Archived:Using drawable windows without Symbian application framework

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

Using drawable windows without application framework

Description

When developing a Symbian GUI application, application framework's control environment sets up a window system automatically, providing access to window server session, screen device, graphics context, window group, etc.
If an application that does not use application framework wants to draw to the screen using a graphics context, all these resources must be constructed manually.

Solution

The following example code demonstrates these steps:
- Connecting to window server (RWsSession)
- Creating a screen device (CWsScreenDevice) and a graphics context (CWindowGc)
- Creating a root (group) window (RWindowGroup) with a high ordinal priority to make it a 'stay on top' window
- Creating a window with specific properties
- Drawing to the window
- Window server event handling

The custom graphics class is derived from CActive. This makes it easy to listen to events from the window server which provides asynchronous notification methods for several types of events. Currently, only redraw events (TWsRedrawEvent) are being handled. Another useful event type to handle would be screen size/mode change event. These events can be activated by calling RWindowGroup::EnableScreenChangeEvents(), and requested from the window server by calling RWsSession::EventReady() and RWsSession::GetEvent(). See the SDK Help for more information about window server events.
On S60 3rd Edition, the following code requires WriteDeviceData capability.

```
-----
#include <e32base.h>
#include <w32std.h>     // ws32.lib
#include <coedef.h>
class CCustomGraphics : public CActive
    {
public:
    static CCustomGraphics* NewL();
    void CustomDraw( TRect& aRect );
    virtual ~CCustomGraphics();
protected:
    CCustomGraphics();
    void    ConstructL();
    // from CActive
    void    DoCancel();
    void    RunL();
protected:
    RWsSession       iWsSession;
    CWsScreenDevice* iScreenDevice;
    RWindowGroup     iWindowGroup;
    RWindow*         iWindow;
    CWindowGc*       iWindowGc;
    };
CCustomGraphics::CCustomGraphics()
: CActive( CActive::EPriorityLow )
    {
    CActiveScheduler::Add( this );
    }
CCustomGraphics* CCustomGraphics::NewL()
    {
    CCustomGraphics* self = new (ELeave) CCustomGraphics;
    CleanupStack::PushL( self );
    self->ConstructL();
    CleanupStack::Pop(); // self;
    return self;
```


(C) Copyright Nokia 2013. All rights reserved.

```
    }
void CCustomGraphics::ConstructL()
    {
    // Connect to window server session
    User::LeaveIfError( iWsSession.Connect() );
    // Set client priority permanently to current level. See documentation on
    // RWsSession for more information
    iWsSession.ComputeMode( RWsSession::EPriorityControlDisabled );
    iScreenDevice = new (ELeave) CWsScreenDevice( iWsSession );
    User::LeaveIfError( iScreenDevice->Construct() );
    iScreenDevice->SetScreenModeEnforcement( ESizeEnforcementNone );
iWindowGroup = RWindowGroup( iWsSession );
    User::LeaveIfError( iWindowGroup.Construct( (TUint32)this, ETrue ) );
    // This window group cannot receive keyboard focus
    iWindowGroup.EnableReceiptOfFocus( EFalse );
    // Window group will always be on top
    iWindowGroup.SetOrdinalPosition( 0, ECoeWinPriorityAlwaysAtFront+1 );
    // Create graphics context for the screen device
    User::LeaveIfError(iScreenDevice->CreateContext( iWindowGc ) );

    // Finally, create a window, set its properties, and display it
    iWindow = new (ELeave) RWindow( iWsSession );
    User::LeaveIfError(iWindow->Construct( iWindowGroup, (TUint32)iWindow ) );
    iWindow->SetExtent( TPoint( 10, 10 ), TSize( 100, 100 ) );
    iWindow->SetOrdinalPosition(0, ECoeWinPriorityAlwaysAtFront+1 );
    iWindow->SetCornerType( EWindowCorner5 );
    iWindow->SetNonFading( ETrue );
    iWindow->SetVisible( ETrue );
    iWindow->Activate();
    // Start waiting for redraw events
    iWsSession.RedrawReady( &iStatus );
    SetActive();
    }
CCustomGraphics::~CCustomGraphics()
    {
    Cancel();
    iWindow->Close();
    delete iWindow;
    iWindowGroup.Close();
    delete iWindowGc;
    delete iScreenDevice;
    iWsSession.Close();
    }
void CCustomGraphics::RunL()
    {
    if( iStatus == KErrNone )
        {
        TWsRedrawEvent event;
        iWsSession.GetRedraw( event );
        // Handle of the window which is the target of redraw event
        // Can be used for ID purposes if multiple windows are used
        RWindow* window = ( RWindow* )( event.Handle() );
        // Rectangle that needs to be redrawn
        TRect eventRect = event.Rect();
        CustomDraw( eventRect );
        iWsSession.RedrawReady( &iStatus );
        SetActive();
        }
    }
void CCustomGraphics::DoCancel()
    {
    iWsSession.RedrawReadyCancel();
    }
void CCustomGraphics::CustomDraw( TRect& /*aRect*/ )
    {
    iWindowGc->Activate( *iWindow );
    iWindow->BeginRedraw();
    iWindowGc->SetBrushStyle( CGraphicsContext::ESolidBrush );
    iWindowGc->SetBrushColor( TRgb( 255, 0, 0 ) );
    iWindowGc->Clear();
    iWindow->EndRedraw();
    iWindowGc->Deactivate();
    iWsSession.Flush();
    }
```

Notes:

The order of construction for CWsScreenDevice and RWindowGroup is important. For the window to work in different screen modes, the CWsScreenDevice should be created before the RWindowGroup object.

This is because on the server side, each group window is associated with a screen device. If the client hasn't created a screen device, there won't be an association.

 (C) Copyright Nokia 2013. All rights reserved.

Each screen device is registered to work in a screen size mode, which can be changed by the client. All windows under a window group will be visible only if the associated screen device is registered to work in a mode that is different from the one the device is currently operating in (there are some exceptions to this). If a group window doesn't have a screen device, it will be registered to work in the first mode (0).

Thus:

- Group windows created before the screen device will only be seen in the first screen size mode.
- Group windows created after the screen device will be visible in all modes, as the registered mode is determined by the screen device.

 (C) Copyright Nokia 2013. All rights reserved.

 (C) Copyright Nokia 2013. All rights reserved.