How to display drop down/fly out menu using Symbian C++

Description

Menu options are a critical component of a mobile application and in turn play a crucial role in ensuring a high usability index if used properly and at the right place. The menu option should be displayed only when the context for which they are being displayed is relevant i.e. say for a selected item on a list view they should be shown only when it applies to the selected item, instead of displaying them by default only to display a error note to the user that the option doesn't apply to this case etc. The user is rather impatient when it comes to interactions with the mobile device and hence should be shown only content and asked for input if it absolutely makes sense in the given context.

Lot of times it makes sense to display a menu option like a drop down menu/fly out menu which allows the user to select a particular option, but it is also important to ensure that they are **displayed at the correct place** as well. For instance if the user has selected an item on the list view it would be very nice to show the drop down menu right there **in situ**, just below the selected item as this would look so much more relevant and elegant from a usability perspective. Also in the case of context menu it would be nice to **show relevant icons** in some cases for instance when the options have to do with Call, showing a telephone icon or a message icon in the case the menu option is SMS/MMS looks nice from a usability perspective.

Advantages

A custom drop down menu allows changing the look and feel to cater to the application context in which it has been called. It also saves a lot of space as the pop up is calculated and sized according to the number of options/content being displayed.

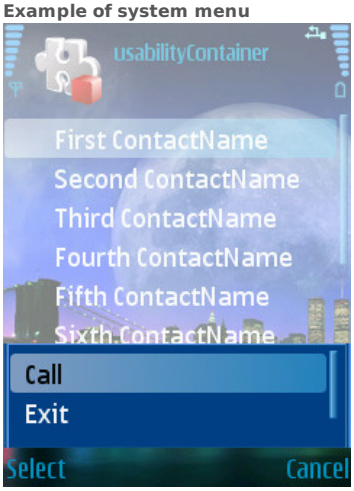
System Menu on Symbian C++

The built in menu system on Symbian C++ doesn't allow the user a great deal of flexibility in terms of where they are displayed, how they are displayed and other usability aspects. For instance they don't allow setting of custom icons like the telephone, messaging icons etc as elaborated earlier, they don't allow changing the positioning either to display them in situ something which would be so much more elegant from a user's point of view.

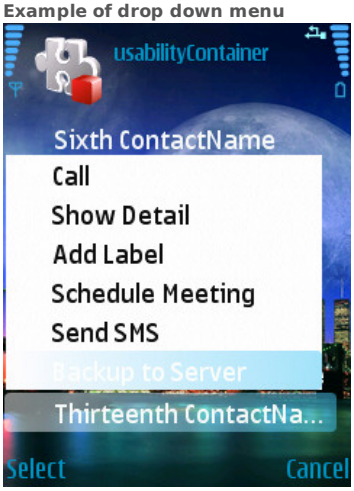
For more details on menu implementation on Symbian C++ check Menu Link

Comparison of System menu & custom drop down implementation

Example of system menu



Example of drop down menu



Use When

The drop down/fly out menu has its own advantages and disadvantages and hence they should be used cautiously and in the right context. Some typical usage could be:-

- Options on a selected item on the list view
- Camera/media kind of applications, as detailed Here
- Custom implementation needed where you want to display icons, colors as per your context/requirements.

(C) Copyright Nokia 2013. All rights reserved.

Implementation

The code snippet below explains the implementation of a drop down/fly out menu on a list view, using Symbian C++ . We subclass a CAknListQueryDialog to implement a drop down menu as per our needs. The obvious details have been omitted for clarity and a complete working code can be downloaded from Usability.zip

The cpp file for the container, with explanation of the main functions follows:-

Right now the drop down menu is created on the press of the select/Center soft key on the device (EStdKeyDevice3), and when the drop down menu is being displayed the key events are passed to it so that the user interaction like key up/down, selection etc are can be honored in the right context. When the user presses center key on the drop down menu, the menu is dismissed and the selected option is handled.

```
TKeyResponse CUsabilityContainer::OfferKeyEventL(
  const TKeyEvent& aKeyEvent,
  TEventCode aType )
 {
 if(iDropDownDlg && aType == EEventKey)
  {
  TKeyResponse Ret = EKeyWasNotConsumed;
  switch(aKeyEvent.iCode)
   {
   case EKeyUpArrow:
   case EKeyDownArrow:
    {
// Scroll implementation to make the current selected option visible
    Ret = EKeyWasConsumed;
    iDropDownDlg->ListBox()->ScrollToMakeItemVisible(iDropDownDlg->ListBox()->CurrentItemIndex());
    iDropDownDlg->ListBox()->OfferKeyEventL(aKeyEvent, aType);
    }
   break;
   default:
    break;
   }
  if(aKeyEvent.iScanCode == EStdKeyDevice3)
   {
   Ret = EKeyWasConsumed;
   CEikListBox* listBox = iDropDownDlg->ListBox();
   iDropDownDlg->DismissQueryL();
   }
  return Ret;
  }

 if(aKeyEvent.iScanCode == EStdKeyDevice3 && aType == EEventKey)
  {
  // CSK press, lets show the drop down menu
  ShowMenu();
  return EKeyWasConsumed;
  }
 if ( iFocusControl != NULL
  && iFocusControl->OfferKeyEventL( aKeyEvent, aType ) == EKeyWasConsumed )
  {
  return EKeyWasConsumed;
  }
 return CCoeControl::OfferKeyEventL( aKeyEvent, aType );
 }
```

Utility function that sets the icons to the drop down menu, write your own logic to pick up the icons you want to display on the menu.

```
void CUsabilityContainer::SetupListBoxIconsL(TBool aDropDown)
 {
 CArrayPtr< CGulIcon >* icons = NULL;
 icons = new (ELeave) CAknIconArray( 1 );
 CleanupStack::PushL( icons );
 // TODO add your logic to get the icons to the icon array
 CleanupStack::Pop( icons );
 iDropDownDlg->SetIconArrayL(icons);
 }
```

Create the drop down menu in this function after specifying the content, icon and the position you want to draw it at/from.

```
void CUsabilityContainer::ShowMenu()
 {
 TInt index = 0;
 iDropDownDlg = new( ELeave ) CDropDownDialog( &index );
 iDropDownDlg->PrepareLC(R_DROP_DOWN_MENU);

 CDesCArray* menuArray = new ( ELeave ) CDesCArrayFlat(2);
 menuArray->AppendL(_L("0\tCall"));
 menuArray->AppendL(_L("0\tShow Detail"));

 iDropDownDlg->SetItemTextArray(menuArray);
 iDropDownDlg->SetOwnershipType(ELbmOwnsItemArray);

 SetupListBoxIconsL(ETrue);

 // Get starting y pos
  TRect r = CEikonEnv::Static()->AppUiFactory()->ClientRect();
  TInt starty = 30; // Right now this is hard-coded change it according to your requirements
  TInt m = r.iBr.iY - 20;

 TSize s = iDropDownDlg->ListBox()->MinimumSize();
 if (s.iHeight > m)
  {
  s.iHeight = m;
  TRect r(TPoint(0,0),s);
  s.iHeight = r.Height();
  }
 // Set position
 if (starty + s.iHeight > r.iBr.iY)
  {
  starty = r.iBr.iY - s.iHeight;
  if (starty < 0)
   {
```


(C) Copyright Nokia 2013. All rights reserved.

```
    starty=0; // should not happen.
    }
   }
  TInt startx = r.iBr.iX - s.iWidth;
  if (startx < r.iTl.iX)
   {
   startx = r.iTl.iX;
   }
  iDropDownDlg->SetRectInfo(TPoint(startx,starty),s); // Set the rect
  iDropDownDlg->ListBox()->SetCurrentItemIndexAndDraw(0);

  TBool returnVal = iDropDownDlg->RunLD();
  iDropDownDlg = NULL;
  if(returnVal == -2 || returnVal == -6)
   {
 // User pressed CSK or the select option; we don't want to do anything if menu was cancelled, handle the choices from here!
   }
  }
```

Pass the key events to the container's offer key event where the logic to handle the key events has been implemented.

```
TKeyResponse CDropDownDialog::OfferKeyEventL(const TKeyEvent& aKeyEvent, TEventCode aType)
 {
 return CCoeControl::OfferKeyEventL( aKeyEvent, aType );
 }
```

Set the size and position of the drop down menu as per our requirements, gets called by the framework.

```
void CDropDownDialog::SetSizeAndPosition(const TSize &aSize)
 {
 CAknListQueryDialog::SetSize(iSize);
 ListBox()->SetSize(iSize);
 CAknListQueryDialog::SetPosition(iPosition);
 }
```

The drop down menu is ready to be shown, let's check if we need a scroll bar to display all the options or not

```
void CDropDownDialog::PostLayoutDynInitL()
 {
 CEikListBox* lb = ListBox();
 if (lb)
  {
  lb->ScrollBarFrame()->SetScrollBarVisibilityL(CEikScrollBarFrame::EOff,
    (lb->CalcHeightBasedOnNumOfItems(lb->Model()->NumberOfItems()) >
    lb->Rect().Height()?CEikScrollBarFrame::EAuto:CEikScrollBarFrame::EOff) );
  }
 }
```

Pass the size changed to the parent dialog

```
void CDropDownDialog::SizeChanged()
 {
 CAknListQueryDialog::SizeChanged();
 }
```

The header file detailing the sub classing from the CaknListQueryDialog to implement the drop down menu:-

```
class CDropDownDialog : public CAknListQueryDialog
    {
    public:
     CDropDownDialog(TInt* aIndex) :CAknListQueryDialog(aIndex)
            {
            // No implementation required
            }
     virtual ~CDropDownDialog()
      {

      }
     virtual void PostLayoutDynInitL();
     TKeyResponse OfferKeyEventL(const TKeyEvent& aKeyEvent, TEventCode aType);
     void SetSizeAndPosition(const TSize &aSize);
     void SetRectInfo(TPoint aPosition, TSize aSize) {iPosition=aPosition;iSize=aSize;};
     void SizeChanged();

protected:
  TPoint iPosition;
  TSize iSize;
     };

class CUsabilityContainer : public CCoeControl
{
CDropDownDialog* iDropDownDlg;
};
```

Post Condition

The drop down menu is displayed as per the size/icons/layout specified in the code.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.