Custom button

This article is about creating a custom button control with a picture and a label. It has a basic functionality of clicking. When the focus is on button control and OK is clicked, this control notifies to its observer about click event.

This is mainly for those who migrated from UIQ to Symbian and missing native button control.

Hearders Required:

```
#include <gulicon.h> //CGulIcon
#include <fbs.h>  //CFbsBitmap
#include <COECNTRL.H> //CCoeControl
#include <coecobs.h>  //MCoeControlObserver
#include <aknscontrolcontext.h> //MAknsControlContext
```

Library required:

```
LIBRARY egul.lib //CGulIcon
LIBRARY fbscli.lib //CFbsBitmap
```

Here is the header file

```
class CCustomButtonControl;

class MCoeButtonControlObserver
{
public:
    virtual void HandleButtonClickedEventL(
                        const CCustomButtonControl* aControl)=0;
};

class CCustomButtonControl: public CCoeControl, MCoeButtonControlObserver
{
public:

    static CCustomButtonControl* NewL(const TRect& aRect,
                        MCoeButtonControlObserver*
                        aButtonObserver,
                        const CCoeControl* aParent);
    ~CCustomButtonControl();
    TKeyResponse OfferKeyEventL(const TKeyEvent& aKeyEvent,
                                TEventCode aType);
    void HandleControlEventL(CCoeControl* aControl,TCoeEvent aEventType);

    CCustomButtonControl(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;
    }

    void SetIcon(CGulIcon*  aIcon) // takes the ownership
    {
       if(iIcon)
       {
          delete iIcon;
       }
       iIcon=aIcon;
    }

    void SetLabel(const TDesC& aDes)
    {
       iLabel.Delete(0,iLabel.Length());
       iLabel.FillZ();
       iLabel.Copy(aDes.Left(50));
    }

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

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

    TBool IsFocused()
    {
       return iFocused;
    }
private:
    void ConstructL(const TRect& aRect,
                    MCoeButtonControlObserver* aButtonObserver,
                    const CCoeControl* aParent);
private:
```


(C) Copyright Nokia 2013. All rights reserved.

```
    MAknsControlContext*  iBackGround;
    MCoeButtonControlObserver*  iButtonObserver;
    CFbsBitmap*                 iFocusedBackgroundImage;
    CFbsBitmap*                 iDeFocusedBackgroundImage;
    TRect iRect;
    CGulIcon*                   iIcon;
    TBuf<50>                    iLabel;
    TBool iFocused;
};
```

Source

```
#include "aknappui.h"
#include "aknsbasicbackgroundcontrolcontext.h"

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

CCustomButtonControl::~CCustomButtonControl()
{
    if(iBackGround)
    {
        delete iBackGround;
        iBackGround = NULL;
    }
    if(iFocusedBackgroundImage)
    {
     delete iFocusedBackgroundImage;
     iFocusedBackgroundImage=NULL;
    }
    if(iDeFocusedBackgroundImage)
    {
     delete iDeFocusedBackgroundImage;
     iDeFocusedBackgroundImage=NULL;
    }

    if(iIcon)
    {
     delete iIcon;
    }
}

void CCustomButtonControl::ConstructL(const TRect& aRect,
                      MCoeButtonControlObserver* aButtonObserver,
                      const CCoeControl* aParent)
{
    iFocusedBackgroundImage=NULL;
    iDeFocusedBackgroundImage=NULL;
    iIcon=NULL;
    iLabel.FillZ();
    iFocused=EFalse;
    if (aParent == NULL)
    {
        CreateWindowL();
        iAvkonAppUi->AddToStackL( this);
    }
    else
    {
        // Part of a compound control, so just share
        // the parent's window
        SetContainerWindowL(*aParent);
    }
    if(aButtonObserver)
        iButtonObserver=aButtonObserver;
    else
        iButtonObserver=NULL;
    SetRect(aRect);
    iBackGround = CAknsBasicBackgroundControlContext::NewL(
                        KAknsIIDQsnBgAreaMain, iRect, EFalse );
    SetRect(aRect);
    ActivateL();

}

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

void CCustomButtonControl::HandleControlEventL(
                           CCoeControl* /*aControl*/,TCoeEvent /*aEventType*/)
{
}

TKeyResponse CCustomButtonControl::OfferKeyEventL(const TKeyEvent& aKeyEvent,
                                   TEventCode aType)
{
    TKeyResponse ret=EKeyWasNotConsumed;
    if(iButtonObserver && IsFocused() && EEventKey == aType)
    {
        switch (aKeyEvent.iScanCode)
        {
            case EStdKeyDevice3:
            {
                iButtonObserver->HandleButtonClickedEventL(this);
         ret=EKeyWasConsumed;
            }   break;

            default:
                break;
        }
    }
    return ret;
```


(C) Copyright Nokia 2013. All rights reserved.

```
}


void CCustomButtonControl::Draw(const TRect& aRect) const
{
    CWindowGc& gc = SystemGc();
    gc.Clear(aRect);
    gc.SetPenStyle(CGraphicsContext::ESolidPen);
    TSize pensize(1,1);
    gc.SetPenSize(pensize);
    gc.SetBrushColor(KRgbGray);
    if(iFocused)
        gc.SetPenColor(KRgbBlack);
    else
        gc.SetPenColor(KRgbGray);
    TSize round(2,2);
    gc.SetBrushStyle(CGraphicsContext::ESolidBrush);
    gc.DrawRoundRect(aRect,round);

    gc.SetBrushStyle(CGraphicsContext::ENullBrush);

    if(iDeFocusedBackgroundImage)
    {
        if(!iFocused)
        {
            TRect backImage(aRect);
            backImage.Shrink(1,1);
            gc.DrawBitmap(backImage, iDeFocusedBackgroundImage);
        }
        else if(iFocusedBackgroundImage)
        {
            TRect backImage(aRect);
            backImage.Shrink(1,1);
            gc.DrawBitmap(backImage, iFocusedBackgroundImage);
        }
    }
    else if(iFocusedBackgroundImage)
    {
        TRect backImage(aRect);
        backImage.Shrink(1,1);
        gc.DrawBitmap(backImage, iFocusedBackgroundImage);
    }

    CFbsBitmap* image=NULL;
    TRect imageRect;
    if(iIcon)
    {
        image = iIcon->Bitmap();
        if(image)
        {
            if(iLabel.Length()>0)
            {
                imageRect.SetRect(
                   aRect.iTl.iX+5,
                  (aRect.Height() - image->Header().iSizeInPixels.iHeight)/2,
                   aRect.iTl.iX+5+aRect.Height() -
                                   image->Header().iSizeInPixels.iWidth,
                  ((aRect.Height()-image->Header().iSizeInPixels.iHeight)/2) +
                   aRect.Height()-image->Header().iSizeInPixels.iHeight);
            }
            else
            {
                TInt Tx=aRect.iTl.iX;
                TInt Ty=aRect.iTl.iY;
                TInt Bx=aRect.Width();
                TInt By=aRect.Height();
                if(image->Header().iSizeInPixels.iWidth<aRect.Width())
                {
                    Tx+=(aRect.Width()/2) -
                        (image->Header().iSizeInPixels.iWidth/2);
                    Bx=Tx+image->Header().iSizeInPixels.iWidth;
                }

                if(image->Header().iSizeInPixels.iHeight<aRect.Height())
                {

      Ty+=(aRect.Height()/2) -
                        (image->Header().iSizeInPixels.iHeight/2);
                    By=Ty+image->Header().iSizeInPixels.iHeight;
                }
                imageRect.SetRect(Tx,Ty,Bx,By);
            }
            TRect rect;
            rect.iTl.iX=0;
            rect.iTl.iY=0;
            rect.iBr.iX=imageRect.Width();
            rect.iBr.iY=imageRect.Height();
            gc.BitBltMasked(imageRect.iTl,image,
                            rect,iIcon->Mask(),EFalse);
            //gc.DrawBitmap(imageRect,image);
        }
    }

    if(iLabel.Length()>0)
    {
        gc.SetPenStyle(CGraphicsContext::ESolidPen);

        const CFont* font = CEikonEnv::Static()->LegendFont();
        gc.UseFont(font);
        TRect textRect(aRect);
        TInt baseline;
        if(image)
        {
            textRect.iTl.iX+=image->Header().iSizeInPixels.iWidth+2;
            baseline=imageRect.iBr.iY-4;
        }
        else
        {
            baseline = textRect.Height()/2;
        }
        gc.DrawText(iLabel,textRect, baseline, CGraphicsContext::ECenter, 1);
    }
}
```


(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.