Dynamic menu

Dynamically show or hide menu item

Dynamic menu handling is made quite easy with the S60 platform, basically all you need to do is to override the *DynInitMenuPaneL()* method in your application view class with which the menu is associated and link application against eikcoctl.lib. Following sample code shows an example of the *DynInitMenuPaneL()*method implementation:

```
void CMyView::DynInitMenuPaneL(TInt aResourceId, CEikMenuPane* aMenuPane)
  {

    if (R_MAIN_MENU == aResourceId)
      {
        if (iRegistered)
          {
            aMenuPane->SetItemDimmed(ERegisterCmd, ETrue);
          }

      }
    else if (R_ADD_MENU == aResourceId)
      {
        CEikMenuPaneItem::SData data;
        data.iText.Copy(KtxAddCommand);
        data.iCommandId = EAddCommand;
        data.iCascadeId=0;
        data.iFlags=0;
        data.iExtraText=KNullDesC;

        aMenuPane->AddMenuItemL(data);
      }
  }
```

Note that before you modify the menu pane you need to check which menu pane of your application you are using, this one can be done be checking which menu pane resource ID is supplied to the function. Note also that the *aResourceId* is the resource id for MENU_PANE not the resource id for MENU_BAR.

This function is called just before the menu pane is shown in the application, Calling *SetItemDimmed()* with **ETrue** will dim the menu item, which is S60 mean hiding the menu item from the list, thus users will not see any of the dimmed items.

You can use all functions defined for the **CEikMenuPane** when you are in the *DynInitMenuPaneL()*method. For example you can also add menu items as shown in the sample code.

The resource definition used with the sample code looks like this:

```
RESOURCE MENU_BAR r_main_menubar
  {
    titles =
      {
        MENU_TITLE
          { menu_pane = r_main_menu; txt = "Options";}
      };
  }
RESOURCE MENU_PANE r_main_menu
  {
    items =
      {
```

```
        MENU_ITEM
          { command = ERegisterCmd; txt= "Register";},
        MENU_ITEM
          { command = ENMsg;cascade=r_add_menu; txt="Add";},
        MENU_ITEM
          { command = EAbout; txt="About";}
      };
  }

RESOURCE MENU_PANE r_add_menu
  {
    items =
      {
        MENU_ITEM
          { command = EAnotherCmdA; txt= "Another A";},
        MENU_ITEM
          { command = EAnotherCmdB; txt= "Another B";}
      };
  }
```

Dynamically change entire menu bar

If you want to change your entire menu content from R_MAIN_MENU to R_ADD_MENU, you should create new MENU_BAR item in your resource file, say like the following:

```
RESOURCE MENU_BAR r_add_menubar
 {
 titles =
  {
  MENU_TITLE { menu_pane = r_add_menu; }
  };
 }
```

Then you could call this function in your cpp source.

```
iEikonEnv->AppUiFactory()->MenuBar()->SetMenuTitleResourceId(R_ADD_MENUBAR);
```

But please be noted, we'd better do not call this from the DynInitMenuPaneL() function, because this behavior will cause weired phenomenons that even the wizards from 2-6 Boundary Row, Southwark, London could not help so much. And calling simply SetMenuTitleResourceId(R_ADD_MENU) might cause BUFL 15 panic code.

Dynamically add cascading menus

You cannot directly create cascading menus in S60, but you can cheat a bit to achieve this.

You can define in your resource files enough empty menus for your needs (say, 10) and then when you get the *DynInitMenuPaneL()* call, you can add these to the menu. Then when you get another call for one of these, you can populate the inner menu. This way you can get the effect of creating dynamic cascading menus. Just remember to add enough empty menus.

Following piece of code to add cascade dynamically:

```
CEikMenuPaneItem::SData item;
item.iCommandId = EAddSubMenu;
item.iFlags = 0;
item.iCascadeId = R_ADD_SUB_MENU;
item.iText.Copy(_L("Cascade"));
```

```
aMenuPane->AddMenuItemL(item);
```