**Лабораторна робота № 5**

**Створення проекту з використанням технології Drag&Drop та контекстних підказок**

Мета: навчитися використовувати можливості технології Drag&Drop у прикладних програмах, додавати до програм контекстні підказки.

Порядок роботи:

1. Користуючись рекомендованою літературою до лабораторної роботи створити проект відповідно до індивідуального завдання.
2. Оформити звіт для захисту лабораторної роботи за зразком
3. назва роботи
4. мета роботи
5. порядок роботи
6. короткі теоретичні відомості
7. алгоритм розв’язку задачі
8. тексти відповідних модулів проекту
9. аналіз отриманих результатів та висновки

**Технологія Drag&Drop**

Технологія Drag&Drop полягає у використанні у Windows-програмах перетаскування за допомогою миші інформації з одного об'єкта (джерела) в інший (приймач). Можна переміщати файли між папками, переміщати самі папки і т. ін.

Усі властивості, методи та події, пов'язані з технологією Drag&Drop, визначені в класі **TControl**, що є базовим класом (прабатьком) усіх візуальних компонент Delphi. Тому вони є спільними для всіх компонент.

Початок перетаскування визначається властивістю **DragMode**, що може встановлюватися в процесі проектування чи програмно рівною **dmManual** або **dmAutomatic**. Значення **dmAutomatic** (автоматичне) визначає негайний початок процесу перетаскування при натисканні користувачем кнопки миші над об'єктом. Однак у цьому випадку подія OnMouseDown, пов'язана з натисканням користувачем кнопки миші, для цього об'єкта взагалі не настає.

Значення **dmManual** (ручне) говорить про те, що початок процесу перетаскування визначає програміст. Для цього він повинен у відповідний момент викликати метод **BeginDrag**. Наприклад, він може помістити виклик цієї функції в обробник події **OnMouseDown**, що настає в момент натискання кнопки миші. У цьому обробнику можна попередньо перевірити якісь умови (режим роботи ужитку, натискання тих чи інших кнопок миші і додаткових клавіш) і при виконанні цих умов викликати **BeginDrag**.

Нехай, наприклад, процес перетаскування повинен розпочатися, якщо користувач натиснув ліву кнопку миші та клавішу Alt над списком ListBox1. Тоді властивість **DragMode** цього об'єкта слід установити в **dmManual**, а його обробник події **OnMouseDown** може мати вигляд:

procedure TForm1.ListBox1MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

begin

if (Button = mbLeft) and (ssAlt in Shift)

then (Sender as TControl).BeginDrag(false);

end;

Параметр **Button** обробника події **OnMouseDown** показує, яка кнопка миші була натиснута, а параметр Shift є множиною, що містить позначення натиснутих у цей момент кнопок миші і допоміжних клавіш клавіатури. Якщо натиснута ліва кнопка миші та клавіша Alt, то викликається метод **BeginDrag** даного об'єкта (Sender).

У функцію **BeginDrag** передається значення False. Це означає, що процес перетаскування почнеться не одразу , а тільки після того, як користувач зрушить мишу з натиснутою при цьому кнопкою. Це дозволяє відрізнити просте натискання клавіші миші від початку перетаскування.

Якщо ж передати в **BeginDrag** значення True, то перетаскування почнеться негайно.

Коли почався процес перетаскування, звичайний вигляд курсору змінюється. Поки він переміщається над формою чи об'єктами на формі, які не можуть прийняти інформацію, він звичайно має вигляд **crNoDrop**. Якщо ж він переміщається над об'єктом, що готовий прийняти інформацію з об'єкта-джерела, то набуває вигляду, що залежить від властивості **DragCursor** об'єкта-джерела (а не того об'єкта, над яким переміщається курсор!). За замовчуванням ця властивість дорівнює crDrag, що відповідає зображенню білого прямокутника з рамкою.

У процесі перетаскування об'єкти, над якими переміщується курсор, можуть інформувати про готовність або неготовність прийняти інформацію від об'єкта-джерела. Для цього в об'єкті повинен бути передбачений обробник події **OnDragOver**, що настає при переміщенні над даним об'єктом курсору, що перетаскує деякий об'єкт. У цьому обробнику слід перевірити, чи може даний об'єкт прийняти інформацію об'єкта-джерела, і, якщо не може, надати значення **False** переданому в обробник параметру **Accept**. За замовчуванням цей параметр дорівнює True, що означає можливість прийняти інформацію від об'єкта-джерела. Обробник для списку може мати, наприклад, наступний вигляд:

procedure TForm1.ListBox1DragOver(Sender, Source: TObject; X, Y: Integer;State: TDragState; var Accept: Boolean);

begin

if(Sender as TControl<> Source) then Accept := Source is TListBox else Accept := False;

end;

У ньому спочатку перевіряється, чи не є даний об'єкт (**Sender**) і об'єкт-джерело (**Source**) тим самим об'єктом.

Це зроблено, щоб уникнути перетаскування інформації усередині того самого списку.

Значення параметра **Accept**, що задається в обробнику події **OnDragOver**, визначає вигляд курсору, що переміщається при перетаскуванні над даним об'єктом. Цей вигляд показує користувачу, чи може даний об'єкт прийняти інформацію, що передається. Якщо в об'єкті не описаний обробник події **OnDragOver**, то вважається, що даний об'єкт не може прийняти інформацію від об'єкта-джерела.

Процедура прийому інформації від об'єкта-джерела, записується в обробнику події **OnDragDrop** об'єкта-приймача. Ця подія настає, якщо після перетаскування користувач відпустив клавішу миші над даним об'єктом. В обробник цієї події передаються параметри **Source** (об'єкт-джерело) і X і Y координати курсору. Якщо продовжити приклад перетаскування інформації з одного списку в іншій, то обробник події **OnDragDrop** може мати вигляд:

procedure TForm1.ListBox1DragDrop(Sender, Source: TObject; X, Y: Integer);

begin

(Sender as TListBox).Items.Add((Source as TListBox).Items[(Source as TListBox).ItemIndex]);

end;

У цьому обробнику рядок, що виділений у списку-джерелі **(Source as TListBox).Items[(Source as TListBox).ItemIndex]**, додається в список-приймач методом **(Sender as TListBox).Items.Add**. Використовується операція **as**, яка дозволяє розглядати параметри **Sender** і **Source** як вказівники на об'єкт класу **TListBox**. Це робиться тому, що ці параметри є оголошені у заголовку процедури як вказівники на об'єкти класу **TObject**. Але в класі TObject немає властивостей Items і ItemIndex, які нам потрібні. Ці властивості є визначені в класі **TListBox**, що є нащадком **TObject**. Тому з параметрами **Sender** і **Source** у даному випадку треба оперувати як із вказівниками на об'єкти **TListBox**, що і виконує операція **as**.

У даному випадку можна було б не використовувати параметр **Sender**, замінивши **(Sender as TListBox)** просто на **ListBox1**. Але запис оператора в загальному вигляді за допомогою параметра **Sender** дозволяє скористатися таким обробником і для інших об'єктів **ListBox**, якщо вони є в ужитку.

Після завершення чи припинення перетаскування настає подія **OnEndDrag**, в обробнику якої можна передбачити якісь додаткові дії. Також пов'язана з перетаскуванням подія **OnStartDrag**, що дозволяє зробити якісь операції на початку перетаскування, коли іншим способом цей момент не можна зафіксувати.

Таким чином, якщо в ужитку є кілька списків і потрібно задати можливість копіювання рядків кожного з цих списків у будь-який іншій, те це вимагає двох операцій:

1. Написати для одного списку обробник події **OnDragOver**. Для всіх інших списків указати для події **OnDragOver** цей же обробник (виділивши у формі всі списки, що залишилися).

2. Написати аналогічним чином для всіх списків обробник події **OnDragDrop**.

Якщо починати перетаскування потрібно тільки при виконанні якої-небудь додаткової умови, наприклад, при натисканні клавіші Alt, то буде потрібно задати для всіх списків значення властивості **DragMode**, рівне **dmManual** і написати обробник події **OnMouseDown**.

**Контекстна підказка**

програмування контекстна підказка

Контекстна підказка - це невелике віконце з текстом, яке з'являться за деякий час після наведення курсору миші на певний об'єкт на формі. Для різних об'єктів текст може бути різним, тому підказка називається контекстною.

Для створення такої підказки у програмі слід задати значення властивості **Hint** (рядкового типу) для відповідного об'єкта, а також установити у **True** значення його властивості **ShowHint.** Ці дії можна виконати як у процесі проектування так і програмно.

**Завдання**

Написати програму, яка показує можливості технології Drag&Drop для заданих об'єктів - джерела та приймача. Необхідну інформацію для користувача програми оформити у вигляді контекстної підказки.

**Варіанти завдання**

1. Edit StringGrid.
2. ListBox ListBox.
3. StringGridMemo.
4. MemoRichEdit.
5. ListBoxListView.
6. LabelMemo.
7. ListViewListView.
8. StringGridStringGrid.
9. ListBoxMemo.
10. ShapeShape.
11. ImageImage.
12. Edit ListBox.