Вообще-то название "контрольная панель" не очень подходит этому компоненту. По большому счету это просто контейнер тулбаров, панелей инструментов, если хотите. Этих самых панелей в самой IDE C++Builder до фига. Они настраиваются, перемещаются, в общем делать с ними можно практически все.

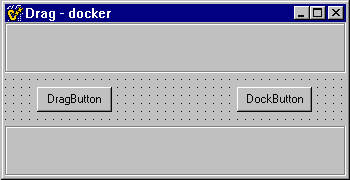
Компоненты, которыми владеет панель ControlBar, принимают специфическую форму дочерней панели - появляется рукоятка, за которую компонент можно таскать по всей области контейнера, и даже перетаскивать на другие docking site. Сейчас объясню последнее. Docking site - это понятие, определяющее способность данного объекта быть областью хранения, причалом для другого объекта. Docking site может быть любой производный TWinControl.

Здесь нужно отметить разницу Drag'n'Drop и Dock'n'Drop. Во-первых, при dock объект всегда перемещается, а не копируется. Во-вторых, форма курсора. При dock по умолчанию курсор остается обыкновенным, а вместо объекта отражается его ClientRect-рамка. При drag отображается (опять-таки по умолчанию) курсор с листиком. Все видели, все знают.

Для того, чтобы подготовить dock-drop операцию, необходимо сделать следующее. Для причала (в данном случае объекта TControlBar) установить значение DockSite в true, DragKind в dkDock. Для перетаскиваемого объекта свойство DragKind также в dkDock, а свойство DragMode в dmAutomatic. В общем-то это и все. Теперь при перетаскивании объекта на панель он на ней будет оставаться. Если же его с панели вытащить, то он будет плавать в отдельной форме, если, конечно, его на другую панель не перетащить.

Но нам же нужен контроль! А то каждый всякую неполагающуюся вещь будет перетаскивать ;). Для этого существую соответствующие события. Для причала это OnDockOver и OnGetSiteInfo, а для перетаскиваемого объекта OnStartDock и OnEndDock. Чтобы Вы не подумали, что я как-то дискриминирую Drag-Drop опреацию, скажу, что для нее нужно практически то же самое, только DragKind нужно поставить в dkDrag, а события будут соответственно OnDragOver, OnStartDrag, OnEndDrag. Как и следовало ожидать, принципы работы с обеими техниками довольно схожи.

В общем я тут говорю, говорю... Читатель скажет (наверное) - "Пора бы и пример привести!". Что ж, пример можно. Допустим, так. Две панели, одна сверху, одна снизу. Две кнопки. Одна таскается по методу drag'n'drop, другая по методу dock'n'drop. Еще сама форма. Вообще, форма - один из наиболее типичных производных TWinControl. Поэтому, все, что сказано выше, и к ней относится также. Проделываем все махинации над свойствоами, как это было сказано выше. Делаем форму.



Вычуров никаких нет. Не почтите скупостью, но обработчик OnDragOver и OnDragDrop я сделал одинаковым для всех трех причалов. Особого фильтра я для каждого случая не делал. Прочтите комментарии.

void \_\_fastcall TForm1::FormDragDrop(TObject \*Sender, TObject \*Source,

int X, int Y)

{

//здесь, когда объект брошен,

//автоматически происходит его перенос на причал

TControl\* drop=(TControl\*)Source;

TWinControl\* dropsite=(TWinControl\*)Sender;

drop->Parent=dropsite;

drop->Top=Y;

drop->Left=X;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::ControlBar1GetSiteInfo(TObject \*Sender,

TControl \*DockClient, TRect &InfluenceRect, TPoint &MousePos,

bool &CanDock)

{

//здесь проверяется, можно ли данный объект бросить на причал

//методом dock.

if(DockClient==DockButton)CanDock=false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::FormDockOver(TObject \*Sender,

TDragDockObject \*Source, int X, int Y, TDragState State,

bool &Accept)

{

//здесь также проверяется, можно ли принять объект

Accept=true;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::FormDragOver(TObject \*Sender, TObject \*Source,

int X, int Y, TDragState State, bool &Accept)

{

//можно ли принять через drag'n'drop

if(Sender==DragButton)Accept=true;

}