



CGif_Reader shows a simple GIF animation reader class that can be used to show a GIF animation in **CCoeControl** based containers. To construct the **CGif_Reader** add the following lines to the *ConstructL()* method inside your container:

```
iGif_Reader = new(ELeave)CGif_Reader(*this);
iGif_Reader->ConstructL(KTxtFileName);
```

KTxtFileName is the GIF file name (with drive, full path, name, and extension). To draw the GIF animation in your container you implement the container's *Draw()* method:

```
void CMycontainer::Draw(const TRect& aRect) const
{
    CWindowGc& gc = SystemGc();

    if(iGif_Reader)
    {
        if(iGif_Reader->Bitmap())
        {
            if(iGif_Reader->Bitmap()->Handle())
            {
                gc.DrawBitmap(aRect, iGif_Reader->Bitmap());
            }
        }
    }
}
```

Gif_Reader.cpp

```
#include "Gif_Reader.h"

#include <cntfield.h>
#include <cntdef.h>
#include <cntitem.h>
#include <cntfldst.h>
#include <cntdb.h>
#include <COEMAIN.H>
#include <akniconutils.h>

_LIT8(KtxTypeImageGif_8, "image/gif");

const TInt KFrameTimeOut = 200000; // 0.2 sec frame timeout

CGif_Reader::CGif_Reader(CCoeControl& aParent)
:
 CActive(0), iParent(aParent), iCurrImg(-1)
{
}

CGif_Reader::~CGif_Reader()
{
    Cancel();
    delete iExampleTimer;
    delete iImageDecoder;
    delete iFrame;
    delete iFrameImg;
}
```

Showing_GIF_animations

```
delete iFrameMsk;

delete iBitmapDevice;
delete iGraphicsContext;
}

void CGif_Reader::ConstructL(const TDesC& aFileName)
{
    CActiveScheduler::Add(this);
    iImageDecoder = CImageDecoder::FileNewL(CCoeEnv::Static()->FsSession(),
aFileName, KtxTypeImageGif_8);
    iCurrCount = iImageDecoder->FrameCount();

    if(iCurrCount > 0)
    {
        iFrame = new(ELeave)CFbsBitmap();
        iFrame->Create(iImageDecoder->FrameInfo(0).iOverallSizeInPixels,
iImageDecoder->FrameInfo(0).iFrameDisplayMode);

        iBitmapDevice = CFbsBitmapDevice::NewL(iFrame);
        User::LeaveIfError(iBitmapDevice->CreateContext(iGraphicsContext));
    }
    //should call NextImageL to actually start showing something
    NextImageL();
}

void CGif_Reader::TimerExpired(TAny* /*aTimer*/, TInt /*aError*/)
{
    Cancel();
    NextImageL();
}

void CGif_Reader::NextImageL(void)
{
    iCurrImg++;
    if(iCurrImg >= iCurrCount || iCurrImg < 0)
    {
        iCurrImg = 0;
    }

    delete iFrameImg;
    iFrameImg = NULL;
    iFrameImg = new(ELeave)CFbsBitmap();
    iFrameImg->Create(iImageDecoder->FrameInfo(iCurrImg).iOverallSizeInPixels,
iImageDecoder->FrameInfo(iCurrImg).iFrameDisplayMode);

    delete iFrameMsk;
    iFrameMsk = NULL;
    iFrameMsk = new(ELeave)CFbsBitmap();
    iFrameMsk->Create(iImageDecoder->FrameInfo(iCurrImg).iOverallSizeInPixels,
EGray2);

    iImageDecoder->Convert(&iStatus, *iFrameImg, *iFrameMsk, iCurrImg);
    SetActive();
}

void CGif_Reader::DoCancel()
{
    iImageDecoder->Cancel();
}
```

Showing_GIF_animations

```
}

CFbsBitmap* CGif_Reader::Bitmap()
{
    return iFrame;
}

void CGif_Reader::RunL()
{
    if(iFrameMsk && iFrameImg && iFrame && iGraphicsContext)
    {
        if(iFrameMsk->Handle() && iFrameImg->Handle() && iFrame->Handle())
        {
            if(TFrameInfo::ERestoreToBackground & iImageDecoder->FrameInfo(iCurrImg).
            {
                iGraphicsContext->DrawBitmap(
TRect(0, 0, iFrame->SizeInPixels().iWidth, iFrame->SizeInPixels().iHeight),
iFrameImg);
            }
            else
            {
                iGraphicsContext->DrawBitmapMasked(iImageDecoder->FrameInfo(iCurr
iFrameImg, TRect(0, 0, iFrameImg->SizeInPixels().iWidth, iFrameImg->SizeInPixels().iHeight),
iFrameMsk, EFalse);
            }
        }
    }

    iParent.DrawNow();

    if(iStatus.Int() != KErrCancel)
    {
        if(!iExampleTimer)
        {
            iExampleTimer = CExampleTimer::NewL(CActive::EPriorityStandard, *this);
        }
        // will be redrawing next frame after interval specified in Gif
        TTimeIntervalMicroSeconds nextDelay = iImageDecoder->FrameInfo(iCurrImg).iDelay;
        iExampleTimer->After(nextDelay.Int64()); //KFrameTimeOut);
    }
}
}
```

Gif_Reader.h

```
#include <e32base.h>
#include <coecntrl.h>
#include <w32std.h>
#include <e32std.h>
#include <cntdef.h>
#include <cntdb.h>
#include <ImageConversion.h>

#include "CExampleTimer.h"

class CGif_Reader : public CActive, MExampleTimerNotify
{
public:
```

Showing_GIF_animations

```
CGif_Reader(CCoeControl& aParent);
void ConstructL(const TDesC& aFileName);
~CGif_Reader();
CFbsBitmap* Bitmap();
protected:
    void TimerExpired(TAny* aTimer,TInt aError);
    void DoCancel();
    void RunL();
private:
    void NextImageL(void);
    void FinnishReadL(void);
private:
    CCoeControl&          iParent;
    TInt                  iCurrImg,iCurrCount;
    CImageDecoder*       iImageDecoder;
    CFbsBitmap*          iFrame;
    CFbsBitmap*          iFrameMsk;
    CFbsBitmap*          iFrameImg;
    CExampleTimer*       iExampleTimer;
    CFbsBitGc*           iGraphicsContext;
    CFbsBitmapDevice*    iBitmapDevice;
};
```