

## How\_to\_draw\_image\_to\_screen\_directly

That sample doesn't satisfy naming convention. You sure it can be "Reviewer Approved"?

**Reviewer Approved**



The speed of image drawing is very important in game development, but the Symbian OS window frame does not offer full support for games. We need to access the screen buffer directly to provide fluent animation.

### Library Needed:

```
LIBRARY ws32.lib //RWSession, CWScreenDevice, RWindowGroup, CDirectScreenAccess
LIBRARY bitgdi.lib //CFbsBitGc
```

## Display.h

```
#include <w32std.h> //RWSession, CWScreenDevice, RWindowGroup, CDirectScreenAccess
#include <bitstd.h> //CFbsBitGc

class CDisplay : public MDirectScreenAccess
{
public:
static CDisplay*NewL( RWSession& pClient, CWScreenDevice& pScreenDevice, RWindow& pWindow );
static CDisplay*NewLC( RWSession& pClient, CWScreenDevice& pScreenDevice, RWindow& pWindow );
    ~CDisplay()

inline CWScreenDevice& GetScreenDevice()
{
return m_pScreenDevice;
}

inline RWindow& GetWindow()
{
return m_pWindow;
}

inline CFbsBitGc* GetScreenGc()
{
return m_pGc;
}

void Start();
void Stop();
void Update();

private:
    CDisplay( RWSession& pClient, CWScreenDevice& pScreenDevice, RWindow& pWindow );
void ConstructL();
void Restart( RDirectScreenAccess::TTerminationReasons aReason );
void AbortNow( RDirectScreenAccess::TTerminationReasons aReason );

private:
    CWScreenDevice m_pScreenDevice
    RWSession ; m_pClient
```

## How\_to\_draw\_image\_to\_screen\_directly

```
&RWindow ;      m_pWindow

    CDirectScreenAccess m_pDirectScreenAccess
    *RRegion ;      m_pRegion
    CFbsBitMap * m_pGc

    TBool ;      m_bDrawing
};
```

## Display.cpp

```
#include "Display.h"
```

```
CDisplay* CDisplay::NewL( RWsSession& pClient, CWScreenDevice& pScreenDevice, RWindow& pWindow )
{
    CDisplay* NewLC( pClient, pScreenDevice, pWindow );
    CleanupStack();
    return self;
}
```

```
CDisplay* CDisplay::NewLC( RWsSession& pClient, CWScreenDevice& pScreenDevice, RWindow& pWindow )
{
    CDisplay* new ( ELeave ) CDisplay( pClient, pScreenDevice, pWindow );
    CleanupStack();
    ->ConstructL();
    return self;
}
```

```
CDisplay::CDisplay( RWsSession& pClient, CWScreenDevice& pScreenDevice, RWindow& pWindow ) :
m_pClient( pClient ),
m_pScreenDevice( pScreenDevice ),
m_pWindow( pWindow )
{
    m_pDirectScreenAccess = NULL;
    m_pRegion = NULL;
    m_pGc = NULL;

    m_bDrawing;
}
```

```
CDisplay::~CDisplay()
{
    delete m_pDirectScreenAccess;
}
```

```
void CDisplay::ConstructL()
{
    m_pDirectScreenAccess = CDirectScreenAccess::NewL( m_pClient, m_pScreenDevice, m_pWindow, *this )
}
```

```
void CDisplay::Start()
{
    if( !m_bDrawing )
    {
        ( dsaErr, m_pDirectScreenAccess->StartL() );
        if( dsaErr == KErrNone )

```

## How\_to\_draw\_image\_to\_screen\_directly

```
{
    = m_pDirectScreenAccess->Gc();
    = m_pDirectScreenAccess->DrawingRegion();
->SetClippingRegion(m_pGpRegion);
    = ETrue;    m_bDrawing
}
}
}

void CDisplay::Stop()
{
if( m_bDrawing )
{
    = EFalse; m_bDrawing
}
}

void CDisplay::Update()
{
    m_pDirectScreenAccess->Update();
}

void CDisplay::Restart( RDirectScreenAccess::TTerminationReasons /*aReason*/ )
{
    ();Start
}

void CDisplay::AbortNow( RDirectScreenAccess::TTerminationReasons /*aReason*/ )
{
    m_pDirectScreenAccess->();
    m_bDrawing;
}
}
```