



If we now wanted to create and display a CCoeControl, we'd need to have a CONE environment (CCoeEnv) as a minimum, as this class is used internally by the control. This makes things easier, given that CCoeEnv, as its name implies, creates a whole environment for us, thus limiting our task to the proper control creation:

Library required:

```
LIBRARY euser.lib //CTrapCleanup
LIBRARY ws32.lib //RWSession,RWindow
LIBRARY cone.lib //CCoeEnv
LIBRARY eikcoctl.lib //CEikTextListBox,CTextListBoxModel
```

Source:

```
#include <coedef.h> // TCoeWinPriority
#include <e32base.h> // CTrapCleanup
#include <w32std.h> // RWSession
#include <coecntrl.h> // CCoeControl
#include <coemain.h> // CCoeEnv

class CMyControl : public CCoeControl
{
public:
    void ConstructL(const TRect& aRect);

private:
    void Draw(const TRect& aRect) const;
};

void CMyControl::ConstructL(const TRect& aRect)
{
    CreateWindowL();

    SetRect(aRect);
    ActivateL();
}

void CMyControl::Draw(const TRect& aRect) const
{
    // Just paint it blue

    CWindowGc& gc = SystemGc();
    gc.SetBrushStyle(CGraphicsContext::ESolidBrush);
    gc.SetBrushColor(KRgbBlue);
    gc.SetPenStyle(CGraphicsContext::ENullPen);

    gc.Clear(aRect);
}

// S60 screen size
const TUint screenWidth = 176;
const TUint screenHeight = 208;

LOCAL_C void ExeMainL()
{
    CCoeEnv::Static()->RootWin().SetOrdinalPosition(0, ECoeWinPriorityAlwaysAtFront);

    CMyControl* ctrl = new(ELeave) CMyControl;
    CleanupStack::PushL(ctrl);
}
```

Displaying_controls_in_Symbian_exe_programs

```
ctrl->ConstructL(TRect(0, 0, screenWidth, screenHeight));
ctrl->DrawNow();

CCoeEnv::Static()->WsSession().Flush();

CleanupStack::PopAndDestroy(ctrl);
}

GLDEF_C TInt E32Main()
{
#ifdef __WINS__
    // WINS already creates environment for us
    CCoeEnv* coe = CCoeEnv::Static();
#else
    CCoeEnv* coe = new CCoeEnv;
    TRAPD(err, coe->ConstructL());
    __ASSERT_ALWAYS(!err, User::Panic(_L("EXECTRL"), err));
#endif

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

    User::After(3*1000*1000);

#ifdef !__WINS__
    coe->DestroyEnvironment();
#endif

    return 0;
}

#ifdef __WINS__
EXPORT_C TInt WinsMain()
{
    E32Main();
    User::Exit(0);
    return KErrNone;
}

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

Note that the emulator creates on our behalf an environment when loading an exedll/epocexe as .app, so we have to avoid creating a new environment on emulator builds, otherwise, we'd get a CONE 2 panic (Environment already exists)

Now moving a step further, we'd like to use existing controls, such as listboxes, dialogs, etc. In most cases, a CONE environment won't be enough, so we'll have to resort on the Eikon environment:

```
#include <coedef.h>           // TCoeWinPriority
#include <e32base.h>          // CTrapCleanup
#include <w32std.h>           // RWsSession
#include <coecntrl.h>         // CCoeControl
#include <eikenv.h>           // CEikonEnv
#include <eiktvlbx.h>         // CEikTextListBox
#include <eiktvlbm.h>         // CTextListBoxModel
```

Displaying_controls_in_Symbian_exe_programs

```
class CMyControl : public CCoeControl
{
public:
    void ConstructL(const TRect& aRect);
    ~CMyControl();

private:
    void SizeChanged();
    TInt CountComponentControls() const;
    CCoeControl* ComponentControl(TInt aIndex) const;

    void Draw(const TRect& aRect) const;

private:
    CEikTextListBox* iListBox;
};

void CMyControl::ConstructL(const TRect& aRect)
{
    CreateWindowL();

    // You may use CAknSingleStyleListBox, etc for Series 60..
    iListBox = new(ELeave) CEikTextListBox;
    iListBox->SetContainerWindowL(*this);
    // iListBox->SetMopParent(this);
    iListBox->ConstructL(this);

    CDesCArray* items = static_cast<CDesCArray*> (iListBox->Model()->ItemTextArray());
    _LIT(KItem1, "First");
    items->AppendL(KItem1);
    _LIT(KItem2, "Second");
    items->AppendL(KItem2);

    iListBox->HandleItemAdditionL();
    iListBox->SetFocus(ETrue);

    // iListBox->CreateScrollBarFrameL();
    // iListBox->ScrollBarFrame()->SetScrollBarVisibilityL(CEikScrollBarFrame::EOff, CEikScrollBarFrame::EOff);

    SetRect(aRect);
    ActivateL();
}

CMyControl::~CMyControl()
{
    delete iListBox;
}

void CMyControl::SizeChanged()
{
    iListBox->SetRect(Rect());
}

TInt CMyControl::CountComponentControls() const
{
    return 1;
}

CCoeControl* CMyControl::ComponentControl(TInt aIndex) const
{
    switch (aIndex)
```

Displaying_controls_in_Symbian_exe_programs

```
{
    case 0:
        return iListBox;

    default:
        return 0;
}
}

void CMyControl::Draw(const TRect& aRect) const
{
    CWindowGc& gc = SystemGc();
    gc.Clear(aRect);
}

// S60 screen size
const TUint screenWidth = 176;
const TUint screenHeight = 208;

LOCAL_C void ExeMainL()
{
    CCoeEnv::Static()->RootWin().SetOrdinalPosition(0, ECoeWinPriorityAlwaysAtFront);

    CMyControl* ctrl = new(ELeave) CMyControl;
    CleanupStack::PushL(ctrl);

    ctrl->ConstructL(TRect(0, 0, screenWidth, screenHeight));
    ctrl->DrawNow();

    CCoeEnv::Static()->WsSession().Flush();

    CleanupStack::PopAndDestroy(ctrl);
}

GLDEF_C TInt E32Main()
{
    #if defined(__WINS__)
        // WINS already creates environment for us
        CEikonEnv* coe = CEikonEnv::Static();
    #else
        CEikonEnv* coe = new CEikonEnv;
        TRAPD(err, coe->ConstructL());
        __ASSERT_ALWAYS(!err, User::Panic(_L("EXECTRL"), err));
    #endif

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

    User::After(3*1000*1000);

    #if !defined(__WINS__)
        delete coe;
    #endif

    return 0;
}

#if defined(__WINS__)
EXPORT_C TInt WinsMain()
{
    E32Main();
    User::Exit(0);
}
#endif
```

Displaying_controls_in_Symbian_exe_programs

```
        return KErrNone;
    }

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

The example above shows a pretty simple compound control, consisting of just one lodger control, a text listbox. But to make this a bit more useful, we'd need to receive events (such as key presses) and pass them to the control. The example below shows a simple implementation. Actually, it mimics somehow `CCoeEnv::ExecuteD()` functionality, which basically starts the active scheduler and waits for events from the window server (note that `CCoeEnv` is a `CActive`)

```
LOCAL_C void ExeMainL()
{
    CCoeEnv* coeEnv = CCoeEnv::Static();
    coeEnv->RootWin().SetOrdinalPosition(0, ECoeWinPriorityAlwaysAtFront);

    CMyControl* ctrl = new(ELeave) CMyControl;
    CleanupStack::PushL(ctrl);

    ctrl->ConstructL(TRect(0, 0, screenWidth, screenHeight));
    ctrl->DrawNow();

    coeEnv->WsSession().Flush();

    // Create a basic UI and set control to receive events
    CMyAppUi* appUi = new(ELeave) CMyAppUi;
    appUi->SetRootControl(ctrl);
    coeEnv->SetAppUi(appUi);    // takes ownership

    for (;;)
    {
        // Wait synchronously for event
        TRequestStatus status;
        coeEnv->WsSession().EventReady(&status);
        User::WaitForRequest(status);

        if (status.Int() == KErrNone)
        {
            TWsEvent event;
            coeEnv->WsSession().GetEvent(event);

            // Check exit key
            if (event.Key()->iCode == EKeyDevice3)
                break;

            // Pass event to control
            appUi->HandleWsEventL(event, ctrl);
        }
    }

    CleanupStack::PopAndDestroy(ctrl);
}
```

You May Download the examples here:

[ExeCtrl.zip \(CONE environment\)](#)

[ExeCtrl.zip \(Eikon environment\)](#)

Related Links:

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