



Original article written at <http://pushl.com/developers/exeui.html>

Generally, exe programs are used to implement servers (exedll, epocexe target type) or simple console programs. This may lead to believe that graphical features are not available for exe programs. This is not the case, and even though it might take some extra work to set up an appropriate environment, this is quite straight forward to achieve, as the following snippet will illustrate. This way, we can make use of graphics without having to resort on a full blown application.

Our aim is to simply emulate the minimum set up provided by the framework when an application (.app) is loaded. Being a dll, an application needs a process to attach to. This is provided by AppRun.exe, which besides catering its own process and thread, calls EikDll::RunAppInsideThread(), which -among other things- creates a CONE environment (CCoeEnv)

When a CONE Environment is created, it sets up a cleanup stack, active scheduler (CCoeScheduler), and sessions to the file and window server (which can be later retrived by calling CCoeEnv::FsSession() and CCoeEnv::WsSession()). Then it creates a screen, a window group and a graphic context, among other things. Later, after the NewApplication() factory function is called by the framework, the CCoeAppUi (from where the application's AppUi derives) creates the control stack, view manager, etc. and the CCoeControl derived class will create a window (RWindow) to provide a basic view. As you can see, this snippet provides all this basic functionality at once, without extra bloat such as a control stack, etc. which aren't needed in this simple case. I haven't included comments, as most of the code speaks by itself.

Headers Required:

```
#include <w32std.h> //RWSession, CWSscreenDevice, RWindowGroup, CWindowGc, RWindow
#include <gdi.h> //CFont, TFontSpec
```

Library Needed:

```
LIBRARY ws32.lib //RWSession, CWSscreenDevice, RWindowGroup, CWindowGc, RWindow
LIBRARY gdi.lib //CFont, TFontSpec
```

Source:

```
LOCAL_C void ExeMainL()
{
    RWSession ws;
    User::LeaveIfError(ws.Connect());
    CleanupClosePushL(ws);

    CWSscreenDevice* screen = new(ELeave) CWSscreenDevice(ws);
    CleanupStack::PushL(screen);
    screen->Construct();

    RWindowGroup wg(ws);
    User::LeaveIfError(wg.Construct(reinterpret_cast<TUint32>(&wg), EFalse));
    CleanupClosePushL(wg);

    wg.SetOrdinalPosition(10, ECoeWinPriorityAlwaysAtFront);

    CWindowGc* gc;
    User::LeaveIfError(screen->CreateContext(gc));
    CleanupStack::PushL(gc);
}
```

Graphics_in_EXE

```
RWindow window(ws);
User::LeaveIfError(window.Construct(wg, reinterpret_cast<TUint32>(&wg) + 1));
CleanupClosePushL(window);
window.SetBackgroundColor(TRgb(0x90, 0x90, 0x90));
window.Activate();
window.SetExtent(TPoint(0, 0), TSize(screenWidth, screenHeight));
window.SetVisible(ETrue);
//if you want to receive event on change in focus, use EnableFocusChangeEvents and implement
//particularly helpful when some app goes to background and ur exe came to foreground.
window.EnableFocusChangeEvents();

gc->Activate(window);

//create font if u want draw text in exe.
CFont* font = NULL;
TFontSpec myFontSpec(_L("Series 60 Sans Regular"), 100);
screen ->GetNearestFontInTwips(font, myFontSpec);
screen->ReleaseFont(font);
// now use this fone.

TRect rect = TRect(window.Size());
window.Invalidate(rect);
window.BeginRedraw(rect);

gc->SetBrushStyle(CGraphicsContext::ESolidBrush);
gc->Clear();

TInt64 seed = User::TickCount();
TRgb color[] = { KRgbRed, KRgbGreen, KRgbBlue, KRgbYellow, KRgbCyan };

for (TUint i = 0; i < 10000; ++i)
{
    TInt x = Math::Rand(seed) % screenWidth;
    TInt y = Math::Rand(seed) % screenHeight;
    TInt w = Math::Rand(seed) % 50;
    TInt h = Math::Rand(seed) % 25;

    TRect rect(TPoint(x, y), TSize(w, h));
    gc->SetBrushColor(color[i % sizeof color]);
    gc->DrawRect(rect);
}

window.EndRedraw();
gc->Deactivate();

ws.Flush();

CleanupStack::PopAndDestroy(5, &ws);    // window, gc, wg, screen, ws
}
```

Let's see now the entry point of the program. If we tried something like this

```
GLDEF_C TInt E32Main()
{
    __UHEAP_MARK;
    CTrapCleanup* cleanup = CTrapCleanup::New();

    TRAPD(error, ExeMainL());
    __ASSERT_ALWAYS(!error, User::Panic(_L("EXEUI"), error));

    delete cleanup;
    __UHEAP_MARKEND;
}
```

Graphics_in_EXE

```
    return 0;
}
```

it would work on the device only. On the emulator, you'd get a panic, as `RWsSession::Connect()` would fail. The reason for this is that when an exe is executed, the emulator doesn't launch the window server as part of its initialization, whereas on the device the server is already loaded. There're a couple of ways to solve this. On Series 60 SDK v1.2, there's an undocumented function which does exactly this for us. `RegisterWsExe()` is provided by `wserv.lib`. So now our entry point function is:

```
void RegisterWsExe(const TDesC&);

GLDEF_C TInt E32Main()
{
    __UHEAP_MARK;
    CTrapCleanup* cleanup = CTrapCleanup::New();

#ifdef __WINS__
    RegisterWsExe(_L("ExeUI.exe"));
#endif

    TRAPD(error, ExeMainL());
    __ASSERT_ALWAYS(!error, User::Panic(_L("EXEUI"), error));

    delete cleanup;
    __UHEAP_MARKEND;

    return 0;
}
```

The problem with this solution is that it's not as generic as we might want. There's another way, which requires some more bit of work, that consists of changing the target type to `exedll` or `epocexe`. This means that on `WINS` (emulator) we get a `.dll`, and on `MARM` (target) we get the `.exe` as usual. Note that all changes will be made to the startup code, that is, no changes are made to `ExeMainL()` code. This is how our new startup code looks, with the two functions required by the `exedll` type (`epocexe` is analogous, except the exported function `InitEmulator()` is replaced by `WinsMain()`):

```
GLDEF_C TInt E32Main()
{
    __UHEAP_MARK;
    CTrapCleanup* cleanup = CTrapCleanup::New();

    TRAPD(error, ExeMainL());
    __ASSERT_ALWAYS(!error, User::Panic(_L("EXEUI"), error));

    delete cleanup;
    __UHEAP_MARKEND;

    return 0;
}

// If using exedll target type
#ifdef __WINS__
EXPORT_C TInt InitEmulator()
{
    E32Main();
    User::Exit(0);
    return KErrNone;
}
#endif
```

Graphics_in_EXE

```
TInt E32Dll(TDllReason)
{
    return KErrNone;
}
#endif
```

Now the idea is renaming ExeUI.dll to ExeUI.app, and copying it to an application directory on the emulator tree to make it look like an ordinary application. So we create the ExeUI directory in Epc32\Wins\c\system\Apps (or Epc32\Release\wins\udeb\z\system\apps if you wish). As we're creating an application, we also need to provide it of a proper UID, so don't forget to add the corresponding line in the .mmp Now you may launch the emulator, that will show the ExeUI program's default icon.

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](#)