How to implement fisheye view in Symbian C++

Introduction

This example shows how to implement a fish eye view using Symbian C++/S60 classes. The fisheye window allows user to view more content when highlighted while all the other items are displayed as is. This kind of implementation allows a more user friendly approach of displaying content.

The drawing is custom so depending upon the requirements more changes on how/where to display the text/image etc can be made, along with changes to font/background color and other aspects which would give the UI the much needed 'wow effect'.

Source code

The container.h file looks like this:-

```
class CFishEyeListBox;

class CFishEyeContainer : public CCoeControl
 {
 private:
  CFishEyeListBox*  iListBox;
      };
```

The other functions have been omitted for clarity, the **CfishEyeListBox** is the custom listbox which will implement the fish eye functionality by doing a custom draw, more details on that later.

The container.cpp file looks like this:-

```
// System includes
#include <aknconsts.h>
#include <avkon.mbg>
#include <akniconarray.h>
#include <eikclbd.h>

// User includes
#include "FishEye.hrh"
#include "FishEyeListBox.h"
#include "FishEyeContainer.h"

void CFishEyeContainer::InitializeControlsL()
 {
 // Creates a new custom list box.
 iListBox = new (ELeave) CFishEyeListBox(); // The listbox which will display the fish
eye window when the item is selected
 iListBox->ConstructL(this, EAknListBoxSelectionList);
 //iListBox->SetListBoxObserver(this);

 // add scrollbars to listbox
 iListBox->CreateScrollBarFrameL();
 iListBox->ScrollBarFrame()->SetScrollBarVisibilityL(CEikScrollBarFrame::EOff,
CEikScrollBarFrame::EAuto);

 CArrayPtr< CGulIcon >* icons = NULL;
 icons = new (ELeave) CAknIconArray( 1 );
 CleanupStack::PushL( icons );
 // for EListBoxListboxList_iconIndex
```


(C) Copyright Nokia 2014. All rights reserved.

```
 icons->AppendL( LoadAndScaleIconL(KAvkonBitmapFile, EMbmAvkonQgn_prop_empty,
EMbmAvkonQgn_prop_empty_mask, NULL, EAspectRatioPreserved)); // Some default icons
 CleanupStack::Pop( icons );

 if (icons != NULL)
  {
  iListBox->ItemDrawer()->ColumnData()->SetIconArray( icons );
  }

 iListBox->Model()->SetOwnershipType( ELbmOwnsItemArray );

 // Feed some content to the listbox

 iListBox->View()->SetListEmptyTextL(KNullDesC);

 if(iListBox)
   {
   iListBox->MakeVisible(ETrue);
   iListBox->ScrollBarFrame()->SetScrollBarVisibilityL(CEikScrollBarFrame::EOff,
CEikScrollBarFrame::EAuto);
   }

 iListBox->SetFocus( ETrue );
 iFocusControl = iListBox;

 CTextListBoxModel* model = iListBox->Model();
 CDesCArray* itemArray = static_cast< CDesCArray* > ( model->ItemTextArray() );
// Some text for demoing the functionality
 itemArray->InsertL(0, _L("First ContactName\tCall After 9 PM"));
 itemArray->InsertL(1, _L("Second ContactName\tGym Instructor"));
 itemArray->InsertL(2, _L("Third ContactName\tStation Incharge"));
 itemArray->InsertL(3, _L("Fourth ContactName\tTempramental"));
 itemArray->InsertL(4, _L("Fifth ContactName\tPossible Lead"));
 itemArray->InsertL(5, _L("Sixth ContactName\tCaptain Designate"));
 itemArray->InsertL(6, _L("Seventh ContactName\tDVD Parlour"));
 iListBox->HandleItemAdditionL();

 }
```

The text displayed on the listbox is hard-coded for demo purposes, in real life cases they can be picked up from the server after a successful GET or any other dynamic methods implemented as the need be. The **iListBox** is derived from CaknSingleHeadingStyleListBox, and handles its own drawing by over-riding the default item drawer, details below.

`AknsDrawUtils` link against library `aknskins.lib` so add following line to your .mmp file.

```
LIBRARY aknskins.lib
```

The FishEyeListBox.h file looks like this:-

```
#include <eiklbi.h>   // CListItemDrawer
#include <eiktxlbx.h> // CEikTextListBox
#include <gdi.h>
#include <eikclb.h>
#include <aknlists.h>
```


(C) Copyright Nokia 2014. All rights reserved.

```
class CFishEyeListBox: public CAknSingleHeadingStyleListBox
 {
private:
 virtual void CreateItemDrawerL();
      };


class CCustomListItemDrawer: public CColumnListBoxItemDrawer
 {
public: // constructor and destructor
 CCustomListItemDrawer(CTextListBoxModel* aTextListBoxModel, const CFont* aFont,
CColumnListBoxData* aColumnData, const CEikTextListBox& aListBox);
 ~CCustomListItemDrawer();

private: // from CListItemDrawer
 virtual void DrawActualItem(TInt aItemIndex, const TRect& aActualItemRect,
         TBool aItemIsCurrent, TBool aViewIsEmphasized, TBool aViewIsDimmed,
   TBool aItemIsSelected) const;

 void DrawItem(TInt aItemIndex, TPoint aItemRectPos,
       TBool aSelected, TBool aItemIsCurrent, TBool aViewIsEmphasized,
   TBool aViewIsDimmed) const;

public: // new methods
 void SetFirstTime(){iFirstTime = ETrue;};
private: // private methods
 void DoDraw(TInt aIndex, TBool aItemIsCurrent, TDesC& aText, TRect& aRect) const;
private:
 CTextListBoxModel* iModel;
 const CEikTextListBox& iListBox;
 mutable TRect   iCurrentItemRect;
 mutable TBool  iFirstTime;
 };
```

The listbox derives from CaknSingleHeadingStyleListBox, whereby all the basic functionalities of the parent listbox are available to the child, and through the function CreateItemDrawerL, the child over-rides the default item drawer so that the drawing is done through the item drawer we set through our code and that handles the custom drawing of the listbox items depending on how we want to do it.

The CcustomListItemDrawer derives from the CcolumnListBoxItemDrawer, and is the item drawer that is reponsible for the custom drawing of the listbox. The DrawItem/ DrawActualItem functionalites are over-ridden from the base class,and hence get called by the framework when it attempts to draw the listbox.

The FishEyeListBox.cpp file looks like this:-

```
#include <aknsfld.h>
#include <aknviewappui.h>
#include <aknview.h>
#include "FishEye.hrh"
#include "FishEyeListBox.h"

void CCustomListItemDrawer::DrawActualItem(TInt aItemIndex, const TRect&
aActualItemRect,
                                      TBool aItemIsCurrent, TBool /*aViewIsEmphasized*/,
                                      TBool /*aViewIsDimmed*/, TBool aItemIsSelected)
const
```


(C) Copyright Nokia 2014. All rights reserved.

```
 {
 TInt selectedIndex = iListBox.CurrentItemIndex();
 TInt itemIndex = aItemIndex;
 TRect itemRect = aActualItemRect;

 if( selectedIndex == iListBox.BottomItemIndex() ) {
  // selected item is at the bottom
  itemRect.iTl.iY -= iItemCellSize.iHeight;
  if( itemIndex < selectedIndex ) {
   itemRect.iBr.iY -= iItemCellSize.iHeight;
  }
 } else {
  if (aItemIsCurrent) {
   itemRect.iBr.iY += iItemCellSize.iHeight;
  } else if (itemIndex > selectedIndex) {
   // Lets move the following entries down to accomodate fish eye window
   itemRect.iTl.iY += iItemCellSize.iHeight;
   itemRect.iBr.iY += iItemCellSize.iHeight;
  }
 }
 // Call the Actual draw function
 }

CFishEyeListBox::CFishEyeListBox()
:CAknSingleHeadingStyleListBox()
 {
 }

/**
 * Override the default item drawer by our own
 */
void CFishEyeListBox::CreateItemDrawerL()
 {
 CColumnListBoxData* columnData = CColumnListBoxData::NewL(); //
CColumnListBoxItemDrawer owns columnData
 const CFont* myFont = CEikonEnv::Static()->DenseFont();
 iItemDrawer = new(ELeave) CCustomListItemDrawer(Model(), myFont, columnData, *this);
 }

/**
 * To handle the item scrolling
 */
TKeyResponse CFishEyeListBox::OfferKeyEventL(const TKeyEvent& aKeyEvent,
  TEventCode aType) {
 if (EEventKey == aType) {
  switch (aKeyEvent.iCode) {
  // Down arrow
  case EKeyDownArrow: {
   ScrollDown();
   DrawDeferred();
   return EKeyWasConsumed;
  }

   // Up arrow
  case EKeyUpArrow: {
   ScrollUp();
   DrawDeferred();
```


(C) Copyright Nokia 2014. All rights reserved.

```
    return EKeyWasConsumed;
   }

   default:
    return CEikTextListBox::OfferKeyEventL(aKeyEvent, aType);
   }
  }
  return EKeyWasNotConsumed;
}

void CFishEyeListBox::ScrollDown(void) {
 TInt currIndex = CurrentItemIndex();
 if( currIndex < Model()->NumberOfItems() - 1 ) {
  SetCurrentItemIndex( currIndex + 1);
 } else {
  // Lets do circular scrolling by setting current index as the first item
  SetCurrentItemIndex(0);
 }
 ScrollToMakeItemVisible( CurrentItemIndex() );
}
void CFishEyeListBox::ScrollUp(void) {
 TInt currIndex = CurrentItemIndex();
 if( currIndex > 0 ) {
  SetCurrentItemIndex( currIndex - 1);
 } else if( Model()->NumberOfItems() > 0) {
  // Lets do circular scrolling by setting current index as the last item
  SetCurrentItemIndex(Model()->NumberOfItems() - 1);
 }
 ScrollToMakeItemVisible( CurrentItemIndex() );
}

/**
 * Override the draw function of the list
 * Otherwise it will apply a default background instead of transparent
 * On s60 3rd mr, there will be bugs when the list is large than default. ( A ugly white
bar at the end of list)
 */
void CFishEyeListBox::Draw(const TRect& aRect) const {
 CListBoxView* view = View();
 TInt start = view->TopItemIndex();
 TInt end = view->BottomItemIndex();
 for( TInt i = start; i <= end; i++ ) {
  view->DrawItem(i);
 }
}
```

The **DrawActualItem** is called by the framework when it tries to draw the listbox items. If the item is highlighted, we increase the bottom rect y co-ordinates so that the window has more space to display the contents and for all the other items below the selected item we keep pushing the entire item rect by the amount we have increased the rect for the selected item, so that they get drawn after leaving enough space for the fish eye window to come up.

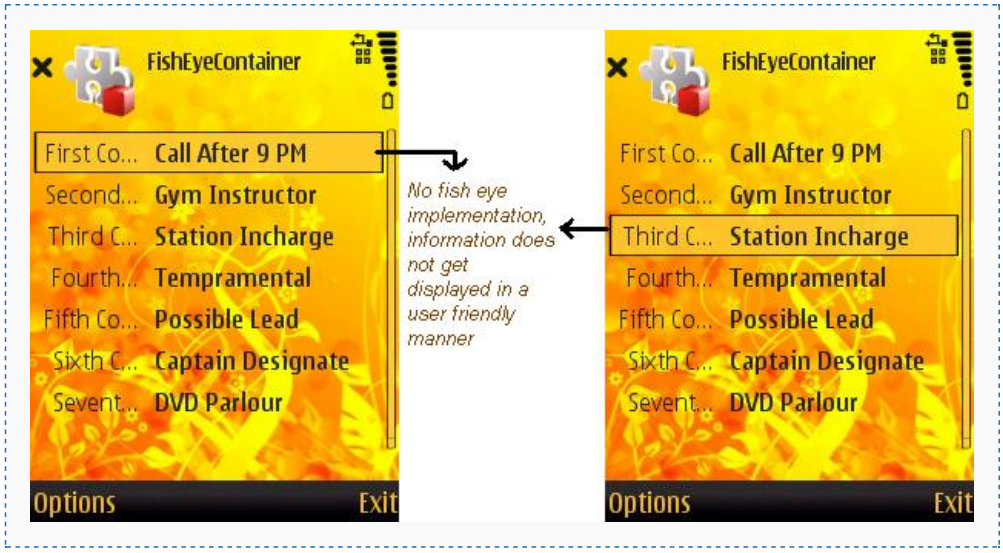
In the **CreateItemDrawerL**, we set the item drawer to our custom item drawer so that it can handle the drawing of the listbox when called by the framework.

All the functions and implementation details have been omitted for clarity

Before implementation of fish eye

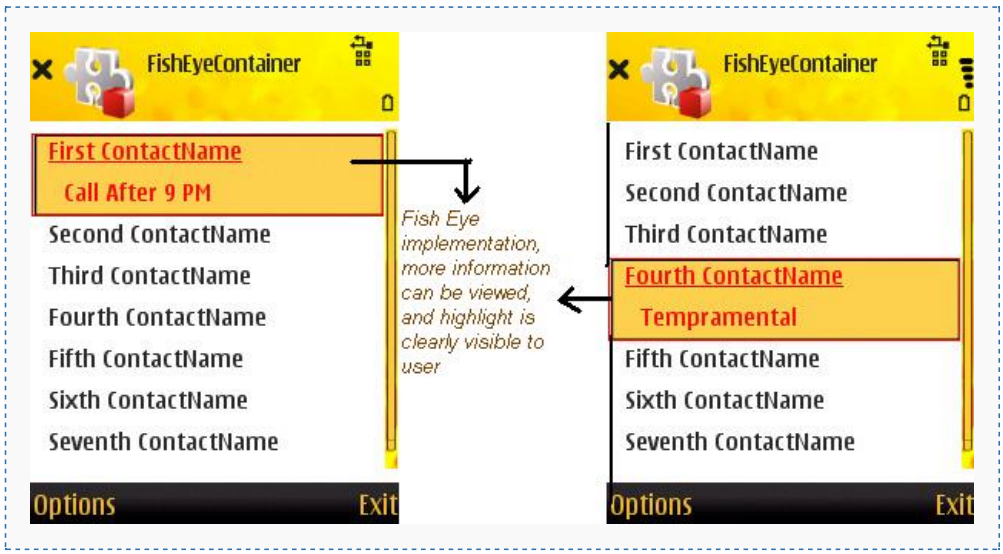
(C) Copyright Nokia 2014. All rights reserved.

The image below shows how the listbox looks without the fish eye implementation, where the content is not displayed in a user friendly way so to speak.



After implementation of fish eye

The image below shows how the listbox looks after the fish eye implementation, where the content is displayed in a user friendly way, in multi lines and the details are more clearly visible and also the highlight with different colored background for the selected text gives the window a distinct visibility as compared to the earlier image.



Download Link

Download a working S60 3rd Edition, FP1 example from :-

File:FishEye.zip

Link

Fisheye List

Anthony Pranata's Example on NewLC

Added by - Mayank on 24/06/2009

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.