

## Drawing\_to\_the\_screen\_outside\_application\_framework

The **CWindowDrawer** class illustrates how to construct a drawable screen outside the standard application framework. You usually use this kind of code when constructing a background server that normally needs to be hidden and only requires an UI in some parts of its operations.

As with this example the **CWindowDrawer** will not take focus which means that the current focused application still receives all key input. To disable this call the *EnableReceiptOfFocus()* method with **ETrue**.

For key catching you can also use [CKeyCatcher](#)

The active object implementation used in this example uses RWSession's *RedrawReady()* method to receive redraw events. To get your own drawing code into it you only need to change the content of the *Draw()* method.

Also if you don't want to take the whole screen for your window you can change the values given in **iWindow**'s *SetExtent()* method inside the *ConstructL()* method.

### Library required:

```
LIBRARY   gdi.lib //CFont
LIBRARY   ws32.lib //RWindowGroup ,CWScreenDevice, CWindowGc
LIBRARY   apgrfx.lib //CApaWindowGroupName
```

## WindowDrawer.cpp

```
#include <apgwgnam.h> //CApaWindowGroupName
#include "WindowDrawer.h"
#include <coedef.h> //for ECoeWinPriorityAlwaysAtFront

CWindowDrawer* CWindowDrawer::NewL(void)
{
    CWindowDrawer CWindowDrawer::NewLC(void);
    CleanupSpace(self);
    return self;
}

CWindowDrawer* CWindowDrawer::NewLC(void)
{
    CWindowDrawer new (ELeave) CWindowDrawer(void);
    CleanupSpace(self);
    ->ConstructL();
    return self;
}

// the active object priority should be relatively low
CWindowDrawer::CWindowDrawer(void): CActive(EPriorityLow)
{
}

CWindowDrawer::~CWindowDrawer()
{
    ()Cancel

    if(iMyFont)
    {
        ->ReleaseFont(iMyFont);
    }
}
```

## Drawing\_to\_the\_screen\_outside\_application\_framework

```

        = NULL; iMyFont
    }

delete iWindowGc;
    iWindowGc;
delete iScreenDevice;
    iScreenDevice;

    CWindow();

    iWindowGroup();
    iWindowGroup();
    iWindowGroup();
}

void CWindowDrawer::ConstructL(void)
{
    CActiveSchedulerAdd(this);

    ::LeaveIfError(iWsSession.Connect());

    iScreenDevice(iWLeave) CWScreenDevice(iWsSession);
    ::LeaveIfError(iScreenDevice->Construct());
    ::LeaveIfError(iScreenDevice->CreateContext(iWindowGc));

    TFontSpec MyFont(KESpeArial, 12*15);
    MyFont.Face.SetIsProportional(ETTrue);
    ::LeaveIfError(iScreenDevice->GetNearestFontInTwips(iMyFont, MyFontSpec));

    iWindowGroup(iWLeave) CWindowGroup(iWsSession);
    ::LeaveIfError(iWindowGroup.Construct((TUint32)&iWindowGroup, EFalse));
// iWindowGroup.SetOrdinalPosition(0, ECoeWinPriorityNormal);
    iWindowGroup.SetOrdinalPosition(0, ECoeWinPriorityAlwaysAtFront);
    iWindowGroup.AcceptOfFocus(EFalse);

    CAppWindowGroupName=CAppWindowGroupName::NewLC(iWsSession);
    winGroup.Hidden(ETTrue);
    winGroup.WindowGroup(iWindowGroup);
    CleanupStackAndDestroy();

    iWindow(iWLeave) CWindow(iWsSession);
    ::LeaveIfError(iWindow.Construct(iWindowGroup, (TUint32)&iWindow));

    TPixelsTwipsAndRotation SizeAndRotation;
    iScreenDevice.DefaultScreenSizeAndRotation(SizeAndRotation);

    iWindow();
    iWindow.SetPosition(TPoint(0,0), SizeAndRotation.iPixelSize);
    iWindow.SetBackgroundColor(KRgbBlue);
// iWindow.SetOrdinalPosition(0, ECoeWinPriorityNormal);
    iWindow.SetOrdinalPosition(0, ECoeWinPriorityAlwaysAtFront);
    iWindow.Wading(ETTrue);
    iWindow.Wable(ETTrue);

    (); Draw
    iWindowReady(&iStatus);
    iWindowActive
}

void CWindowDrawer::RunL()
{
    if (iStatus != KErrCancel)
    {

```

## Drawing\_to\_the\_screen\_outside\_application\_framework

```
        TWSRedrawEvent e
        GetRedrawSession.

// if Windows Server does not want a redraw the window handle is 0
if (e.Handle() != 0)
{
// draw our only window
    Draw
    ();
}

    RedrawReady(iDstStatus);
    ();    SetActive
}
}

void CWindowDrawer::Draw(void)
{
    iWindowGate(iWindow);

    TRect(DrRect(0,0), iWindow.Size());

    iWindowGate(DrwRect);
    BeginDraw();
    iWindowGate(DrwRect);

if(iMyFont)
{
    ->UseFont(iMyFont);

    ((DrRect.Height() - iMyFont->HeightInPixels()) / 2);

    ->DrawText(KtxHelloThere, TRect(0, StartY, DrwRect.Width(), (StartY + iMyFont->HeightInPixel
}

    EndDraw();

    iWindowGate();
    iWSession();
}

void CWindowDrawer::DoCancel()
{
    iWSessionReadyCancel();
}
```

## WindowDrawer.h

```
#include <gdi.h> // CFont
#include <w32std.h> //RWindowGroup, CWScreenDevice, CWindowGc

_LIT(KtxHelloThere "Hi there !");
_LIT(KFontArial "Arial");

class CWindowDrawer : public CActive
{
public:
    static CWindowDrawer* NewL(void);
    static CWindowDrawer* NewLC(void);
    ~CWindowDrawer
protected:
```

## Drawing\_to\_the\_screen\_outside\_application\_framework

```
void DoCancel();
void RunL();
private:
    CWindowDrawer
void ConstructL(void);
void Draw(void);
private:
    RWsSession      ;          iWsSession
    CWindowGc      ;          iWindowGc
    CWScreenDevice iScreenDevice
    RWindowGroup   ;          iWindowGroup
    RWindow        ;          iWindow
    * CFont;           iMyFont
};
```

## Related Links:

- [How to get ReDraw event in exe from window server?](#)
- [How to capture Keyevents in thread or exe](#)
- [How to Launch an EXE and Pass Command Line Arguments](#)
- [How to start EXE from an EXE in 3rd Edition](#)
- [Displaying controls in Symbian exe programs](#)
- [Graphics in EXE](#)