Archived:Scaling bitmaps using Symbian C++

Archived: This article is archived because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user=~~~~|*write your reason here*}}.

Overview

This code snippet shows how to scale a bitmap using CBitmapScaler.

CBitmapScaler is the public API that clients can call for BitmapTransforms Library bitmap scaling.

Before scaling the bitmap you must decode CFbsBitmap from an image. Bitmaps are decoded using CImageDecoder.

MMP file

The following capabilities and libraries are required:

```
CAPABILITY None
LIBRARY bitmaptransforms.lib
LIBRARY fbscli.lib
```

Header

```
#include <bitmaptransforms.h>
#include <e32base.h>
#include <fbs.h>

class MBitmapScalerObserver
    {
    public:
        virtual void Completed(TInt aErr) = 0;
    };

class CMyBitmapScaler : public CActive
    {
    public:
        static CMyBitmapScaler* NewL(MBitmapScalerObserver* aObserver);
        virtual ~CMyBitmapScaler();

    public:
        void Scale( TInt aPercent );
        CFbsBitmap* GetBitmap();

    protected: // From CActive
        void DoCancel();
        void RunL();
        TInt RunError(TInt aError);

    private:
        CMyBitmapScaler(MBitmapScalerObserver* aObserver);
        void ConstructL();

    private:
        CFbsBitmap*             iBitmap;
        CBitmapScaler*          iScaler;
        MBitmapScalerObserver*   iObserver;
    };
```

Source

```
CMyBitmapScaler* CMyBitmapScaler::NewL(
    MBitmapScalerObserver* aObserver )
    {
    CMyBitmapScaler* self =
        new(ELeave) CMyBitmapScaler( aObserver );
    CleanupStack::PushL( self );
    self->ConstructL();
    CleanupStack::Pop(self);
    return self;
    }

CMyBitmapScaler::CMyBitmapScaler(
    MBitmapScalerObserver* aObserver ) :
    CActive( EPriorityStandard ), iObserver( aObserver )
    {
    }

void CMyBitmapScaler::ConstructL()
    {
    iScaler = CBitmapScaler::NewL();
    CActiveScheduler::Add(this);
    }

CMyBitmapScaler::~CMyBitmapScaler()
    {
    Cancel();
```


(C) Copyright Nokia 2013. All rights reserved.

```
    delete iBitmap;
    delete iScaler;
    }

void CMyBitmapScaler::DoCancel()
    {
    iScaler->Cancel();
    }

void CMyBitmapScaler::RunL()
    {
    iObserver->Completed(iStatus.Int());
    }

TInt CMyBitmapScaler::RunError(TInt aError)
    {
    iObserver->Completed(aError);
    return KErrNone;
    }

void CMyBitmapScaler::Scale(TInt aPercent)
    {
    if (IsActive())
        {
        return;
        }

    TReal32 multiplier;
    multiplier = aPercent/(TReal32)100.0;

    TInt newWidth = (TInt)(iBitmap->SizeInPixels().iWidth * multiplier);
    TInt newHeight = (TInt)(iBitmap->SizeInPixels().iHeight * multiplier);

    // The maintain aspect ratio is by default set to true
    iScaler->Scale( &iStatus, *iBitmap, TSize(newWidth,newHeight) );

    SetActive();
    }

CFbsBitmap* CMyBitmapScaler::GetBitmap()
    {
    return iBitmap;
    }
```

Using

```
TInt KScaleUpFactor = 150;    // Scale up by 50%
TInt KScaleDownFactor = 50;   // Scale down by 50%

iScaler = CMyBitmapScaler::NewL(this);
iScalser->Scale(KScaleUpFactor);
```

Postconditions

The bitmap is scaled.

See also

Archived:Displaying image information using Symbian C++

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.