Archived:Displaying viewfinder data using ECam

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*}}.

The official example is much more useful than this. This might be worth recovering by turning into a simpler example that just covers displaying the viewfinder.

Overview

This snippet demonstrates how to display the image data received from the viewfinder of the device on the screen.

MMP file

This snippet requires the following capabilities and libraries:

```
CAPABILITY UserEnvironment
```

```
LIBRARY aknskins.lib
LIBRARY bitgdi.lib
LIBRARY ecam.lib
LIBRARY fbscli.lib
LIBRARY ws32.lib
```

Header file: CCameraEngine.h

```
#ifndef __CCAMERAENGINE_H_
#define __CCAMERAENGINE_H_

#include <ECam.h>  // CCamera, MCameraObserver
#include <GDI.H>  // TDisplayMode

#include "CCameraController.h"

class CCameraEngine : public MCameraObserver
    {
    public:  // Constructors and destructor
        /**
         * Two-phased constructor.
         */
        static CCameraEngine* NewL(CCameraController* aController);

        /**
         * Destructor.
         */
        virtual ~CCameraEngine();

    private:  // Constructors and destructor
        /**
         * Symbian OS default constructor.
         */
        CCameraEngine(CCameraController* aController);

        /**
         * Symbian OS constructor.
         */
       void ConstructL();

    private:  // Methods from base classes
        /**
         * From MCameraObserver.
         * Gets called when CCamera::Reserve() is completed.
         */
        virtual void ReserveComplete(TInt aError);

        /**
         * From MCameraObserver.
         * Gets called when CCamera::PowerOn() is completed.
         */
        virtual void PowerOnComplete(TInt aError);

        /**
         * From MCameraObserver.
         * Gets called when CCamera::StartViewFinderBitmapsL() is completed.
         */
        virtual void ViewFinderFrameReady(CFbsBitmap& aFrame);

        // Other functions from MCameraObserver omitted for brevity
        // ...

    public:  // New Functions
        /**
         * Reserves the camera.
         */
        void ReserveCameraL();

        /**
         * Called when the client rect size changes.
         *
         * @param aRect the new client rect
         */
        void ClientRectChanged(const TRect& aRect);
```


(C) Copyright Nokia 2013. All rights reserved.

```
    private:  // New functions
        /**
         * Starts the viewfinder
         */
        void StartViewFinderL();

    private:  // Data
        CCamera* iCamera;
        CCameraController* iController;
        TSize iRectSize;
    };

#endif /*__CCAMERAENGINE_H_*/
```

Source file: CCameraEngine.cpp

```
#include "CCameraEngine.h"
#include "CCameraController.h"

/*
-----------------------------------------------------------------------------
Two-phased constructor
-----------------------------------------------------------------------------
*/
CCameraEngine* CCameraEngine::NewL(CCameraController* aController)
    {
    CCameraEngine* self = new (ELeave) CCameraEngine(aController);
    CleanupStack::PushL(self);
    self->ConstructL();
    CleanupStack::Pop(self);
    return self;
    }

/*
-----------------------------------------------------------------------------
C++ default constructor
-----------------------------------------------------------------------------
*/
CCameraEngine::CCameraEngine(CCameraController* aController) :
    iController(aController)
    {
    }

/*
-----------------------------------------------------------------------------
Symbian OS 2nd phase constructor
-----------------------------------------------------------------------------
*/
void CCameraEngine::ConstructL()
    {
    // TODO: It is assumed here that the device has a camera.
    // Add error handling, if this may not be the case.

    // Camera index 0 is the main camera
    iCamera = CCamera::NewL(*this, 0);
    }

/*
-----------------------------------------------------------------------------
Destructor. Frees allocated resources.
-----------------------------------------------------------------------------
*/
CCameraEngine::~CCameraEngine()
    {
    delete iCamera;
    }

/*
-----------------------------------------------------------------------------
Called when the client rect size changes.
-----------------------------------------------------------------------------
*/
void CCameraEngine::ClientRectChanged(const TRect& aRect)
    {
    // Store the given client rect size into the member variable
    iRectSize = aRect.Size();
    }

/*
-----------------------------------------------------------------------------
Reserves the camera.
-----------------------------------------------------------------------------
*/
void CCameraEngine::ReserveCameraL()
    {
    iCamera->Reserve();

    // On completion, MCameraObserver::ReserveComplete() will be called
    }

/*
-----------------------------------------------------------------------------
Symbian Onboard Camera API observer. Gets called after CCamera::Reserve() is
called.
-----------------------------------------------------------------------------
*/
void CCameraEngine::ReserveComplete(TInt aError)
    {
    // TODO: Error handling (aError)

    iCamera->PowerOn();
    }

/*
-----------------------------------------------------------------------------
Symbian Onboard Camera API observer. Gets called after CCamera::PowerOn() is
called.
-----------------------------------------------------------------------------
*/
void CCameraEngine::PowerOnComplete(TInt aError)
    {
    // TODO: Error handling (aError)

    // Everything is ready. Start the viewfinder.
    StartViewFinderL();
```


(C) Copyright Nokia 2013. All rights reserved.

```
    }

/*
-----------------------------------------------------------------------------
Starts the viewfinder.
-----------------------------------------------------------------------------
*/
void CCameraEngine::StartViewFinderL()
    {
    // It is assumed here that the device supports viewfinding
    // (TCameraInfo::EViewFinderBitmapsSupported).

    // Bitmaps are returned by MCameraObserver::ViewFinderFrameReady().
    iCamera->StartViewFinderBitmapsL(iRectSize);
    }

/*
-----------------------------------------------------------------------------
Symbian Onboard Camera API observer. Gets called after StartViewFinderBitmapsL
is called.
-----------------------------------------------------------------------------
*/
void CCameraEngine::ViewFinderFrameReady(CFbsBitmap& aFrame)
    {
    // Pass the bitmap frame to the controller
    iController->ShowFrame(aFrame);
    }

// Other functions from MCameraObserver omitted for brevity
// ...
```

Header file: CCameraController.h

```
#ifndef __CCAMERACONTROLLER_H_
#define __CCAMERACONTROLLER_H_

#include <e32base.h>

#include "CCameraContainer.h"

// Forward declarations
class CCameraEngine;

/**
 * Interface between the UI and the external modules.
 */
class CCameraController : public CBase
    {
    public:  // Constructors and destructor
        /**
         * Two-phased constructor.
         */
        static CCameraController* NewL();

        /**
         * Destructor.
         */
        virtual ~CCameraController();

    private:
        /**
         * Symbian OS default constructor.
         */
        CCameraController();

        /**
         * Symbian OS constructor.
         */
        void ConstructL();

    public:  // New functions
        /**
         * Gives to controller a reference to active view container
         *
         * @param aContainer reference to the view container
         */
        void SetContainer(CCameraContainer* aContainer);

        /**
         * Initializes the camera.
         *
         * @param aRect
         */
        void InitializeCameraL();

        /**
         * Returns the default display mode.
         */
        TDisplayMode DisplayMode() const;

        /**
         * Displays a frame from the camera engine on the screen
         *
         * @param aFrame a frame to be displayed
         */
        void ShowFrame(CFbsBitmap& aFrame);

        /**
         * Called if the client rect size changes.
         *
         * @param aRect the new client rect
         */
        void ClientRectChanged(const TRect& aRect);

    private:  // Data
        CCameraEngine* iCameraEngine;
        CCameraContainer* iContainer; // Not owned
    };

#endif /*__CCAMERACONTROLLER_H_*/
```

Source file: CCameraController.cpp

(C) Copyright Nokia 2013. All rights reserved.

```
#include <EIKENV.H>
#include <GDI.H>

#include "CCameraController.h"
#include "CCameraEngine.h"

/*
----------------------------------------------------------------------------
Two-phased constructor
----------------------------------------------------------------------------
*/
CCameraController* CCameraController::NewL()
    {
    CCameraController* self = new (ELeave) CCameraController();
    CleanupStack::PushL(self);
    self->ConstructL();
    CleanupStack::Pop(self);
    return self;
    }

/*
----------------------------------------------------------------------------
Destructor. Frees allocated resources.
----------------------------------------------------------------------------
*/
CCameraController::~CCameraController()
    {
    delete iCameraEngine;
    }

/*
----------------------------------------------------------------------------
C++ default constructor
----------------------------------------------------------------------------
*/
CCameraController::CCameraController()
    {
    }

/*
----------------------------------------------------------------------------
Symbian OS 2nd phase constructor
----------------------------------------------------------------------------
*/
void CCameraController::ConstructL()
    {
    }

/*
----------------------------------------------------------------------------
Gives to controller the reference to active view container
----------------------------------------------------------------------------
*/
void CCameraController::SetContainer(CCameraContainer* aContainer)
    {
    iContainer = aContainer;
    }

/*
----------------------------------------------------------------------------
Initializes the camera engine
----------------------------------------------------------------------------
*/
void CCameraController::InitializeCameraL()
    {
    if (!iCameraEngine)
        {
        iCameraEngine = CCameraEngine::NewL(this);

        iCameraEngine->ReserveCameraL();
        }
    }

/*
----------------------------------------------------------------------------
Called when the client rect size changes.
----------------------------------------------------------------------------
*/
void CCameraController::ClientRectChanged(const TRect& aRect)
    {
    // Notify the engine about rect change
    iCameraEngine->ClientRectChanged(aRect);
    }

/*
----------------------------------------------------------------------------
Returns the default display mode.
----------------------------------------------------------------------------
*/
TDisplayMode CCameraController::DisplayMode() const
    {
    TInt color;
    TInt gray;
    TDisplayMode displayMode =
        CEikonEnv::Static()->WsSession().GetDefModeMaxNumColors(color, gray);
    return displayMode;
    }

/*
----------------------------------------------------------------------------
Displays a bitmap frame on the screen.
Called by the camera engine to display the bitmap.
----------------------------------------------------------------------------
*/
void CCameraController::ShowFrame(CFbsBitmap& aFrame)
    {
    iContainer->DrawImageNow(aFrame);
    }
```

Header file: CCameraContainer.h

```
#ifndef __CCAMERACONTAINER_H__
```


(C) Copyright Nokia 2013. All rights reserved.

```
#define __CCAMERACONTAINER_H__

#include <aknutils.h>
#include <coecntrl.h>

class CCameraController;

class CCameraContainer : public CCoeControl
    {
    public:  // Constructors and destructor
        /**
         * Two-phased constructor.
         */
        static CCameraContainer* NewL(CCameraController* aController,
            const TRect& aRect);

        /**
         * Two-phased constructor.
         */
        static CCameraContainer* NewLC(CCameraController* aController,
            const TRect& aRect);

        /**
         * Destructor.
         */
        virtual ~CCameraContainer();

    private:  // Constructors and destructor
        /**
         * Symbian OS default constructor.
         *
         * @param aController Camera controller
         */
        CCameraContainer(CCameraController* aController);

        /**
         * Symbian OS constructor.
         *
         * @param aRect Frame rectangle for container.
         */
        void ConstructL(const TRect& aRect);

    public:  // New functions
        /**
         * Draws the bitmap image immediately.
         *
         * @param aBitmap bitmap to be displayed
         */
        void DrawImageNow(CFbsBitmap& aBitmap);

        /**
         * Redraws the offscreen bitmap.
         */
        void ReDrawOffScreenBitmap();

    private:  // New functions
        /**
         *  Creates the offscreen bitmap.
         */
        void CreateOffScreenBitmapL();

        /**
         * Draws a bitmap onto the real screen.
         */
        void DrawImage(CWindowGc& aGc, const TRect& aRect) const;

    private: // Functions from base classes
        /**
         * From CCoeControl.
         * Called when this control's rect changes.
         */
        void SizeChanged();

        /**
         * From CCoeControl.
         */
        void Draw(const TRect& aRect) const;

    private:  // Data
        CCameraController* iController;
        CFbsBitmap* iBitmap;
        CWsBitmap* iOffScreenBitmap;
        TBool iOffScreenBitmapCreated;
        CFbsBitGc* iFbsBitGc;
        CFbsBitmapDevice* iBmpDevice;
   };

#endif // __CCAMERACONTAINER_H__
```

Source file: CCameraContainer.cpp

```
#include <aknsutils.h>

#include "CCameraContainer.h"
#include "CCameraController.h"

/*
-----------------------------------------------------------------------------
Two-phased constructor.
-----------------------------------------------------------------------------
*/
CCameraContainer* CCameraContainer::NewL(CCameraController* aController,
    const TRect& aRect)
    {
    CCameraContainer* self = CCameraContainer::NewLC(aController, aRect);
    CleanupStack::Pop(self);
    return self;
    }

/*
-----------------------------------------------------------------------------
Two-phased constructor.
-----------------------------------------------------------------------------
*/
CCameraContainer* CCameraContainer::NewLC(CCameraController* aController,
```


(C) Copyright Nokia 2013. All rights reserved.

```
    const TRect& aRect)
    {
    CCameraContainer* self = new (ELeave) CCameraContainer(aController);
    CleanupStack::PushL(self);
    self->ConstructL(aRect);
    return self;
    }

/*
----------------------------------------------------------------------------
Constructor. Initializes member variables.
----------------------------------------------------------------------------
*/
CCameraContainer::CCameraContainer(CCameraController* aController) :
    iController(aController),
    iBitmap(0),
    iOffScreenBitmap(0),
    iFbsBitGc(0),
    iBmpDevice(0)
    {
    iController->SetContainer(this);
    }

/*
----------------------------------------------------------------------------
Two-phased constructor.
----------------------------------------------------------------------------
*/
void CCameraContainer::ConstructL(const TRect& aRect)
    {
    CreateWindowL();
    SetRect(aRect);
    ActivateL();

    // This should be called after the windows has been activated
    CreateOffScreenBitmapL();
    }

/*
----------------------------------------------------------------------------
Destructor. Frees allocated resources.
----------------------------------------------------------------------------
*/
CCameraContainer::~CCameraContainer()
    {
    delete iOffScreenBitmap;
    delete iFbsBitGc;
    delete iBmpDevice;
    }

/*
----------------------------------------------------------------------------
Creates the offscreen bitmap.
----------------------------------------------------------------------------
*/
void CCameraContainer::CreateOffScreenBitmapL()
    {
    iOffScreenBitmap = new (ELeave) CWsBitmap(iCoeEnv->WsSession());

    TSize size = Rect().Size();

    TInt bmpCreateErr = iOffScreenBitmap->Create(size,
        iController->DisplayMode());
    User::LeaveIfError(bmpCreateErr);

    iFbsBitGc = CFbsBitGc::NewL();  // Graphics context
    iBmpDevice = CFbsBitmapDevice::NewL(iOffScreenBitmap);
    iFbsBitGc->Activate(iBmpDevice);
    iFbsBitGc->SetBrushColor(KRgbBlack);
    iFbsBitGc->Clear();

    MAknsSkinInstance* skin = AknsUtils::SkinInstance();
    CFbsBitmap* bitmap = AknsUtils::GetCachedBitmap(skin,
        KAknsIIDQsnBgAreaMain);
    if (bitmap)
        {
        iFbsBitGc->BitBlt(TPoint(0,0), bitmap);
        }

    iOffScreenBitmapCreated = ETrue;
    }

/*
----------------------------------------------------------------------------
Calls DrawNow to draw aBitmap to the screen.
----------------------------------------------------------------------------
*/
void CCameraContainer::DrawImageNow(CFbsBitmap& aBitmap)
    {
    if (iOffScreenBitmapCreated && IsActivated() && IsReadyToDraw())
        {
        iBitmap = &aBitmap;
        DrawNow();
        }
    }

/*
----------------------------------------------------------------------------
Called by the framework to draw the screen
----------------------------------------------------------------------------
*/
void CCameraContainer::Draw(const TRect& /*aRect*/) const
    {
    TRect drawingRect = Rect();
    CWindowGc& gc = SystemGc();

    if (iOffScreenBitmapCreated)
        {
        if (iBitmap)  // Viewfinding ongoing, draw the bitmap
            {
            DrawImage(gc, drawingRect);
            }
        else
            {
            MAknsSkinInstance* skin = AknsUtils::SkinInstance();
            CFbsBitmap* bitmap = AknsUtils::GetCachedBitmap(skin,
                KAknsIIDQsnBgAreaMain);
            if (bitmap)
                {
                gc.BitBlt(TPoint(0,0), bitmap);
```


(C) Copyright Nokia 2013. All rights reserved.

```
                }
            else
                {
                // Draws bitmap with indicators on the screen
                TSize size(iOffScreenBitmap->SizeInPixels());
                gc.BitBlt(TPoint(0,0), iOffScreenBitmap,
                        TRect(TPoint(0,0), size));
                }
            }
        }
    }

/*
-------------------------------------------------------------------------------
Renders a bitmap image onto a real screen.
-------------------------------------------------------------------------------
*/
void CCameraContainer::DrawImage(CWindowGc& aGc, const TRect& aRect) const
    {
    TSize bmpSizeInPixels(iBitmap->SizeInPixels());
    TInt xDelta = (aRect.Width() - bmpSizeInPixels.iWidth) / 2;
    TInt yDelta = (aRect.Height() - bmpSizeInPixels.iHeight) / 2;
    TPoint pos(xDelta, yDelta);  // Displacement vector
    pos += aRect.iTl;  // Bitmap top left corner position

    // Drawing viewfinder image to bitmap
    iFbsBitGc->BitBlt(pos, iBitmap, TRect(TPoint(0, 0), bmpSizeInPixels));

    // Draws bitmap with indicators on the screen
    TSize size(iOffScreenBitmap->SizeInPixels());
    aGc.BitBlt(TPoint(0,0), iOffScreenBitmap, TRect(TPoint(0,0), size));
    }

/*
-------------------------------------------------------------------------------
Redraws offscreen bitmap with landscape focus indicators.
-------------------------------------------------------------------------------
*/
void CCameraContainer::ReDrawOffScreenBitmap()
    {
    iOffScreenBitmapCreated = ETrue;

    iFbsBitGc->SetBrushColor(KRgbBlack);
    iFbsBitGc->Clear();

    MAknsSkinInstance* skin = AknsUtils::SkinInstance();
    CFbsBitmap* bitmap = AknsUtils::GetCachedBitmap(skin,
        KAknsIIDQsnBgAreaMain);
    if (bitmap)
        {
        iFbsBitGc->BitBlt(TPoint(0,0), bitmap);
        }

    DrawNow();
    }

/*
-------------------------------------------------------------------------------
Called when this control's rect changes.
-------------------------------------------------------------------------------
*/
void CCameraContainer::SizeChanged()
    {
    TRect rect = Rect();
    if(iOffScreenBitmapCreated)
        {
        iOffScreenBitmap->Resize(rect.Size());
        iBmpDevice->Resize(rect.Size());
        iFbsBitGc->Resized();
        }
    if (iController)
        {
        iController->ClientRectChanged(rect);
        }
    }
```

Postconditions

Image data received from the viewfinder is displayed on the screen.

See also

- Archived:Capturing an image using the Symbian Camera API
- S60 Platform: Camera Example with AutoFocus Support

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.