберите TypeLibrary.



Откроется окно, где я реализовал CallBack-методы MapInfo и создал сервер автоматизации MICallBack. Обратите внимание, что у данного сервера, помимо присутствия интерфейса IMapInfoCallBack, присутствует и еще интерфейс ImapInfoCallBackEvents (он нам нужен будет для перенаправления событий в компонент и далее в обработчик).

### Листинг интерфейсного модуля

```
unit Call;  
  
{$WARN SYMBOL_PLATFORM OFF}  
  
interface  
  
uses  
  ComObj, ActiveX, Dialogs, AxCtrls, Classes, MICallBack_TLB, StdVcl;  
  
type  
  TMapInfoCallBack = class(TAutoObject, IConnectionPointContainer,  
    IMapInfoCallBack)  
  private  
    { Private declarations }  
    FConnectionPoints: TConnectionPoints;  
    FConnectionPoint: TConnectionPoint;  
    FEvents: IMapInfoCallBackEvents;  
    { note: FEvents maintains a *single* event sink. For access to more  
      than one event sink, use FConnectionPoint.SinkList, and iterate  
      through the list of sinks. }  
  public  
    procedure Initialize; override;  
  protected
```

```
{ Protected declarations }
property ConnectionPoints: TConnectionPoints read FConnectionPoints
    implements IConnectionPointContainer;
procedure EventSinkChanged(const EventSink: IUnknown); override;
procedure SetStatusText(const Status: WideString); safecall;
procedure WindowContentsChanged(ID: Integer); safecall;
procedure MyEvent(const Info: WideString); safecall;
end;

var
    FDLLCall : THandle;

implementation

uses ComServ;

procedure TMapInfoCallBack.EventSinkChanged(const EventSink: IUnknown);
begin
    FEvents := EventSink as IMapInfoCallBackEvents;
end;

procedure TMapInfoCallBack.Initialize;
begin
    inherited Initialize;
    FConnectionPoints := TConnectionPoints.Create(Self);
    if AutoFactory.EventTypeInfo <> nil then
        FConnectionPoint := FConnectionPoints.CreateConnectionPoint(
            AutoFactory.EventIID, ckSingle, EventConnect)
    else FConnectionPoint := nil;
end;

procedure TMapInfoCallBack.SetStatusText(const Status: WideString);
begin
    IF FEvents <> nil Then
    begin
        FEvents.OnChangeStatusText(Status);
    end;
end;

procedure TMapInfoCallBack.WindowContentsChanged(ID: Integer);
begin
    IF FEvents <> nil Then
    begin
        FEvents.OnChangeWindowContentsChanged(ID);
    end;
end;

procedure TMapInfoCallBack.MyEvent(const Info: WideString);
begin
    IF FEvents <> nil Then
    begin
        FEvents.OnChangeMyEvent(Info);
    end;
end;

initialization
    TAutoObjectFactory.Create(ComServer, TMapInfoCallBack,
    Class_MapInfoCallBack,
    ciMultiInstance, tmApartment);
end.
```

Обратите внимание на присутствие двух predefined методов MapInfo SetStatusText и WindowContentsChanged.

Метод MyEvent я пока зарезервировал для реализации своих сообщений (более подробно будет изложено в 3 части цикла).

Итак, что мы видим:

```
IF FEvents <> nil Then // если есть обработчик
begin
    FEvents.OnChangeStatusText(Status); // Отправка сообщения далее - в
данном случае в компонент
```

#### 4. Как заставить MapInfo пересылать CallBack данному OLE серверу и как нам обрабатывать сообщения в компоненте от OLE сервера.

Итак, представляю переработанный компонент:

```
unit KDMapInfoServer;

interface

uses
    StdCtrls, Dialogs, ComObj, Controls, Variants, ExtCtrls, Windows, ActiveX,
    Messages, SysUtils, Classes, MIPCallback_TLB; // - сгенерировано из DLL

Type
    // запись «типа» Variant
    TEvalResult = record
        AsVariant: OLEVariant;
        AsString: String;
        AsInteger: Integer;
        AsFloat: Extended;
        AsBoolean: Boolean;
    end;

type
    // Событие на изменение SetStatusText // генерируется при обратном вызове
    TSetStatusTextEvent = procedure(Sender : TObject; StatusText:
WideString) of object;
    // WindowContentsChanged
    TWindowContentsChanged = procedure(Sender : TObject; ID : Integer) of
object;
    // Для собственных событий
    TMyEvent = procedure(Sender : TObject; Info : WideString) of
object;

    TEvent = class(TInterfacedObject, IUnknown, IDispatch)
    private
        FAppConnection : Integer;
        FAppDispatch : IDispatch;
        FAppDispIntfIID : TGUID;
    protected
        function QueryInterface(const IID: TGUID; out Obj): HRESULT; stdcall;
        function _AddRef: Integer; stdcall;
        function _Release: Integer; stdcall;
        function GetTypeInfoCount(out Count: Integer): HRESULT; stdcall;
```

```
function GetTypeInfo(Index, LocaleID: Integer; out TypeInfo): HRESULT;
stdcall;
function GetIDsOfNames(const IID: TGUID; Names: Pointer; NameCount,
LocaleID: Integer; DispIDs: Pointer): HRESULT; stdcall;
function Invoke(DispID: Integer; const IID: TGUID; LocaleID: Integer;
Flags: Word; var Params; VarResult, ExcepInfo, ArgErr: Pointer): HRESULT;
stdcall;
public
  Constructor Create( AnAppDispatch : IDispatch;
                    Const AnAppDispIntfIID : TGUID);
  Destructor Destroy ; override;
end;

TKDMapInfoServer = class(TComponent)
private
  FOwner          : TWinControl;           // Владелец
  Responder       : Variant;              // Для OLE Disp

  FServer : Variant;
  FHandle : THandle;                       // Зарезервировано
  FActive  : Boolean;                       // Запущен/ не запущен
  FPanel   : TPanel;                       // Панель вывода

  srv_OLE      : OLEVariant;
  srv_disp     : IMapInfoCallbackDisp;
  srv_vTable   : IMapInfoCallback;

  FEvent       : TEvent;

  FSetStatusTextEvent : TSetStatusTextEvent; // СОБЫТИЯ КОМПОНЕНТА
  FWindowContentsChanged : TWindowContentsChanged;
  FMyEvent       : TMyEvent;

  Connected : Boolean;                       // Установлено ли соединение
  MapperID  : Cardinal;                     // ID окна

  procedure SetActive(const Value: Boolean);
  procedure SetPanel(const Value: TPanel);
  procedure CreateMapInfoServer;
  procedure DestroyMapInfoServer;
  { Private declarations }
protected
  { Protected declarations }
public
  { Public declarations }
  constructor Create(AOwner: TComponent); override;
  destructor Destroy; override;
  // Данная процедура выполняет метод сервера MapInfo - Do
  procedure ExecuteCommandMapBasic(Command: String; const Args: array of
const);
  function Eval(Command: String; const Args: array of const): TEvalResult;
virtual;

  procedure WindowMapDef;
  procedure OpenMap(Path : String);
  procedure RepaintWindowMap;

  // Дополнил для генерации события SetStatus при изменении строки
состояния
  // в MapInfo
  procedure DoSetStatus(StatusText: WideString);
  // Дополнил.для генерации события WindowContentsChanged при изменении
окна
```

```
// в MapInfo
procedure DoWindowContentsChanged(ID : Integer);
// Дополнил для генерации собственно события в MapInfo
procedure DoMyEvent(Info: WideString);

published
{ Published declarations }

// Создает соединение с сервером MapInfo
property Active: Boolean read FActive write SetActive;
property PanelMap : TPanel read FPanel write SetPanel;

// Событие возникающее при изменении строки состояния MapInfo
property StatusTextChange : TSetStatusTextEvent read FSetStatusTextEvent
write FSetStatusTextEvent;
Property WindowContentsChanged : TWindowContentsChanged read
FWindowContentsChanged write
FWindowContentsChanged;

Property MyEventChange : TMyEvent read FMyEvent write FMyEvent;
end;

var
// О это вообще хитрость - используется для определения созданного
компонента
// TKDMapInfoServer (см. SetStatusText и Create
KDMapInfoServ : TKDMapInfoServer;

procedure Register;

implementation

// Вот тут-то и хитрость: если сервер создан, то тогда и вызываем SetStatus
///// IF KDMapInfoServ <> nil Then
///// KDMapInfoServ.SetStatus(StatusText);

procedure Register;
begin
RegisterComponents('Kuzan', [TKDMapInfoServer]);
end;

{ TKDMapInfoServer }

constructor TKDMapInfoServer.Create(AOwner: TComponent);
begin
inherited Create(AOwner);
FOwner := AOwner as TWinControl;

KDMapInfoServ := Self; // **** Вот тут и указывается созданный компонент
// TKDMapInfoServer

FHandle := 0;
FActive := False;
Connected := False;
end;

destructor TKDMapInfoServer.Destroy;
begin
DestroyMapInfoServer;
inherited Destroy;
end;
//-----
procedure TKDMapInfoServer.CreateMapInfoServer;
begin
```

```
try
  FServer := CreateOleObject('MapInfo.Application');
except
  FServer := Unassigned;
end;
// Скрываем панели управления MapInfo
ExecuteCommandMapBasic('Alter ButtonPad ID 4 ToolbarPosition (0, 0) Show
Fixed', []);
ExecuteCommandMapBasic('Alter ButtonPad ID 3 ToolbarPosition (0, 2) Show
Fixed', []);
ExecuteCommandMapBasic('Alter ButtonPad ID 1 ToolbarPosition (1, 0) Show
Fixed', []);
ExecuteCommandMapBasic('Alter ButtonPad ID 2 ToolbarPosition (1, 1) Show
Fixed', []);
ExecuteCommandMapBasic('Close All', []);
ExecuteCommandMapBasic('Set ProgressBars Off', []);
ExecuteCommandMapBasic('Set Application Window %D', [FOwner.Handle]);
ExecuteCommandMapBasic('Set Window Info Parent %D', [FOwner.Handle]);

FServer.Application.Visible := True;
if IsIconic(FOwner.Handle) then ShowWindow(FOwner.Handle, SW_Restore);
  BringWindowToTop(FOwner.Handle);

srv_ole := CreateOleObject('MICallback.MapInfoCallback') as IDispatch;
srv_vtable := CoMapInfoCallback.Create;
srv_disp := CreateComObject(CLASS_MapInfoCallback) as
IMapInfoCallbackDisp;
FEvent := TEvent.Create(srv_disp, IMapInfoCallbackEvents);

// Указываем MapInfo, что нужно передавать обратные вызовы нашему OLE
// а там далее по цепочке (см.начало)
FServer.SetCallBack(srv_disp);
end;

procedure TKDMapInfoServer.DestroyMapInfoServer;
begin
  ExecuteCommandMapBasic('End MapInfo', []);
  FServer := Unassigned;
end;

//-----
procedure TKDMapInfoServer.ExecuteCommandMapBasic(Command: String;
const Args: array of const);
begin
  if Connected then
    try
      FServer.Do(Format(Command, Args));
    except
      on E: Exception do MessageBox(FOwner.Handle,
PChar(Format('Ошибка выполнения () - %S',
[E.Message])),
'Warning',
MB_ICONINFORMATION OR MB_OK);
    end;
  end;
end;
//-----
function TKDMapInfoServer.Eval(Command: String;
const Args: array of const): TEvalResult;

Function IsInt(Str : String): Boolean;
var
  Pos : Integer;
begin
  Result := True;
```

```
For Pos := 1 To Length(Trim(Str)) do
begin
  IF (Str[Pos] <> '0') and (Str[Pos] <> '1') and
    (Str[Pos] <> '2') and (Str[Pos] <> '3') and
    (Str[Pos] <> '4') and (Str[Pos] <> '5') and
    (Str[Pos] <> '6') and (Str[Pos] <> '7') and
    (Str[Pos] <> '8') and (Str[Pos] <> '9') and
    (Str[Pos] <> '.') Then
  Begin
    Result := False;
    Exit;
  end;
end;
end;

var
ds_save: Char;
begin
if Connected then
begin
Result.AsVariant := FServer.Eval(Format(Command, Args));
Result.AsString := Result.AsVariant;
Result.AsBoolean := (Result.AsString = 'T') OR (Result.AsString = 't');

IF IsInt(Result.AsVariant) Then
Begin
try
ds_save := DecimalSeparator;
try
DecimalSeparator := '.';
Result.AsFloat := StrToFloat(Result.AsString);
finally
DecimalSeparator := ds_save;
end;
except
Result.AsFloat := 0.00;
end;

try
Result.AsInteger := Trunc(Result.AsFloat);
except
Result.AsInteger := 0;
end;

end
else
Begin
Result.AsInteger := 0;
Result.AsFloat := 0.00;
end;
end;
end;

//-----
procedure TKDMapInfoServer.SetActive(const Value: Boolean);
begin
FActive := Value;

IF FActive then
begin
CreateMapInfoServer;
WindowMapDef;
Connected := True;
end
end
```

```
else
begin
  IF Connected then
  begin
    DestroyMapInfoServer;
    Connected := False;
  end;
end;
end;

//-----
procedure TKDMapInfoServer.SetPanel(const Value: TPanel);
begin
  FPanel := Value;
end;

procedure TKDMapInfoServer.WindowMapDef;
begin
  ExecuteCommandMapBasic('Set Next Document Parent %D Style 1',
[FPanel.Handle]);
  RepaintWindowMap;
end;

procedure TKDMapInfoServer.OpenMap(Path: String);
begin
  ExecuteCommandMapBasic('Run Application «%S»', [Path]);
  MapperID := Eval('WindowInfo(FrontWindow(),%D)',[12]).AsInteger;
  RepaintWindowMap;
end;

procedure TKDMapInfoServer.DoSetStatus(StatusText: WideString);
begin
  IF Assigned(FSetStatusTextEvent) then
    FSetStatusTextEvent(Self,StatusText);
end;

procedure TKDMapInfoServer.DoWindowContentsChanged(ID: Integer);
begin
  IF Assigned(FWindowContentsChanged) then
    FWindowContentsChanged(Self,ID);
end;

procedure TKDMapInfoServer.DoMyEvent(Info: WideString);
begin
  IF Assigned(FWindowContentsChanged) then
    FMyEvent(Self,Info);
end;

procedure TKDMapInfoServer.RepaintWindowMap;
begin
  with PanelMap do MoveWindow(MapperID, 0, 0, FPanel.ClientWidth,
FPanel.ClientHeight, True);
end;

{ TEvent }

function TEvent._AddRef: Integer;
begin
  Result := 2; // Заглушка
end;

function TEvent._Release: Integer;
begin
  Result := 1; // Заглушка
```

```
end;

constructor TEvent.Create(AnAppDispatch: IDispatch;
  const AnAppDispIntfIID: TGUID);
begin
  Inherited Create;
  FAppDispatch := AnAppDispatch;
  FAppDispIntfIID := AnAppDispIntfIID;
  // Передадим серверу
  InterfaceConnect(FAppDispatch, FAppDispIntfIID, self, FAppConnection);
end;

destructor TEvent.Destroy;
begin
  InterfaceDisconnect(FAppDispatch, FAppDispIntfIID, FAppConnection);
  inherited;
end;

function TEvent.GetIDsOfNames(const IID: TGUID; Names: Pointer; NameCount,
  LocaleID: Integer; DispIDs: Pointer): HRESULT;
begin
  // Заглушка не реализована
  Result := E_NOTIMPL;
end;

function TEvent.GetTypeInfo(Index, LocaleID: Integer;
  out TypeInfo): HRESULT;
begin
  // Заглушка не реализована
  Result := E_NOTIMPL;
end;

function TEvent.GetTypeInfoCount(out Count: Integer): HRESULT;
begin
  // Заглушка не реализована
  Count := 0;
  Result := S_OK;
end;

function TEvent.Invoke(DispID: Integer; const IID: TGUID;
  LocaleID: Integer; Flags: Word; var Params; VarResult, ExcepInfo,
  ArgErr: Pointer): HRESULT;
var
  Info, Status : String;
  IDWin       : Integer;
begin
  Case DispID of
    1 : begin
      Status := TDispParams(Params).rgvarg^[0].bstrval;
      IF KMapInfoServ <> nil Then
        KMapInfoServ.DoSetStatus(Status);
      end;
    2 : begin
      IDWin := TDispParams(Params).rgvarg^[0].bval;
      IF KMapInfoServ <> nil Then
        KMapInfoServ.DoWindowContentsChanged(IDWin);
      end;
    3 : begin
      Info := TDispParams(Params).rgvarg^[0].bstrval;
      IF KMapInfoServ <> nil Then
        KMapInfoServ.DoMyEvent(Info);
      end;
  end;
  Result := S_OK;
end;
```

```
end;

function TEvent.QueryInterface(const IID: TGUID; out Obj): HRESULT;
begin
  Result := E_NOINTERFACE;
  IF GetInterface(IID,Obj) Then
    Result := S_OK;
  If IsEqualGUID(IID,FAppDispIntfIID) and GetInterface(IDispatch,Obj) Then
    Result := S_OK;
end;

end.
И так что добавилось - Метод CreateMapInfoServer;
// Создаем наш сервер OLE
srv_ole := CreateOleObject('MICallback.MapInfoCallback') as IDispatch;
srv_vtable := CoMapInfoCallback.Create;
// Получаем Idispatch созданного сервера
srv_disp := CreateComObject(CLASS_MapInfoCallback) as
IMapInfoCallbackDisp;
FEvent := TEvent.Create(srv_disp,IMapInfoCallbackEvents);
// Указываем MapInfo, что нужно передавать обратные вызовы нашему OLE
серверу
// а там далее по цепочке (см.начало)
FServer.SetCallBack(srv_disp);
end;
```

Здесь мы столкнулись с еще одним методом MapInfo помимо рассмотренных ранее методов Do и Eval:

### Метод SetCallBack(IDispatch)

Метод SetCallBack регистрирует объект механизма-управления объектами OLE (OLE Automation) как получатель уведомлений, генерируемых программой MapInfo. Только одна функция уведомления может быть зарегистрирована в каждый данный момент. Параметр: интерфейс **IDispatch** объекта OLE (COM).

Реализация FServer.SetCallBack(srv\_disp); - данным кодом мы заставили MapInfo уведомлять наш OLE сервер.

Хорошо, скажете вы, ну заставили, но он-то уведомляет сервер OLE, а не нашу программу. Для этого я ввел следующий код (прим. *Реализацию использования интерфейса событий OLE сервера я подробно расписывать не стану - для этого читайте в книгах главы по COM*).

Я сделал так: ввел класс отвечающий за принятие событий от COM(OLE) объекта.

```
TEvent = class(TInterfacedObject, IUnknown, IDispatch)
private
  FAppConnection : Integer;
  FAppDispatch : IDispatch;
  FAppDispIntfIID : TGUID;
protected
  function QueryInterface(const IID: TGUID; out Obj): HRESULT; stdcall;
  function _AddRef: Integer; stdcall;
  function _Release: Integer; stdcall;
  function GetTypeInfoCount(out Count: Integer): HRESULT; stdcall;
  function GetTypeInfo(Index, LocaleID: Integer; out TypeInfo): HRESULT;
stdcall;
  function GetIDsOfNames(const IID: TGUID; Names: Pointer; NameCount,
LocaleID: Integer; DispIDs: Pointer): HRESULT; stdcall;
```

```
function Invoke(DispID: Integer; const IID: TGUID; LocaleID: Integer;  
Flags: Word; var Params; VarResult, ExcepInfo, ArgErr: Pointer): HRESULT;  
stdcall;  
public  
  Constructor Create( AnAppDispatch : IDispatch;  
                    Const AnAppDispIntfIID : TGUID);  
  Destructor Destroy ; override;  
end;
```

Создание этого класса в компоненте реализовано так

```
FEvent := TEvent.Create(srv_disp, IMapInfoCallBackEvents);
```

В методе Invoke и происходит прием и получение сообщений и пересылка их в обработчик моего компонента.

Еще раз на последующие вопросы касательно COM (OLE) серверов отвечу: данная тема выходит за рамки данной статьи – советую почитать книгу Александровского А.Д «Delphi 5 разработка корпоративных приложений».

Напоследок. Модуль MICallBack\_TLB.pas импортирован из DLL командой меню DELPHI Import Type Libray.

**Примечание:**

*при импорте данный сервер устанавливать не нужно, нет смысла, он нам нужен только для приема сообщений из MapInfo.*

Вот в принципе все во второй части, создание пользовательских событий и обработка их в следующей главе.

## Просто и ясно о MapInfo и Delphi III

[Дмитрий Кузан](#)

дата публикации 09-04-2002 13:23

### 1. Интеграция инструментальных панелей MapInfo – краткий вводный курс.

Вы не можете переподчинить стандартные инструментальные панели MapInfo. Если вы хотите, чтобы ваша клиентская программа имела такие панели, вы должны сами создать панели и кнопки на Delphi (например, используя Tpanel и Tbutton) и в их обработчике посылать специальные команды MapInfo для того, чтобы MapInfo включало или переключало режимы работы (например, с выбора объекта на перемещения окна карты).

Если вы хотите, чтобы кнопка панели эмулировала стандартную кнопку MapInfo, используйте метод MapInfo **Run Menu Command**.

Например в обработчике OnClick пропишите следующую команду:

```
KDMapInfoServer1.ExecuteCommandMapBasic('Run Menu Command 1702',[]);
```

Когда пользователь нажмет на эту кнопку, программа вызовет метод MapInfo – Run Menu Command, который активизирует инструмент под номером 1702 (инструмент перемещение карты «рука»).

«Магический» номер 1702 ссылается на инструмент «рука», служащий для перемещения (сдвига) карты.

Вместо того, чтобы использовать такие числа, вы можете использовать идентификаторы, более понятные в тексте программы. MapBasic определяет стандартный идентификатор M\_TOOLS\_RECENTER который имеет значение 1702. Таким образом, этот пример можно записать так:

```
KDMapInfoServer1.ExecuteCommandMapBasic('Run Menu Command %S',  
[M_TOOLS_RECENTER]);
```

Использование идентификаторов (типа M\_TOOLS\_RECENTER) делает вашу программу более читабельной, но перед использованием вы должны включить в программу (в Uses) соответствующий заголовочный файл MapBasic. Для Delphi я положил файл Global.pas.

В следующей таблице приведены кратко идентификаторы основных инструментальных кнопок MapInfo (для более подробной информации смотрите документацию по MapBasic).

Кнопки панели Операции	Номер	Идентификатор	Прим.
Выбор	1701	M_TOOLS_SELECTOR	Панель ОПЕРАЦИИ
Выбор в прямоугольнике	1722	M_TOOLS_SEARCH_RECT	Панель ОПЕРАЦИИ
Выбор в круге	1703	M_TOOLS_SEARCH_RADIUS	Панель ОПЕРАЦИИ
Выбор в области	1704	M_TOOLS_SEARCH_BOUNDARY	Панель ОПЕРАЦИИ
Увеличивающая лупа	1705	M_TOOLS_EXPAND	Панель ОПЕРАЦИИ
Уменьшающая лупа	1706	M_TOOLS_SHRINK	Панель ОПЕРАЦИИ
Ладонка (рука)	1702	M_TOOLS_RECENTER	Панель ОПЕРАЦИИ
Информация	1707	M_TOOLS_PNT_QUERY	Панель ОПЕРАЦИИ
Подпись	1708	M_TOOLS_LABELER	Панель ОПЕРАЦИИ

Линейка	1710	M_TOOLS_RULER	Панель ОПЕРАЦИИ
Переноска	1734	M_TOOLS_DRAGWINDOW	Панель ОПЕРАЦИИ
Символ	1711	M_TOOLS_POINT	Панель ПЕНАЛ
Линия	1712	M_TOOLS_LINE	Панель ПЕНАЛ
Полилиния	1713	M_TOOLS_POLYLINE	Панель ПЕНАЛ
Дуга	1716	M_TOOLS_ARC	Панель ПЕНАЛ
Полигон	1714	M_TOOLS_POLYGON	Панель ПЕНАЛ
Эллипс	1715	M_TOOLS_ELLIPSE	Панель ПЕНАЛ
Прямоугольник	1717	M_TOOLS_RECTANGLE	Панель ПЕНАЛ
Прямоугольник скругленный	1718	M_TOOLS_ROUNDEDRECT	Панель ПЕНАЛ
Текст	1709	M_TOOLS_TEXT	Панель ПЕНАЛ
Рамка	1719	M_TOOLS_FRAME	Панель ПЕНАЛ

## 2. Настройка «быстрых» меню MapInfo

MapInfo вызывает «быстрые» меню, если пользователь нажимает правую кнопку мышки в окне MapInfo. Эти меню появляются даже во внедренных приложениях. В зависимости от характера вашего приложения вы можете захотеть модифицировать или даже удалить такое меню. Например, вы, возможно, захотите удалить команду ДУБЛИРОВАТЬ ОКНО, так как эта команда не работает в OLE-приложении.

Чтобы удалить одну или несколько команд из локального меню, используйте оператор MapBasic Alter Menu... Remove или переопределите меню целиком, используя оператор Create Menu. Подробнее смотрите в Справочнике MapBasic.

Чтобы добавить команду к локальному меню, используйте оператор MapBasic Alter Menu ... Add и синтаксис предложений Calling OLE.

Чтобы удалить «быстрое» меню полностью, используйте оператор MapBasic Create Menu и управляющий код «(-» как новое определение меню. Например, следующий оператор разрушает «быстрое» меню для окон Карты:

```
KDMapInfoServer1.ExecuteCommandMapBasic(' «Create Menu «MapperShortcut»« ID
17 As «(-»« « ', []);
```

## 3. Создание собственных уведомляющих вызовов (Callbacks).

Во второй части мы рассмотрели возможность перехвата двух стандартных вызовов MapInfo – это дало нам возможность подключить к своей программе статус бар MapInfo и узнавать об изменениях окон MapInfo. Все это очень неплохо, но сразу возник вопрос: а как создавать и обрабатывать сообщения собственные, не входящие в MapInfo.

Если вы хотите, чтобы MapInfo сообщало вашей клиентской программе, когда пользователь применяет инструментальную кнопку, создайте такую кнопку оператором Alter ButtonPad... Add. Определите кнопку в соответствии с именем метода для обработки.

### Примечание.

Этот метод определен мной как MyEvent в OLE объекте)

Пример:

```
KDMapInfoServer1.ExecuteCommandMapBasic('Alter ButtonPad ID 1 Add ToolButton
calling ole                                     <MyEvent> ID 1 Icon 0 Cursor 0
DrawMode 34 uncheck',[]);
```

Заметьте, что инструментальные панели MapInfo скрыты, подобно остальной части интерфейса пользователя MapInfo. Пользователь не будет видеть новую кнопку. Вы можете добавить иконку, кнопку или другой видимый элемент управления к интерфейсу пользователя вашей клиентской программы. Когда пользователь укажет на него мышкой, пошлите MapInfo оператор Run Menu Command ID , с идентификатором созданной кнопки чтобы активизировать этот инструмент.

```
KDMapInfoServer1.ExecuteCommandMapBasic('Run Menu Command ID 1',[]);
```

**Примечание:**

*Информацию по Alter Button Pad смотрите в документации.*

Если вы хотите, чтобы MapInfo сообщала вашей клиентской программе, когда пользователь выбирает созданную вами команду меню, определите такую кнопку оператором Alter Menu... Add с указанием имени OLE метода (см. выше).

Внутри метода (в данном случае в обработчике компонента MyEventChange) обработайте аргументы (Info), посланные MapInfo.

#### 4. Обработка переданных данных

Когда пользователь использует команды или кнопки, MapInfo посылает вашему OLE-методу строку, содержащую восемь элементов, разделенных запятыми. Например, строка, посланная MapInfo, может выглядеть так:

```
<<MI:-73.5548,42.122,F,F,-72.867702,43.025,202,>>
```

Содержание такой строки проще понять, если вы уже знакомы с функцией MapBasic CommandInfo(). Когда вы пишете приложения, вы можете создать новые команды меню и кнопки, вызывающие MapBasic-процедуры. Внутри процедуры-обработчика вызовите функцию CommandInfo(), чтобы получить информацию. Например, следующее обращение к функции определяет, координату X и Y места на карте где пользователи применил инструмент.

```
var
X,Y : String;
begin
KDMapInfoServer1.ExecuteCommandMapBasic('Set CoordSys Layout Units
<<mm>',[]);
X := KDMapInfoServer1.Eval('CommandInfo(%S)',[CMD_INFO_X]).AsString;
Y := KDMapInfoServer1.Eval('CommandInfo(%S)',[ CMD_INFO_Y]).AsString;
ShowMessage('X= ' + X + ' Y = ' + Y);
```

Значение	Код для событий, связанных с меню	Код для событий, связанных с кнопкой
1		CMD_INFO_X
2		CMD_INFO_Y
3		CMD_INFO_SHIFT
4		CMD_INFO_CTRL
5		CMD_INFO_X2

6		CMD_INFO_Y2
7		CMD_INFO_TOOLBTN
8	CMD_INFO_MENUITEM	

Когда вы создаете команду меню или кнопку, которая использует синтаксис вызова OLE, MapInfo создает строку, содержащую разделенные запятой все восемь возвращаемых CommandInfo() значений. Строка начинается с префикса «MI:», чтобы ваш OLE-сервер мог определять, что обращение метода было сделано MapInfo.

Строка, которую MapInfo посылает вашему методу, выглядит следующим образом:

```
«MI:» +  
CommandInfo(1) + «,» + CommandInfo (2) + «,» +  
CommandInfo(3) + «,» + CommandInfo (4) + «,» +  
CommandInfo(5) + «,» + CommandInfo (6) + «,» +  
CommandInfo (7) + «,» + CommandInfo (8)
```

Предположим, что ваше приложение добавляет команду меню к локальному меню OLE-методу строку. Если команда меню имеет номер 101 , строка будут выглядеть следующим образом:

```
«MI :,,,,,, 101»
```

В этом случае большинство элементов строки пусто, потому что функция CommandInfo( ) может возвращать только эту одну часть информации.

Теперь предположим что вы создаете кнопку которая позволяет пользователю выбирать линии на карте.Строка теперь примет вид:

```
«MI: -73.5548,42.122,F,F,-72.867702,43.025,202,»
```

Теперь строка включает несколько элементов.

Первые два элемента содержат x- и y- координаты точки, на которые пользователь указал мышкой. Следующие два элемента сообщают, была ли нажата клавиша SHIFT или CTRL. Предпоследние два элемента содержат координаты точки, где пользователь отпустил кнопку мышки. И последний – указывает номер идентификатора кнопки.

**Совет:**

*Если вы приписываете уникальный идентификатор каждой из ваших кнопок, вы можете сделать так, что все кнопки будут вызывать один и тот же метод. ваш метод может определять, какая из кнопок вызвала его, используя седьмой аргумент в переданной строке.*

**5. Примечание 1: Описание констант MapInfo (Global.pas)**

Примечание - данный файл был взят мной с Интернета. Хочу сразу сделать предупреждение – разработчики MapInfo заявляют что набор констант может быть подвергнут изменениям в следующих редакциях MapInfo. Данный набор констант адаптирован под пятую версию.

Вот в принципе и все что нужно для работы с MapInfo в Delphi, держайте.