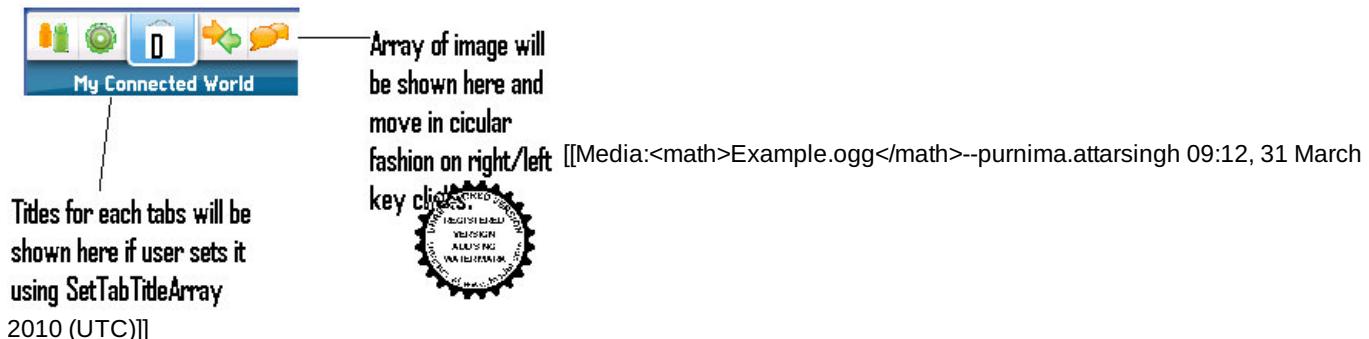


Custom tabs

This code example shows how to create customized tabs that improve over the native S60 tabs on S60 2nd Edition. It will be familiar to tabs provided in many other closed platform phone (LG, Motorola, Samsung etc). Though its not exactly same as native tabs but it has most of the functionality. Also source is open to customize as per the needs.

Screen shot



Code

[File:CustomTabControl.zip](#)

Functional Description

- Tab group can bring to focus or de-focus using `setFocus()`.
- Once the tab group is in focus, left and right keys will move tabs in desired direction.
- User has to create an array of `CGuiIcon*` and set it to this control using `SetIconArray()`. These icons will show up as icon on tab group as tabs.
- Also, optionally user can set an array of descriptors (`CDesCArrayFlat*`) for titles respective to each tab.
- For first time show, user has to call `setDefaultTabByIndex()` to set which tab will show as the default tab (first time when u show up the tab group).
 - **There is no need to have `setActiveTabByIndex()` equivalent in this control as the control itself keeps track of left-right keys and set tabs accordingly.**
- Optionally, user can set background images, one to be shown when tabgroup is in focus and one to be shown when tabgroup is defocused. See `SetFocusedBackgroundImage()` and `SetDefocusedBackgroundImage()`

Header File

```
#ifndef __CUSTOMTABCONTROL_H__
#define __CUSTOMTABCONTROL_H__
#include <coecntrl.h>
#include <AknsDrawUtils.h> // skin
#include <AknsBasicBackgroundControlContext.h> //skin
#include <aknscontrolcontext.h> //MAknsControlContext

// FORWARD DECLARATIONS

class CGuiIcon;
class CFbsBitmap;

// custom tab control
typedef enum
{
    SHIFT_LEFT=0,
    SHIFT_RIGHT
}
```

```
    } SHIFT_DIR;

class CCustomTabControl: public CCoeControl, MCoeControlObserver
{
public:

    static CCustomTabControl* NewL(const TRect& aRect,const CCoeControl* aParent);

    ~CCustomTabControl();

TKeyResponse OfferKeyEventL(const TKeyEvent& aKeyEvent,TEventCode aType);

void HandleControlEventL(CCoeControl* aControl,TcoeEvent aEventType);

CCustomTabControl(const TRect& aRect):iRect(aRect)
{
}

TTypeUid::Ptr MopSupplyObject(TTypeUid aId);

void Draw(const TRect& aRect) const;

// takes the ownership
void SetFocusedBackgroundImage(CFbsBitmap* aBackgroundImage)
{
    iFocusedBackgroundImage=aBackgroundImage;
}

        // takes the ownership
void SetDeFocusedBackgroundImage(CFbsBitmap* aBackgroundImage)
{
    iDeFocusedBackgroundImage=aBackgroundImage;
}

TInt ActiveTabIndex()
{
    return iActiveTabIndex;
}

void SetDefaultTabByIndex(TInt aIndex);

        // takes the ownership
void SetIconArray(CArrayPtr<CGuiIcon>* aIconArray)
{
    if(iIconArray != NULL)
    {
        iIconArray->ResetAndDestroy();
    }
    iIconArray=aIconArray;
}

        // takes the ownership
```

```
void SetTabTitleArray(CDesCArrayFlat* aTabTitleArray)
{
    if(iTabTitleArray != NULL)
    {
        delete iTabTitleArray;
    }
    iTabTitleArray=aTabTitleArray;
}

void SetNumberOfTabsToBeShown(TInt aCount)
{
//if(aCount>0)
// iNumberOfTabsToBeShown=aCount;
}

void SetFocus(TBool aFocus)
{
    iFocused=aFocus;
    DrawDeferred();
}

TBool IsFocused()
{
    return iFocused;
}

void MakeVisible(TBool aVisible)
{
    iVisible=aVisible;
    CCoeControl::MakeVisible(aVisible);
}

TBool IsVisible()
{
    return iVisible;
}

TInt TabCount()
{
    if(iIconArray)
        return iIconArray->Count();
    else
        return 0;
}

TInt TabIdFromIndex(TInt aIndex)
{
    return aIndex;
}
```

```
void ShiftTabsBack();

private:
    void ConstructL(const TRect& aRect,const CCoeControl* aParent);

    void ShiftTabs(SHIFT_DIR aDir);

private:
    MAknsControlContext* iBackGround;

TRect     iRect;

CFbsBitmap*      iFocusedBackgroundImage;

CFbsBitmap*      iDeFocusedBackgroundImage;

    // Currently active tab
TInt     iActiveTabIndex;

    // Position of the active tab; currently assumed to be 2
TInt     iActiveTabPosition;

CArryPtr<CGuiIcon>*  iIconArray;

CDesCArrayFlat*   iTabTitleArray;

TInt     iNumberOfTabsToBeShown;

TBool     iFocused;

TBool     iVisible;

};

#endif __CUSTOMTABCONTROL_H__
```

Source File

```
#include <gulicon.h>
#include <eikenv.h>
#include <AknAppUi.h>
#include <aknsbasicbackgroundcontrolcontext.h> //CAknsBasicBackgroundControlContext
#include "CustomTabControl.h"

CCustomTabControl* CCustomTabControl::NewL(const TRect& aRect,const CCoeControl*
aParent)
{
    CCustomTabControl* self = new(ELeave) CCustomTabControl(aRect);
    CleanupStack::PushL(self);
    self->ConstructL(aRect,aParent);
    CleanupStack::Pop(); // self
    return self;
```

```
}

CCustomTabControl::~CCustomTabControl()
{
    if(iBackGround)
    {
        delete iBackGround;
        iBackGround = NULL;
    }
    if(iFocusedBackgroundImage)
    {
        delete iFocusedBackgroundImage;
        iFocusedBackgroundImage=NULL;
    }
    if(iDeFocusedBackgroundImage)
    {
        delete iDeFocusedBackgroundImage;
        iDeFocusedBackgroundImage=NULL;
    }
    if(iIconArray)
    {
        iIconArray->ResetAndDestroy();
    }
    if(iTabTitleArray)
    {
        delete iTabTitleArray;
    }
}

void CCustomTabControl::ConstructL(const TRect& aRect,const CCoeControl* aParent)
{
    iFocusedBackgroundImage=NULL;
    iDeFocusedBackgroundImage=NULL;
    iIconArray=NULL;
    iTabTitleArray=NULL;
    iNumberOfTabsToBeShown=5;
    iActiveTabPosition=2;
    iActiveTabIndex=2;
    if (aParent == NULL)
    {
        CreateWindowL();
        iAvkonAppUi->AddToStackL( this);
    }
    else
    {
        // Part of a compound control, so just share
        // the parent's window
        SetContainerWindowL(*aParent);
    }
    SetRect(aRect);
    iBackGround = CAknsBasicBackgroundControlContext::NewL( KAknsIIDQsnBgAreaMain,
iRect, EFalse );
    SetRect(aRect);
    ActivateL();
}
```

```
TTypeUid::Ptr CCustomTabControl::MopSupplyObject(TTypeUid aId)
{
    if(aId.iUid == MAKnsControlContext::ETypeId && iBackGround)
    {
        return MAKnsControlContext::SupplyMopObject( aId, iBackGround);
    }
    return CCoeControl::MopSupplyObject( aId );
}

void CCustomTabControl::HandleControlEventL(CCoeControl* /*aControl*/, TcoeEvent
/*aEventType*/)
{
}

void CCustomTabControl::ShiftTabsBack()
{
    if(iActiveTabIndex>0)
    {
        iActiveTabIndex--;
    }

    else
    {
        iActiveTabIndex=iIconArray->Count()-1;
    }

    ShiftTabs(SHIFT_LEFT);
    DrawNow();
}

TKeyResponse CCustomTabControl::OfferKeyEventL(const TKeyEvent& aKeyEvent, TEventCode
aType)
{
    KN_UI_LOG((KN_UI_LOG_TYPE_DEBUG, "CCustomTabControl::OfferKeyEventL\n"));

    TKeyResponse ret=EKeyWasNotConsumed;
    if(IsFocused() && EEventKeyUp == aType)
    {
        switch (aKeyEvent.iScanCode)
        {
            case EStdKeyLeftArrow:
            {
                if(iActiveTabIndex>0)
                    iActiveTabIndex--;
                else
                    iActiveTabIndex=iIconArray->Count()-1;
                ShiftTabs(SHIFT_LEFT);
                DrawNow();
                ret=EKeyWasConsumed;
            }
            break;
            case EStdKeyRightArrow:
            {
                if(iActiveTabIndex<iIconArray->Count()-1)
                    iActiveTabIndex++;
                else

```

```
iActiveTabIndex=0;
ShiftTabs(SHIFT_RIGHT);
DrawNow();
ret=EKeyWasConsumed;
}
break;
default:
break;
}
}
return ret;
}

void CCustomTabControl::ShiftTabs(SHIFT_DIR aDir)
{
if(iIconArray && iIconArray->Count()>0)
{
CGuiIcon* icon=NULL;
if(aDir==SHIFT_RIGHT)
{
icon=iIconArray->At(0);
iIconArray->Delete(0);
iIconArray->Compress();
if(icon)
iIconArray->AppendL(icon);
}
else
{
icon=iIconArray->At(iIconArray->Count()-1);
iIconArray->Delete(iIconArray->Count()-1);
iIconArray->Compress();
if(icon)
iIconArray->InsertL(0,icon);
}
}
}

void CCustomTabControl::SetDefaultTabByIndex(TInt aIndex)
{
if(aIndex>=0 && aIndex<iActiveTabPosition)
{
for(TInt i=0;i<iActiveTabPosition-aIndex;i++)
{
ShiftTabs(SHIFT_LEFT);
}
}
if(aIndex>iActiveTabPosition && aIndex<iIconArray->Count())
{
for(TInt i=0;i<aIndex-iActiveTabPosition;i++)
{
ShiftTabs(SHIFT_RIGHT);
}
}
if(iIconArray && aIndex>=0 && aIndex<iIconArray->Count())
{
iActiveTabIndex=aIndex;
DrawNow();
}
```

```
}

void CCustomTabControl::Draw(const TRect& aRect) const
{
    CWindowGc& gc = SystemGc();
    //TRect aRect(aaRect);
    //aRect.iTl.iX=0;
    //aRect.iTl.iY=0;
    gc.Clear(aRect);
    gc.SetPenStyle(CGraphicsContext::ESolidPen);
    TSize pensize(2,2);
    gc.SetPenSize(pensize);
    gc.SetPenColor(KRgbBlack);
    gc.SetBrushStyle(CGraphicsContext::ENullBrush);
    gc.DrawRect(aRect);
    MAknsSkinInstance* skin = AknsUtils::SkinInstance();
    MAknsControlContext* cc = AknsDrawUtils::ControlContext( this );
    AknsDrawUtils::Background( skin, cc, this, gc, aRect );
    if(iFocused)
    {
        if(iFocusedBackgroundImage)
        {
            gc.DrawBitmap(aRect, iFocusedBackgroundImage);
        }
    }
    else
    {
        if(iDeFocusedBackgroundImage)
        {
            gc.DrawBitmap(aRect, iDeFocusedBackgroundImage);
        }
    }
}

if(iIconArray && iIconArray->Count()>0 && iNumberOfTabsToBeShown>0
    && iNumberOfTabsToBeShown<=iIconArray->Count())
{
    TInt tabWidth=aRect.Width()/iNumberOfTabsToBeShown;
    TSize tabSize(tabWidth,aRect.Height());
    TRect tabRect(tabSize);
    tabRect.Shrink(4,4);
    tabRect.iTl.iY-=3;
    tabRect.iBr.iY-=3;
    TInt dX=tabRect.Width()+3;
#ifdef __UI_FRAMEWORKS_V2
    dX=tabRect.Width()+5;
#endif
    for(TInt i=0;i<iNumberOfTabsToBeShown;i++)
    {
        CFbsBitmap* tab=NULL;
        CGuiIcon* icon=NULL;
        if(i<iIconArray->Count())
            icon = iIconArray->At(i);
        if(icon)
            tab = icon->Bitmap();
        // to adjust the middle tab (middle tab is bigger than other tabs)
        if(i==2)
```

```
{  
#ifdef __UI_FRAMEWORKS_V2  
    tabRect.iTl.iX+=5;  
    tabRect.iBr.iX+=5;  
#else  
    tabRect.iTl.iX+=9;  
    tabRect.iBr.iX+=9;  
#endif  
}  
if(tab)  
{  
    TRect rect;  
    rect.iTl.iX=0;  
    rect.iTl.iY=0;  
    rect.iBr.iX=tabRect.Width();  
    rect.iBr.iY=tabRect.Height();  
    gc.BitBltMasked(tabRect.iTl,tab,rect,icon->Mask(),EFalse);  
}  
tabRect.iTl.iX+=dX;  
tabRect.iBr.iX+=dX;  
// to adjust the middle tab (middle tab is bigger than other tabs)  
if(i==2)  
{  
    tabRect.iTl.iX+=10;  
    tabRect.iBr.iX+=10;  
}  
  
}  
}  
  
if(iTabTitleArray && iTabTitleArray->Count()>0 && iActiveTabIndex>=0  
&& iActiveTabIndex<iTabTitleArray->Count())  
{  
    gc.SetPenStyle(CGraphicsContext::ESolidPen);  
    gc.SetPenColor(KRgbWhite);  
    const CFont* font = CEikonEnv::Static()->LegendFont();  
    gc.UseFont(font);  
    TRect textRect(aRect);  
    TInt baseline = textRect.iBr.iY - 5;  
    TBuf<200> titleText;  
    titleText.FillZ();  
    titleText.Copy(iTabTitleArray->MdcaPoint(iActiveTabIndex));  
    gc.DrawText(titleText,textRect, baseline, CGraphicsContext::ECenter, 1);  
}  
  
}  
}
```

How to use this control?

Here is the sample code on how to use it. It is very similar on how you use the standard tabgroup.

```
void CMyAppUi::CreateAndShowTabsL()  
{  
    // custom tab construction  
    TRect tabRect(ClientRect());
```

```
//tabRect.iTl.iY+=40;
tabRect.iBr.iY=tabRect.iTl.iY+55;
iTabGroup=CCustomTabControl::NewL(tabRect,NULL);
//iTabGroup->SetContainerWindowL(*this);
iTabGroup->MakeVisible(ETrue);
iTabGroup->SetFocus(ETrue);

CFbsBitmap* bitmap = iEikonEnv->CreateBitmapL( KTabBackgroundFileName,
EMbmTabgroupGlobalnav_active );
iTabGroup->SetFocusedBackgroundImage(bitmap);

CFbsBitmap* bitmap1 = iEikonEnv->CreateBitmapL( KTabBackgroundFileName,
EMbmTabgroupGlobalnav_inactive );
iTabGroup->SetDeFocusedBackgroundImage(bitmap1);

CArryPtr<CGuiIcon>* tabiconArray = new( ELeave ) CAknIconArray(40);
CleanupStack::PushL( tabiconArray );
tabiconArray->AppendL( iEikonEnv->CreateIconL(
KTabsFileName,EMbmTabsGlobalnav_contacts) );
tabiconArray->AppendL( iEikonEnv->CreateIconL(
KTabsFileName,EMbmTabsGlobalnav_settings) );
tabiconArray->AppendL( iEikonEnv->CreateIconL( KTabsFileName,EMbmTabsGlobalnav_home) );
tabiconArray->AppendL( iEikonEnv->CreateIconL(
KTabsFileName,EMbmTabsGlobalnav_communicate));
tabiconArray->AppendL( iEikonEnv->CreateIconL(
KTabsFileName,EMbmTabsGlobalnav_convos));

CleanupStack::Pop();

// Transfers the ownership
iTabGroup->SetIconArray(tabiconArray);

CDesCArrayFlat* array = new (ELeave) CDesCArrayFlat(10);
CleanupStack::PushL(array);
array->AppendL(KItemTextTab1);
array->AppendL(KItemTextTab2);
array->AppendL(KItemTextTab3);
array->AppendL(KItemTextTab4);
array->AppendL(KItemTextTab5);
CleanupStack::Pop();
// Transfers the ownership
iTabGroup->SetTabTitleArray(array);
iTabGroup->SetDefaultTabByIndex(2);
//AddToStackL(iTabGroup);

iTabGroup->DrawNow();
}
```

Then you can do the following to move between your multiple views...

```
TKeyResponse CMyAppUi::HandleKeyEventL(const TKeyEvent& aKeyEvent, TEventCode /*aType*/)
{
```

```
    if ( iTabGroup == NULL )
http://developer.nokia.com/community/wiki/Custom_tabs
```

(C) Copyright Nokia 2014. All rights reserved.

```
{  
    return EKeyWasNotConsumed;  
}  
  
if(iTabGroup->IsFocused())  
{  
    TInt active = iTabGroup->ActiveTabIndex();  
    TInt count = iTabGroup->TabCount();  
  
    switch ( aKeyEvent.iCode )  
    {  
        case EKeyLeftArrow:  
            if ( active > 0 )  
            {  
                active--;  
                SwitchView(TUid::Uid(iTabGroup->TabIdFromIndex(active)+1));  
            }  
            else  
            {  
                active=count-1;  
                SwitchView(TUid::Uid(iTabGroup->TabIdFromIndex(active)+1));  
            }  
            return EKeyWasConsumed;  
        case EKeyRightArrow:  
            if( (active + 1) < count )  
            {  
                active++;  
                SwitchView(TUid::Uid(iTabGroup->TabIdFromIndex(active)+1));  
            }  
            else  
            {  
                active=0;  
                SwitchView(TUid::Uid(iTabGroup->TabIdFromIndex(active)+1));  
            }  
            return EKeyWasConsumed;  
        default:  
            return EKeyWasNotConsumed;  
    }  
}  
  
return EKeyWasNotConsumed;  
}
```

