

ФЕДЕРАЛЬНОЕ АГЕНСТВО ПО ОБРАЗОВАНИЮ
КЕМЕРОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Математический факультет

Реферат

На тему: «Интеграция приложений и данных в MapInfo»

студента 3 курса, группы М-064

Ткаченко Ивана Сергеевича

Кемерово 2008

Что такое Интегрированная Картография?

Интегрированная картография позволяет управлять пакетом MapInfo, используя языки программирования отличные от MapBasic. Например, если Вам хорошо знакомо программирование на языке Visual Basic, можно включить (интегрировать) окно Карты MapInfo в Ваше приложение, написанное на языке Visual Basic, выполняя при этом основную часть или даже всю работу по программированию в среде Visual Basic. Такой способ разработки приложений известен как Интегрированная Картография, так как при этом интегрируются элементы MapInfo в другое приложение.

Причем основную работу по поддержанию векторных карт берет на себя MapInfo (MapBasic) а 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).

Метод Интегрированной Картографии обеспечивает простейший способ включения окон MapInfo в приложения, разработанные в других средах программирования, не использующих MapBasic.

Вид на экране компьютера приложения с Интегрированной Картой определяется разработчиком. При желании можно создать интерфейс пользователя, радикально отличающийся от интерфейса MapInfo. Например, на следующем рисунке показано окно Карты MapInfo, интегрированное в форму окна диалога Visual Basic.

При интегрировании окна Карты MapInfo в программу пользователь видит на экране оригинальное полнофункциональное окно MapInfo, а не растр, метафайл или графическое представление какого-либо другого типа. Можно разрешить пользователю интерактивно взаимодействовать с Картой (используя, например, инструменты Лупа для увеличения Карты). Интегрированное окно Карты имеет все возможности, присущие окну Карты в среде MapInfo.

Когда пользователь запускает приложение с встроенной Картой, заставка MapInfo не демонстрируется.

Для создания приложения с Интегрированной Картой нужно написать программу — но не программу на языке MapBasic. Приложения с Интегрированной Картой могут быть написаны на нескольких языках программирования, среди которых наиболее часто используются С и Visual Basic.

В программе должна присутствовать инструкция, запускающая MapInfo в фоновом режиме. Например, в программе на языке Visual Basic Можно запустить MapInfo вызовом функции CreateObject(). Программа MapInfo запускается в фоновом режиме незаметно для пользователя, не выводя заставку на дисплей.

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.

Если нужно открыть окно Карта, используется оператор Map From языка MapBasic точно таким же образом, как в обычной MapBasic-программе. Однако в приложении с Интегрированной Картой Нужно также использовать дополнительные операторы (например, Set Next Document Parent), чтобы окно Карты могло стать подчиненным (порожденным) окном Вашего приложения. Этот процесс известен как "переподчинение" (reparenting) окна. Можно переподчинить окна Карты, Списка, Графика, Отчета и Легенды.

Переподчинение окон MapInfo другому приложению не дает программе MapInfo автоматического доступа к данным этого приложения. Для отображения данных приложения в окне MapInfo Нужно предварительно записать эти данные в таблицу MapInfo. Простейший пример на Visual Basic.

Следующая программа на Visual Basic дает представление о том, как легко встроить окно MapInfo в другую программу.

Сначала создадим новый проект Visual Basic. В процедуре General Declarations (общие определения) объявим переменную типа Object. (В этом примере она будет называться mi) - Dim mi As Object.

Затем добавим в процедуру Form_Load следующие строки: Sub Form_Load():

- Set mi = CreateObject("MapInfo.application");
- mi.do "Set Application Window " & Form1.hWnd;
- mi.do "Set Next Document Parent " & Form1.hWnd & " Style 1";
- mi.do "Open Table ""World"" Interactive Map From World";
- mi.RunMenuCommand 1702;
- mi.do "Create Menu ""MapperShortcut"" ID 17 As ""(-"" ";
- End Sub.

Если запустить программу на Visual Basic, она запускает MapInfo, которая создает окно Карты. При этом MapInfo действует как "скрытый" сервер, а окно Карты ведет себя как порожденное программой Visual Basic.

Подробное обсуждение интегрирования MapInfo в приложения, написанные на VB.

Последующее обсуждение показывает, как интегрировать элементы MapInfo в приложение, написанное на языке Visual Basic (в дальнейшем VB-приложение или VB-программа).

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

- Set mapinfo = CreateObject("MapInfo.Application").

Для подключения к ранее исполнявшемуся экземпляру MapInfo, который не был запущен вызовом функции CreateObject(), используется функция GetObject():

- Set mapinfo = GetObject(, "MapInfo.Application").

Функции CreateObject() и GetObject() используют механизм управления объектами OLE (OLE Automation) для связи с MapInfo. Если необходимо применить DDE-связь вместо OLE-связи, используется функция Shell() языка Visual Basic для запуска программы MapInfo, а затем используется свойство (property) LinkMode для установления DDE-связи.

В 32-разрядной версии Windows (Windows95 или Windows NT) можно запускать несколько экземпляров MapInfo. Если запустить MapInfo и вслед за этим программу, использующую Интегрированную Картографию и вызывающую CreateObject(), то будут работать два независимых экземпляра MapInfo. Однако в 16-разрядной версии может работать только один экземпляр MapInfo; если запустить MapInfo и вслед за этим программу, использующую Интегрированную Картографию и вызывающую CreateObject(), то она не сможет запустить второй экземпляр MapInfo.

Пересылка команд в программу MapInfo. После запуска программы MapInfo необходимо сконструировать текстовые строки, представляющие операторы языка MapBasic. Например, для исполнения программой MapInfo MapBasic-оператора Open Table Можно задать в VB-программе следующую строку:

- msg = "Open Table ""RUSSIA.TAB""Interactive".

Если установить связь с MapInfo, используя механизм управления объектами OLE (OLE Automation), нужно передавать командную строку программе MapInfo методом Do. Например:

- mapinfo.Do msg.

При использовании метода Do программа MapInfo исполняет командную строку точно так же, как если бы пользователь ввел команду с клавиатуры в окно MapBasic.

Если установили связь с MapInfo, используя динамический обмен данными (DDE), передаётся командная строка программе MapInfo DDE-методом LinkExecute.

Замечание: Можно передать оператор в программу MapInfo, если этот оператор допустим в окне MapBasic. Например, нельзя переслать MapBasic-оператор Dialog, поскольку его использование не разрешено в окне MapBasic.

Запрос данных от программы MapInfo. Для выполнения запроса из программы-клиента значения MapBasic-выражения задайте в VB-программе строку, представляющую выражение. Например, если нужно определить значение, возвращаемое MapBasic-функцией WindowID(O), задается следующая строка (в среде Visual Basic):

- msg = "WindowID(0)".

Если установлена связь с MapInfo, используя механизм управления объектами OLE (OLE Automation), передайте строку выражения программе MapInfo OLE-методом Eval. Например:

- Dim result As String;
- result = mapinfo.Eval "WindowID(0)".

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, "T" или "F" соответственно.

Если установлена связь с MapInfo, используя динамический обмен данными (DDE), запросите значение выражения DDE-методом LinkRequest.

Переподчинение окон MapInfo. После запуска MapInfo используется оператор Set Application Window языка MapBasic для обеспечения перехвата управления программой-клиентом диалоговых окон и сообщений об ошибках программы MapInfo. (В следующем примере "FormName" является именем формы в среде Visual Basic):

- msg = "Set Application Window " & FormName.hWnd mapinfo.Do msg.

Затем, в желаемой точке включения окна MapInfo в Ваше VB-приложение передается MapInfo оператор Set Next Document, за которым следует MapBasic-оператор, создающий окно. Например, следующий фрагмент кода создает окно Карта MapInfo как подчиненное окно программы-клиента. (В этом примере "MapFrame" является именем элемента управления Picture Box (Поле Иллюстрации) в среде Visual Basic):

```
- msg = "Set Next Document Parent " & MapFrame.hWnd & " Style 1"  
mapinfo.Do msg;
```

```
- msg = "Map From States" mapinfo.Do msg.
```

Оператор Set Next Document позволяет "переподчинять" окна документов. Синтаксис этого оператора требует указания уникального номера HWND элемента управления в VB-программе. При последующем создании окна документа MapInfo (с использованием операторов Map, Graph, Browse, Layout или Create Legend) создаваемое окно переподчиняется таким образом, что элемент управления программы-клиента с этим значением HWND становится для окна порождающим объектом.

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(O) — целочисленный ID-номер окна (Window ID) в MapInfo, так как многие операторы языка MapBasic требуют задания этого номера. Этот запрос выполняется следующим образом:

```
- mapid = Val(mapinfo.eval("WindowID(0)")).
```

Нужно заметить, что даже после переподчинения окна Карты, MapInfo продолжает управлять им. Если часть окна нужно перерисовать, MapInfo автоматически обновляет его. Поэтому клиентская программа может не обращать внимания на сообщения о перерисовке, адресованные подчиненному окну.

Если программировать на С, то просто так игнорировать сообщения о перерисовке не удастся. В этом случае нужно добавить в описание порождающего окна стиль `WS_CLIPCHILDREN`.

Переподчинение окон Легенд, растровых диалогов и других окон MapInfo. MapInfo имеет несколько немодальных окон, включая окно Информации, окно Сообщений, диалогов, относящихся к растрам и окно Статистики. Чтобы изменить порождающее окно для одного из этих специальных "плавающих" окон, используется оператор `MapBasic Set Window ... Parent`. Например, программа-пример FindZip использует следующее предложение:

- `mapinfo.do "Set Window Info Parent " & FindZipForm.hWnd`.

Нужно заметить, 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` Можно создавать дополнительные окна Легенды.

Для одного окна Легенды используется оператор `MapBasic Window Legend Parent`.

Чтобы создать дополнительное окно Легенды, используется оператор `MapBasic Set Next Document` и оператор `Create Legend`. Нужно заметить, 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.

Можно создать "плавающее" окно Легенды внутри окна Карты. Для этого в операторе `Set Next Document` указать окно Карты как порождающее.

Интеграция инструментальных панелей MapInfo. Нельзя переподчинить инструментальные панели MapInfo. Если требуется, чтобы клиентская программа имела такие панели, нужно создать кнопки на языке,

который используется. Например, если используется Visual Basic, то нужно создать кнопки при помощи Visual Basic.

Если требуется, чтобы кнопка панели Visual Basic эмулировала стандартную кнопку MapInfo, используется метод RunMenuCommand. (Этот метод действует так же, как оператор MapBasic Run Menu Command). Например, программа может содержать процедуру InfoTool_Click с оператором mapinfo.RunMenuCommand 1707.

Когда пользователь нажимает на соответствующую кнопку, программа вызывает метод MapInfo RunMenuCommand, который активизирует инструмент под номером 1707 (инструмент Информация MapInfo).

"Магический" номер 1707 ссылается на инструмент Информация. Вместо того, чтобы использовать такие числа, можно использовать идентификаторы, более понятные в тексте программы. MapBasic определяет идентификатор M_TOOLS_PNT_QUERY, который имеет значение 1707.

Таким образом, этот пример можно записать так:

- mapinfo.RunMenuCommand M_TOOLS_PNT_QUERY.

Использование идентификаторов (типа M_TOOLS_PNT_QUERY) делает программу более легкой для чтения. Однако, если планируется использовать идентификаторы в коде, то нужно включить соответствующий заголовочный файл MapBasic. Если используется Visual Basic, используется файл MAPBASIC.BAS. Для C используется файл MAPBAS1C.H.

В следующей таблице приведены идентификаторы инструментальных кнопок MapInfo. Они содержатся в файлах MAPBASIC.BAS (для Visual Basic), MAPBASIC.H (для C), и MENUS.DEF (для MapBasic).

Таблица 1 - Идентификаторы инструментальных кнопок MapInfo

Кнопки панели Операции	Номер	Идентификатор
Стрелка	1701	M_TOOLS_SELECTOR
Выбор в рамке	1722	M_TOOLS_SEARCH_RECT
Выбор в круге	1703	M_TOOLS_SEARCH_RADIUS
Выбор в области	1704	M_TOOLS_SEARCH_BOUNDARY

Увеличивающая лупа	1705	M_TOOLS_EXPAND
Уменьшающая лупа	1706	M_TOOLS_SHRINK
Ладонка	1702	M_TOOLS_RECENTER
Информация	1707	M_TOOLS_PNT_QUERY
Подпись	1708	M_TOOLS_LABELER
Линейка	1710	M_TOOLS_RULER
Переноска	1734	M_TOOLS_DRAGWINDOW
Символ	1711	M_TOOLS_POINT
Линия	1712	M_TOOLS_LINE
Полилиния	1713	M_TOOLS_POLYLINE
Дуга	1716	M_TOOLS_ARC
Полигон	1714	M_TOOLS_POLYGON
Эллипс	1715	M_TOOLS_ELLIPSE
Прямоугольник	1717	M_TOOLS_RECTANGLE
Скругленный прямоугольник	1718	M_TOOLS_ROUNDEDRECT
Текст	1709	M_TOOLS_TEXT
Рамка	1719	M_TOOLS_FRAME

Также можно создавать пользовательские кнопки, которые вызывают определенные процедуры при нажатии на них.

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 приложения можно модифицировать или даже удалить такое меню.

Чтобы удалить одну или несколько команд из локального меню, используется оператор MapBasic Alter Menu ... Remove или переопределяется меню целиком, используя оператор Create Menu.

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

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

- mapinfo.do "Create Menu ""MapperShortcut"" ID 17 As ""(-"" "

Вывод на печать интегрированного окна MapInfo. Можно использовать оператор Print Win языка MapBasic для вывода окна MapInfo на печать даже в том случае, когда оно переподчинено:

- Private Sub Menu_PrintMap_Click();
- mapinfo.do "PrintWin" End Sub.

Нужно заметить, что оператор PrintWin печатает Карту на отдельной странице.

Также можно использовать оператор MapBasic Save Window для сохранения содержимого окна Карты в формате Windows metafile (WMF). Если пользователь выбирает PrintForm, программа создает метафайл с содержимым окна Карты, присоединяет его к форме и затем использует метод Visual Basic PrintForm.

Обнаружение ошибок времени исполнения. Когда клиентская программа посылает в MapInfo командную строку, возможно возникновение ошибок. Например, команда Map From World потерпит неудачу, если таблица World не открыта. MapInfo в этом случае сгенерирует код ошибки.

Чтобы обработать ошибку MapInfo, установите обработчик. В Visual Basic, например, используется оператор On Error.

Распечатка кодов ошибок приведена в текстовом файле ERRORS.DOC. Нужно заметить, что свойство LastErrorCode возвращает значения, которые на 1000 больше, чем коды в ERRORS.DOC. Другими словами, если возникла ошибка с кодом 311, то LastError Code=1311.

Завершение программы MapInfo. Если запущен новый экземпляр MapInfo вызовом функции CreateObject(), то этот экземпляр MapInfo завершается автоматически при освобождении соответствующей объектной переменной. Если объектная переменная является локальной, она автоматически освобождается при выходе из локальной процедуры. Для освобождения глобальной объектной переменной необходимо явно присвоить этой переменной значение "Nothing":

- Set mapinfo = Nothing.

Если используется для связи с MapInfo динамический обмен данными (DDE), можно завершить исполнение MapInfo, используя DDE-метод LinkExecute для пересылки командной строки "End MapInfo" из Вашей программы в программу MapInfo.

Замечание о командных строках MapBasic. Как показано ранее, можно в VB-программе создавать строки, представляющие операторы языка MapBasic, и затем пересылать эти строки в программу MapInfo посредством OLE-метода Do. Нужно заметить, что можно объединить два и более оператора в одну командную строку, как показано ниже (в языке Visual Basic символ & выполняет конкатенацию строк):

- Dim msg As String;
- msg="Open Table ""States""Interactive";
- msg=msg & "Set Next Document Parent " & Frm.hWnd & " Style 1";
- msg=msg & "Map From States";
- mapinfo.do msg.

При обработке командной строки в процессе исполнения MapInfo автоматически определяет, что данная строка содержит три отдельных MapBasic-оператора — оператор Open Table, оператор Set Next Document, и оператор Map From. Программа MapInfo способна различать в строке отдельные операторы, так как слова Open, Set и Map являются зарезервированными ключевыми словами языка MapBasic.

Нужно отметить наличие пробела после ключевого слова Interactive. Его присутствие необходимо, так как без этого пробела командная строка содержала бы подстроку "InteractiveSet", не имеющую смысла в синтаксисе языка MapBasic. Поскольку каждая командная строка оканчивается пробелом, MapInfo может определить, что подстроки Interactive и Set являются отдельными ключевыми словами.

Использование уведомляющих вызовов (Callbacks) для получения информации из MapInfo.

Можно построить приложение так, чтобы MapInfo автоматически посылало информацию клиентской программе. Например, можно сделать так, чтобы всякий раз, когда изменяется активное окно Карты, MapInfo вызывало клиентскую программу, чтобы сообщить ID-номер окна. Такой тип уведомления известен как обратный вызов или уведомление (callback).

Обратные вызовы позволяют, чтобы MapInfo посылало информацию клиентской программе в следующих случаях:

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

Требования к функциям уведомления. Если планируется использовать обратные вызовы, клиентская программа должна быть способна функционировать, как DDE-сервер или как сервер Автоматизации OLE. Visual Basic 4.0 Professional Edition и C++ могут создавать такие приложения.

Однако приложения Visual Basic 3.0 не могут являться серверами Автоматизации OLE, поэтому они должны использовать DDE.

Схема использования уведомлений в OLE. Краткая схема использования повторных вызовов посредством OLE:

1. Используя Visual Basic 4.0, C++, или другой язык, позволяющий создать OLE-сервер, пишется определение класса, включающее один или большее количество методов OLE.

2. Если требуется имитировать строку состояния MapInfo, создается метод, называемый. SetStatusText. Определяется этот метод так, чтобы у него был один аргумент: строка.

3. Если нужно, чтобы MapInfo сообщало клиентской программе о выборе команды меню или кнопки, пишется один или несколько дополнительных методов с произвольными именами. Каждый из этих методов должен иметь один аргумент: строку.

4. Создаётся объект, используя класс. Например, если название класса "CMyClass", следующий оператор Visual Basic создает объект этого класса:

- Public myObject As New CMyClass.

5. После того, как программа запустит MapInfo, вызывается метод MapInfo SetCallback и точно указывается название объекта:

- mapinfo.SetCallback myObject.

6. Если нужно, чтобы MapInfo сообщало клиентской программе, когда пользователь применяет инструментальную кнопку, создаётся такая кнопка оператором Alter ButtonPad ... Add. Определяется кнопка в соответствии с именем метода (см. шаг 4).

8. Если нужно, чтобы MapInfo сообщала клиентской программе, когда пользователь выбирает команду меню, определите такую кнопку оператором Alter Menu ... Add с указанием имени метода (см. шаг 4).

Синтаксис C/C++ для функций уведомления. Если используется метод MapInfo SetCallback, MapInfo может автоматически генерировать обратные

вызовы для объекта IDispatch. Обратные вызовы стандарта MapInfo имеют следующий синтаксис:

- SCODE SetStatusText(LPCTSTR lpszMessage).

MapInfo вызывает метод SetStatusText всякий раз, когда изменяется содержание строки сообщений в MapInfo. Единственный аргумент — текст сообщения в строке состояний:

- SCODE WindowContentsChanged(Unsigned Long windowID).

MapInfo вызывает метод WindowContentsChanged всякий раз, когда изменяется содержание окна Карты. Единственный аргумент представляет собой идентификатор этого окна. Этот повторный вызов аналогичен процедуре MapBasic WinChangedHandler.

Другие способы использования уведомлений. Как говорилось ранее, MapInfo может использовать обратные вызовы OLE, чтобы послать информацию клиентской программе. В некоторых случаях, однако, нужно применять повторные вызовы, которые не используют OLE. Например, если вы пишете программы в Visual Basic 3.0, нельзя использовать OLE, потому что Visual Basic 3.0 не позволяет создавать собственные серверы Автоматизации OLE.

MapInfo поддерживает два типа уведомлений, которые не используют механизм OLE: использующие DDE, и использующие приложения MapBasic (файлы MBX).

Обратные вызовы DDE. Когда вы создаете кнопки на инструментальной панели или команды меню, вы указываете предложение Calling. Чтобы пользоваться обратным вызовом посредством DDE, используется синтаксис вызова DDE-сервера. Всякий раз, когда пользователь использует кнопку или команду меню, MapInfo открывает DDE-связь с DDE-сервером, и затем посылает строку Вашему объекту. Строка использует формат, обсужденный в предыдущей секции (например, "M1:,,,,,, 101 ").

Процедура Form Load посылает MapInfo оператор Alter ButtonPad ... Add с предложением Calling DDE "FindZip", "MainForm".

Всякий раз, когда пользователь применяет инструмент, MapInfo открывает DDE-связь с программой и посылает строку "MainForm" объекту. ("MainForm" — значение свойства формы LinkTopic).

Обратные вызовы MBX. Если вы создаете приложение MapBasic (файл MBX), можно сконструировать Ваши кнопки и команды меню так, чтобы они вызвали MapBasic процедуры в MBX. В предложении вызова используется синтаксис вызова процедуры MapBasic-программы.

После того, как Приложение на языке Visual Basic запустит MapInfo, запустите MBX, пошлав MapInfo строку вида:

- mapinfo.do "Run Application ""C:\MB\MYAPP.MBX"" "

Картография с поддержкой Visual C++ и MFC.

Создание нового проекта:

1. Запустить Visual C++ 2.x (32-битную) или 1.5x (16-битную).
2. Выполнить команду FILE > NEW для создания нового проекта или (PROJECT > APPWIZARD... в версии 1.5).

3. Запустить ассистирующую процедуру MFC App Wizard. Для первого раза выберите режим одно документного окна (SDI), а не многодокументного интерфейса (MDI). Помните, что не обязательно сразу подключать стандартную поддержку OLE. Если нужно использовать уведомления из приложения в MapInfo, то заказать поддержку OLE Automation на шаге 3 из 6 процедуры MFC App Wizard.

4. Собрать программу и запустить ее, чтобы убедиться в том, что она работает.

Добавление клиентской поддержки OLE Automation. Если Вы не заказали поддержку OLE в процедуре App Wizard, можно добавить клиентскую поддержку OLE Automation следующим образом:

1. Откройте файл STDAFX.H и добавьте строки:
- `#include <afxole.h>`
- `#include <afxdisp.h>`.
2. Откройте главный файл текста программы (т.е. имя проекта CPP) и добавьте следующие строки в начало `Cu^n^oeK7?iaApp::InitInstance`:

- if (!AfxOleInit()) {AfxMessageBox(IDP_OLE_INIT_FAILED) ; return FALSE.

3. Добавьте строку в файл строчных ресурсов (с названием имяпроекта.HC). Для этого откройте ресурс "String Table", выполните команду Resource > New String (в версии 1.5 для этого используется AppStudio). Присвойте значение ID: "IDP_OLE_INIT_FAILED" и значение Caption: "Не удалось инициализировать OLE. Проверьте правильность версии OLE-библиотек." Закройте и сохраните файл ресурсов.

Создание класса поддержки MapInfo. В диалоге PROJECT > CLASS WIZARD (BROWSE > CLASSWIZARD в версии 1.5) откройте раздел OLE Automation и нажмите на кнопку "Read Type Library". Найдите в Вашем каталоге MapInfo файл MAPINFOW.TLB. Нажмите OK, и будет создан класс, с помощью которого Можно обращаться к MapInfo через механизм управления объектами OLE (OLE Automation).

Откройте главный файл с текстом программы (с названием .CPP) и добавьте в него следующие строки.

- После всех директив # includes:
- #include "MapInfow.h".
- Сразу за объявлением "СимяпроектаApp theApp" добавьте объявление переменной:
- DMapInfo mapinfo.
- Ближе к концу Cuiw#npoeK7?iaApp::InitInstance, но перед вызовом OnFileNew(): mapinfo.CreateDispatch("MapInfo.Application").

Откройте файл MAPINFOW.H и добавьте в конец файла следующие строки: extern DMapInfo mapinfo; #include "маршрут-к-каталогу\mapbasic.h".

Если Вы работаете в Visual C++ 1.5, Нужно еще вручную добавить названия OLE-библиотек в командную строку сборки (link command line); Visual C++ 2.x делает это самостоятельно. Для этого нужно выполнить команду Options > Project... и нажать на кнопку Linker — Выберите переключатель "Common to both" в окошко "Libraries:" слова:

- compobj, storage, ole2, ole2disp, ole2nls, mfcoleui.

Тестирование. Добавьте еще одну строку в конец функции Cu^#n^oeK7?iaApp::InitInstance, сразу после вызова CreateDispatch (его добавление описано выше):

- MessageBox(0, mapinfo.GetFullName(), mapinfo.GetName(), MB_OK).

Соберите снова Вашу программу. При ее запуске Нужно увидеть сообщение с заголовком "Mapinf o Professional" и полным DOS-маршрутом к Mapinf o. Это означает, что MapInfo успешно запустилась через механизм OLE Automation. Позже тестирующую строку "::MessageBox..." можно закомментировать или удалить.

Переопределение "быстрых" меню. Встраивая Карту в Ваше приложение, Можно украсить ее всеми сервисными средствами MapInfo. Иногда этот сервис не нужен и мешает; например, стандартное быстрое меню для окна Карты включает в себя команду дублирования окна. Ее нужно удалить из быстрого меню, чтобы она не вводила в заблуждение пользователей Вашего приложения.

Для этого ближе к концу текста СимяпроектаApp::1пШпв1апсе сразу после вызова CreateDispatch добавьте следующие строки:

// удалить вызов Справочной системы:

- mapinfo.Do("Set Window Help Off");

// Удаление команды дублирования из "быстрого меню":

- mapinfo.Do("Create Menu \"MapperShortcut\" ID 17 as \"(-\\\")").

Здесь же можно добавить другие инструкции, сокращающие объем сервиса, идущего от MapInfo.

Переподчинение диалогов MapInfo. Очень важно научиться переподчинять диалоги MapInfo, появляющиеся из окна Вашего приложения, особенно, если они предназначены для заполнения пользователем. Этот прием дает уверенность в том, что диалог будет появляться над окном приложения и что окно приложения будет неактивным все время, пока пользователь заполняет диалог MapInfo. В следующем примере показано, как

переподчинить два диалога MapInfo (например, используя RunMenuCommand с заданным аргументом) и сообщения об ошибках, которые MapInfo показывает, обнаруживая непонятные события.

В тексте MainFrm.CPP, для функции CMainFrame::OnCreate надо добавить:

- после всех директив #includes:
- #include "MapInfow.h".
- В конце текста CMainFrame::OnCreate:
- char str[256];
- sprintf(str, "Set Application Window %lu", (long)(UINT)m_hWnd);
- mapinfo.Do(str).

Чтобы убедиться в том, что это сработало, добавьте строку: mapinfo.Do("Note \ "Привет от MapInfo\"); в текст функции СиямпроектаApp::InitInstance сразу после вызова OnFileNew(). Это приведет к тому, что MapInfo покажет один из своих стандартных диалогов изнутри прикладной программы.

После организации подобного переподчинения рекомендуется провести промежуточное тестирование.

Добавление окна Карты. Теперь, когда MFC-приложение заработало и Вы убедились, что к MapInfo можно обращаться через OLE Automation, пора сделать то, ради чего все это затевалось — добавить окно Карты в приложение.

Откройте диалог Project > ClassWizard (или Browse > ClassWizard в версии 1.5). Выберите класс представлений (CuManpoeKmaView) и раздел "Message Maps". В самом левом окошке списка выберите объект "CuM&npoeKmd4i&w".

В списке "Messages" выберите "WM_CREATE", нажмите на "Add Function"; выберите "WM_DESTROY", нажмите на "Add Function"; выберите "WM_SIZE", и нажмите на "Add Function".

В заголовочный файл для представлений (имя проекта УЧУ.Н) добавьте строки:

- unsigned long m_windowid;

- HWND;

m_windowhwnd.

В текст файла представлений (имя проекта Ч^Н .CPP) добавьте строки:

- после всех директив # includes:

ttinclude "MapInfow.h".

- В конструкторе (CuM&npoeKma4iew.:CuM&npoeKma4iew)

инициализируйте переменные: m_windowid = 0 ; m_windowhwnd = 0.

- В метод OnCreate добавьте следующий отрывок после вызова CView: :OnCreate:

- //стиль для окна Карты должен быть ClipChildren;

- SetWindowLong(m_hWnd, GWL_STYLE;

- GetWindowLong(m_hWnd, GWL_STYLE);

- |WS_CLIPCHILDREN);

- char str [256];

- mapinfo . Do ("Open Table \"States\" Interactive");

- sprintf (str;

- "Set Next Document Parent %lu Style 1 Map From States" , (long) (UINT)m_hWnd);

- mapinfo . Do (str);

- m_windowid = atol (mapinfo . Eval ("WindowID (0) ")) ; sprintf (str, "WindowInfo (0 , %u)", WIN_INFO_WND) ; m_windowhwnd = (HWND) atol (mapinfo. Eval (str)).

- В текст метода OnDestroy добавьте перед вызовом CView: :OnDestroy:

- if (m_windowhwnd) { :DestroyWindow(m_windowhwnd) ; m_windowhwnd = NULL; m_windowid = 0L.

В текст метода OnSize добавьте после вызова CView: :OnSize: if (m_windowhwnd && ex > 0 && cy > 0) { :MoveWindow(m_windowhwnd, 0, 0, ex, cy, TRUE).

Добавление команд меню для Карты. Все элементы меню добавляются с помощью описанной ниже процедуры. Пример показывает, как добавить команду Карта > Управление слоями.

1. Откройте файл ресурсов (имяпроекта.'КС'), откройте ресурс "Мепи"и выберите IDR_MAINFRAME. (В версии Visual C++ 1.5 нужно воспользоваться AppStudio).

2. Добавьте новое меню "Карта" и команду "Управление слоями" и сохраните КС-файл.

3. В диалоге Project > ClassWizard (Browse > ClassWizard... в версии 1.5) откройте раздел "Message Map" и выберите CuManpoeKmaView из списка "Class Name". В списке "Object ID" выберите ID-номер для создаваемой команды меню — по умолчанию это ID_MAP_LAYERCONTROL. Как только Вы это сделаете, появятся сообщения COMMAND и UPDATE_COMMAND_UI в окне "Messages". Добавьте прототипы функций, нажимая для каждого сообщения кнопку "Add Function" и принимая стандартные имена.

4. В тексте класса CuManpoeKmaView Вы увидите, что добавлены обе функции. Добавьте к текстам функций следующие строки:

- void CuManpoeKmaView: : OnMapLayercontrol ();
- {mapinfo.RunMenuCommand(M_MAP_LAYER_CONTROL); };
- void СиямяпроектаЧ±еУ1: :OnUpdateMapLayercontrol(CCcmdUI* pCmdUI) {pdndUI->Enable(m_windowid) ; }.

Добавление кнопок и процедур-обработчиков. Все кнопки на инструментальные панели могут быть добавлены описанным ниже способом. Этот пример показывает, как добавить кнопки Mapinfo: Стрелку, Ладонку и обе Лупы. Для удобства мы также добавим их в новое меню "Программы"

(или "Tools"); позволяет добавить их в инструментальную панель несколько более простым способом, используя ClassWizard.

1. Сначала, следуя приведенным выше инструкциям добавления команды меню, создайте новое меню "Программы" с четырьмя новыми элементами ("Выбрать", "Сдвинуть", "Увеличить" и "Уменьшить"). Для каждой команды определите функции `UPDATE_COMMAND_UI` и `COMMAND`, используя коды из файла `MAPBASIC.H` (`M_TOOLS_SELECTOR`, `M_TOOLS_RECENTER`, `M_TOOLS_EXPAND`, и `M_TOOLS_SHRINK`); эта процедура также описана выше. После этого скомпилируйте и протестируйте программу.

2. Открыв RC-файл для проекта, выберите растровый ресурс `IDR_MAINFRAME` и создайте растр на 64 пиксела шире (чтоб поместились 4 кнопки шириной 16-пикселей). На этом растре нужно разместить изображения четырех кнопок справа от кнопки вставки.

3. Откройте строчный ресурс и добавьте описания для каждой кнопки. При этом нужно следить за тем, чтобы ID-номера строк-описаний совпадали с номерами ранее заданных команд; строки можно задавать также, как и в файле `MAPINFOW.MNU`, например, номеру `ID_TOOLS_SELECTOR` соответствует "Выбрать объект на Карте\пСтрелка".

4. В тексте `MAINFRM.CPP` найдите массив `UINT BASED_CODE buttons[]` типа `static` и вставьте ID-константы в этот массив в том порядке, в котором они появляются в растровом ресурсе.

5. Чтобы корректно работать с кнопками, мы должны следить за тем, какая из них выбрана в данный момент. В текст файла `CuMapроеKmaView` добавьте объявление целой переменной, которая будет хранить значение выбранной кнопки: `int m_eMouseMove;`

6. Эту переменную нужно инициализировать в конструкторе классов, чтобы задать начальную нажатую кнопку на экране. Для этого нужно пользоваться заданными в `MapInfo` константами:

- `m_eMouseMove = M_TOOLS_SELECTOR.`

7. Если Вы сначала задали команды меню, то у Вас уже есть описания функций COMMAND и UPDATE_COMMAND_UI; если нет, то добавим их способом, описанном в предыдущем примере.

8. Задайте обновление интерфейса, вызывая CCmdUI::SetRadio в каждой OnUpdate-процедуре и задайте переменные m_eMouseMove соответственно процедурам ОпТсхЛзИмяИнструмента. Ваши добавления должны выглядеть примерно так:

```
- void CuMHNpoeKtndViewi: :OnToolsSelector ( ) {m_eMouseMove =  
M_TOOLS_SELECTOR; mapinf o. RunMenuCommand  
(M_TOOLS_SELECTOR); };
```

```
- void СимяпроектаЧ±етя: :OnToolsGrabber ( ) {m_eMouseMove =  
M_TOOLS_RECENTER;  
mapinfo.RunMenuConrmand(M_TOOLS_RECENTER); };
```

```
- void СимяпроектаЧ±етя: :OnToolsZoomin( ) {m_eMouseMove =  
M_TOOLS_EXPAND; mapinfo.RunMenuCommand(M_TOOLS_EXPAND); };
```

```
- void СимяпроектаЧ±етя: :OnToolsZoomout ( ) {m_eMouseMove =  
M_TOOLS_SHRINK; mapinfo.RunMenuCommand(M_TOOLS_SHRINK); };
```

```
- void CuMHNpoeKtndViewi: :OnUpdateToolsSelector (CCmdUI* pCmdUI)  
{pCmdUI->SetRadio(m_eMouseMove == M_TOOLS_SELECTOR); pCmdUI-  
>Enable(m_windowid); };
```

```
- void CuMHNpoeKtndViewi: rOnUpdateToolsGrabber (CCmdUI* pCmdUI)  
{pCmdUI->SetRadio(m_eMouseMove == M_TOOLS_RECENTER); pCmdUI-  
>Enable(m_windowid); };
```

```
- void CuMHNpoeKtndViewi: :OnUpdateToolsZoomin(CCmdUI* pCmdUI)  
{pCmdUI->SetRadio(m_eMouseMove == M_TOOLS_EXPAND); pCmdUI-  
>Enable(m_windowid); };
```

```
- void CuManpoeKmaViewi: :OnUpdateToolsZoomout (CCmdUI* pCmdUI)  
{pCmdUI->SetRadio(m_eMouseMove == M_TOOLS_SHRINK) ; pCmdUI-  
>Enable(m_windowid) ; }.
```


Обработчики ошибок MapInfo. MapInfo пересылает информацию об ошибках в приложение, использующее Интегрированную Картографию, посредством MFC-класса COleDispatchException. MapInfo возвращает код ошибки в переменной m_wCode класса COleDispatchException и описание ошибки в переменной m_str Description класса COleDispatchException. Обычно ошибки OLE общего характера передаются через класс COleException. Нужно обработать эти ошибки внутри Вашего приложения, иначе ими займется обработчик MFC-ошибок более высокого уровня, а мы получим сообщение "Command failed". Можно добавить обработчик для каждой ошибки в каждый метод DMapInfo. В следующем примере показано, как это делается для метода DMapInfo::Do.

Оригинальный текст метода DMapInfo::Do, порожденный Class Wizard, выглядит так:

```
- void DMapInfo::Do(LPCTSTR command) {Static BYTE BASED_CODE  
parms[] = VTS_BSTR;  
- InvokeHelper ( 0x6 0 0 1 0 0 Ob , DISPATCH_METHOD , VT_EMPTY ,  
NULL, parms , command).
```

Усовершенствованный метод DMapInfo::Do, включающий обработку исключительных ситуаций, выглядит так:

```
- void DMapInfo::Do(LPCTSTR command) {Static BYTE BASED_CODE  
parms [] = VTS_BSTR; try {InvokeHelper (0x60 0100 Ob,  
DISPATCH_METHOD, VT_EMPTY, NULL, parms, command); } catch  
(COleDispatchException *e) {// Обработка исключительной ситуации в Вашем  
приложении. // Ошибка помещается в e->m_wCode;  
- AfxMessageBox(e->m_strDescription);  
- e->Delete( ); } catch (COleException *e) {AfxMessageBox("Fatal OLE  
Exception!");  
- e->Delete( ).
```

Добавление поддержки сервера OLE Automation. В файле `СимяпроектаТУос.стртр` добавьте настройку диспетчера (Dispatch map) после настройки сообщений (Message map):

```
- BEGIN_DISPATCH_JV[AP(Cu^7yweK7?iaDoc, CDocument) //  
{{AFX_B18PATCH_MAP(СильжгароегетаВос) //ВНИМАНИЕ: здесь  
ClassWizard добавит или удалит настроечные макросы //Не редактируйте  
содержимое этих автоматически созданных фрагментов //  
кода //}}AFX_DISPATCH_MAP END_DISPATCH_MAP( ).
```

Добавьте в текст файла `СимяпроекпаТ)ос.cpp` конструктор `СимяпроекпаТ)ос: EnableAutomation(); Af xOleLockApp ().`

Добавьте в текст файла `СимяпроекпаТ)ос.cpp` деструктор `СимяпроекпаТ)ос:`

```
- Af xOleUnlockApp ( ).
```

Добавьте в текст файла `СимяпроектаТ)ос.1с1` раздел "Dispatch" после разметки сообщений: `// Автоматически созданные функции OLE-диспетчера //ВНИМАНИЕ: здесь ClassWizard добавит или удалит функции //Не редактируйте содержимое этих автоматически созданных фрагментов // кода //}}AFX_DISPATCH DECLARE_DISPATCH_MAP().`

В приведенном выше фрагменте кода показано, как добавить поддержку механизма управления OLE-объектами в порожденный `CDocument`-класс. Используя MFC, можно также просто добавить поддержку механизма управления OLE-объектами любому классу, порожденному `CCmdTarget`. Так для MDI-приложения Можно добавить интерфейс механизма управления OLE-объектами либо порожденному классу `CWinApp`, либо порожденному классу `CMDIFrameWnd`; оба они были порождены классом `CCmdTarget`. Причиной этого является то, что указатель `IDispatch` для уведомлений `MapInfo` можно устанавливать только раз. В MDI-приложении окна документов уничтожаются при закрытии. Если указатель `IDispatch` будет установлен для документа, то он исчезнет вместе с окном.

Добавление уведомления (callback) WindowContentsChanged. Если Вы создаете приложение с одним окном (SDI) и добавили в класс Симапроекта Оос список сообщений для диспетчера, то тогда можно установить указатель на уведомление в конструкторе СимапроектаТ)ос или в любом другом месте, где он будет вызван только один раз:

- mapinfo.SetCallback(this->GetIDispatch(FALSE)).

В диалоге Project > Class Wizard выберите раздел "OLE Automation" и из списка "Class Name" выберите класс, для которого разрешено использование OLE Automation (в нашем примере это СимапроектаТУос). Выберите "Add Method" и задайте имя метода "WindowContentsChanged", возвращаемый тип "SCODE" и список аргументов "long IWindowID". После нажатия на ОК и выхода из диалога Class Wizard автоматически обновляет файлы СимапроектаОос.CPP и файлы заголовков. В CPP-файле заполните тело функции WindowContentsChanged и обеспечьте обработку сообщений. В этой функции удобнее всего обрабатывать легенду.

Интеграция с Delphi. Запуск и связывание с сервером MapInfo. Итак рассмотрим простейший компонент для запуска и управления MapInfo (TKDMapInfoServer):

- 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 { Private
declarations } // 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;
- 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;
- 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.

```

Мы установили связь с сервером MapInfo.

У сервера MapInfo есть метод Do - он предназначен для отправки команд MapBasic серверу точно также как если бы пользователь набирал их в окне MapBasic-а самой программы MapInfo.

У сервера MapInfo есть метод Eval- он предназначен для получения значения функций после отправки команд MapBasic серверу.

Дополнительная информация. Для ознакомления Интегрированной Картографии, предоставлены программы, поставляемые в комплекте со средой разработки MapBasic:

- SAMPLES\VB\FINDZIP: программа на языке Visual Basic, используемая как примерный образец для разработки собственных приложений.

- SAMPLES\VB\VMAPTOOL: Visual Basic-программы, демонстрирующие более сложные примеры интеграции с Visual Basic 4.0 Professional Edition.
- SAMPLES\MFC\FINDZIP: Простое приложение, использующее MFC.
- SAMPLES\PWRBLDR\CAPITALS: Простое приложение на языке PowerBuilder.
- Внимание: это 16-битное приложение, и Вам нужно иметь runtime-версию PowerBuilder, чтобы увидеть, как оно работает.
- SAMPLES\DELPHI\TABEDMAP: Простое приложение на языке Delphi.
- В каталоге SAMPLES Можно найти другие приложения, в том числе и не указанные в данном Руководстве.