Урок 4. Создание диалогов (VC7)

1. Диалоги бывают стандартные (ими можно просто пользоваться, ниже будут приведены примеры, как) и свои собственные. Для создания своего диалога надо сделать следующие действия: 1) Создать ресурс диалога (т.е. его вид, кнопочки и прочее); 2) создать класс диалога, привязав его к созданному ресурсу, 3) создать члены класса диалога, связанные с элементами диалога (кнопками, переключателями и т.д.) и 4) в одном из классов (например C…Doc или C…View) создать объект класса диалога и вызвать у него функцию, приводящую к изображению диалогового окна.
   Теперь подробнее.
2. Создание ресурса диалога: меню Project|Add Resource, выделяете Dialog и выбираете New. Далее, появившееся изображение диалогового окна можно редактировать, перетаскивать на него из Toolbox (если нет списка элементов, то нажать ToolBox и в нем Dialog Editor). Созданному диалогу будет присвоен идентификационный номер: IDD_DIALOG1, например). После добавления элементов диалога им надо дать осмысленные названия (Caption) и осмысленное название идентификатор, с ним связанного. Это делается так: вы делаете элемент активным, щелкнув по нему мышкой. В окне Properties появляются свойства этого элемента. В этих свойствах вы меняете свойство Caption и ID (например, ID будет состоять из IDC_ и слова, согласованного с Caption, можно просто повторить содержание Caption, если оно недлинное). Нам понадобятся, например, элементы диалога Edit Control, Static Text и группа (несколько) Radio Buttons. При создании группы из Radio Buttons у первой из Radio Buttons в свойствах ставится Group – true (это означает, что она главная в группе и реакция на группу будет связана с ее ID). Суть группы из Radio Buttons состоит в том, что может быть выбран только один элемент из группы. Кнопки, которые вы хотите, чтобы были группой, обведите с помощью добавления элемента Group Box вокруг них (от этого они группой не станут, но будет красиво). Выравнивание группы элементов делается с помощью их выделения и меню Format (Align и Space Evenly). Важен порядок обхода всех элементов диалогового окна (по клавише Tab). Чтобы узнать и изменить порядок, надо вызвать Format – Tab order и пощелкать по элементам в нужном порядке. Группа Radio Buttons зависит от порядка элементов: она начинается с элемента со свойством Group – True и заканчивается, когда встречается следующий элемент с таким свойством.
3. Создание класса диалога. В появившемся окне с видом диалога двойным щелчком мыши в окне (вне элементов диалога) вы попадаете в MFC ClassWizard. Поле BaseClass надо исправить на CDialog, поставить в Class Name имя класса далога, связанное со смыслом диалога (например, CFormatDlg). Внизу автоматически сгенерируются имена файлов, в которых будут находиться прототип класса диалога и его реализация. Поле Dialog ID показывает, с каким ресурсом (видом) связан этот диалог. <Finish>.
4. Теперь надо связать с нужными элементами (в нашем случае – с окошком типа Edit Control и с группой radio кнопок) члены класса окна. Для этого на вкладке Resources двойным щелчком по идентификатору диалога (ID_DIALOG1) попадаем снова в редактор созданного уже диалога. Там выделяем нужный элемент, по правой кнопке из контекстного меню выбираем Add Variable и попадаем в Add Member Variable Wizard. (Иначе туда же можно попасть через добавление переменной в класс, в данном случае - диалога, как это делалось ранее). Там должна стоять галочка Control Variable и в поле ID стоять идентификатор элемента диалога. С каждым элементом диалога можно связать значение, а можно связать класс-окно для управления видом этого элемента. Чтобы связать значение, в поле Category выбираем Value. После чего, не меняя access - public, задаем имя переменной в поле Variable name. Имя обычно начинается с m_ и затем, например, повторяется caption элемента. Для создания класса в поле Category выбирается Control и задается другое имя. Например, если мы с окошком Edit свяжем Value, то имя для control выберем такое же, только добавив в конец Edit. Для группы из radio-кнопок переменная связывается именно с той radio-кнопкой, у которой поле group имеет значение true, а имя переменной дается на основе Caption у Group Box, который окружает группу radio-кнопок. Для полей категории Value для Edit в поле Variable Type устанавливается тот тип, который задается с помощью данного окошка. Если мы будем задавать там ширину линии для рисования, то это будет int, если длину стороны квадрата, то double. Для группы radio-buttons задается тип int. При создании Value, связанного с Edit Box, в полях Min value и Max value можно указать границы, при задании значение вне которых будет выскакивать стандартное сообщение (но это может быть неудобно при вводе).
5. Пользоваться классом диалога можно как самым обычным классом: через #include подключаете файл .h с описание класса диалога, заводите объект этого класса (например, Dial d;)и вызываете функцию класса d.DoModal();. Функция возвращает значение, соответствующее способу вашего выхода из диалога. Так как выход из диалога происходит не всегда по нажатию кнопки OK, то лучше делать проверку if(d.DoModal()==IDOK) {действия}. Тогда при нажатии другой кнопки (Cancel) действия происходить не будут. Обращение к полям диалога происходит с помощью заведенных переменных, связанных с полями диалога. После вызова диалога можно узнать о результате также по значениям

переменных. Для группы radio-кнопок заведенная для них переменная принимает значения 0, 1, … соответственно номеру выбранной radio-кнопки в группе.

6. Использование стандартных диалогов (уже существующие классы, которыми можно пользоваться также, как и другими диалогами).
   а) Работа с файлами, класс CFileDialog.
   Если диалог на открытие файла, то в конструкторе нужно указать TRUE:
   `CFileDialog FileO(TRUE);`
   Если диалог на сохранение файла, то в конструкторе нужно указать FALSE:
   `CFileDialog FileS(FALSE);`
   После вызова диалога (FileO.DoModal();) к выбранному в диалоге имени файла можно добраться с пом. функции FileO.GetFileName();(только имя) или FileO.GetPathName(); (имя вместе с путем)
   б) Задание цвета
   ```
   CColorDialog dlg(RGB(255,0,0));
   if(dlg.DoModal()==IDOK)
   {
   COLORREF m_PenColor=dlg.GetColor();
   }
   ```
7. Заводить объект класса диалога можно прямо перед тем, как вы его вызываете с помощью DoModal. Но тогда все изменения значений, которые вы сделаете, не сохранятся. Чтобы при новом вызове диалога там были те значения, которые были установлены при предыдущем его вызове, объект класса диалог надо завести как член класса, функции которого вызывают диалог. Тогда в конструкторе вызывающего класса его надо инициализировать, а при каждом новом вызове диалога в нем сохранятся предыдущие установки. То же самое можно сделать, заведя в вызывающем классе дубликаты всех переменных класса диалога, перед вызовом диалога переменные диалога заполнять значениями переменных вызывающего класса, а после вызова диалога наоборот, переменные вызывающего класса заполнять значениями переменных диалога. В любом случае, надо не забыть проинициализировать поля диалога разумными значениями, чтобы при первом вызове диалога выбор, предложенный там, был осмысленным.
8. **Задание**: Сделать так, чтобы при вызове пункта меню Format вызывался диалог, в котором можно установить ширину линии и выбрать один из трех цветов: синий, зеленый или красный. После чего в окне рисовалась бы линия выбранной толщины и цвета. По пункту меню Color должен вызываться стандартный диалог выбора цвета, после чего рисоваться прямоугольник выбранного цвета.
9. Обмен данными внутри диалога:
   Предположим, что изменение некоторого элемента диалога должно привести к изменению другого элемента диалога. Например, пусть в окне диалога есть два элемента Edit Box и им соответствуют переменные m_edit1 и m_edit2. Два раза щелкнув по первому Edit Box (находясь в редакторе ресурсов), в классе диалога можно создать функцию реакции на изменение содержимого этого Edit Box (также можно создать функцию, вызываемую при щелчке по radio-кнопке). Например, мы хотим, чтобы при изменении первого Edit Box, во втором Edit Box возникало число, в два раза большее. Тогда внутри функции OnEnChangeEdit1() мы пишем m_edit2=m_edit1*2; Но этого мало. Дело в том, что отдельно существуют числа, которые видны в диалоге и отдельно существуют переменные, относящиеся к элементам диалога. Чтобы перенести видимую информацию в переменные, нужно вызвать UpdateData(TRUE); а чтобы значения переменных перенести в диалог - UpdateData(FALSE); Таким образом, получится:
   ```
   UpdateData(1);
   m_edit2=m_edit1*2;
   UpdateData(0);
   ```
10. Если вы хотите в процессе работы с диалогом изменять его элементы (не значения – а именно элементы, например, сделать недоступным окно или кнопку, изменить их расположение), то с данным элементом диалога можно связать переменную со значением Control в поле Category. Свяжем с ID_EDIT2 переменную m_ce как Control. Тогда мы сможем, например, написать в функции OnEnChangeEdit1() m_ce.EnableWindow(0); и при изменении в первом Edit Box второй Edit Box станет недоступным.
11. Другой пример: вы хотите, чтобы при выходе значения из Edit Box за границы значение устанавливалось в максимально возможное, то в OnEnChangeEdit1() можно написать
    ```
    UpdateData(TRUE);
    if (m_edit1>4) m_edit1=3;
    UpdateData(FALSE)
    ```
12. Таким же способом можно проверять все поля в форме в реакции на щелчок на кнопку OK. Если что-то не устраивает, то (1) можно исправить или выдать предупреждение с помощью и (2) при необходимости можно не выходить из диалога, не вызывая CDialog::OnOk внутри реакции.
13. **Задание:** Сделать так, чтобы был контроль за значение ширины линии в границах от 1 до 3. Сделать так, чтобы при выборе синего цвета изменение ширины линии стало недоступным. При выборе другого цвета – снова доступным.
14. **ПОЛЕЗНО:** static подсоединение MFC, TRACE для отладки, утечки памяти в окне Output