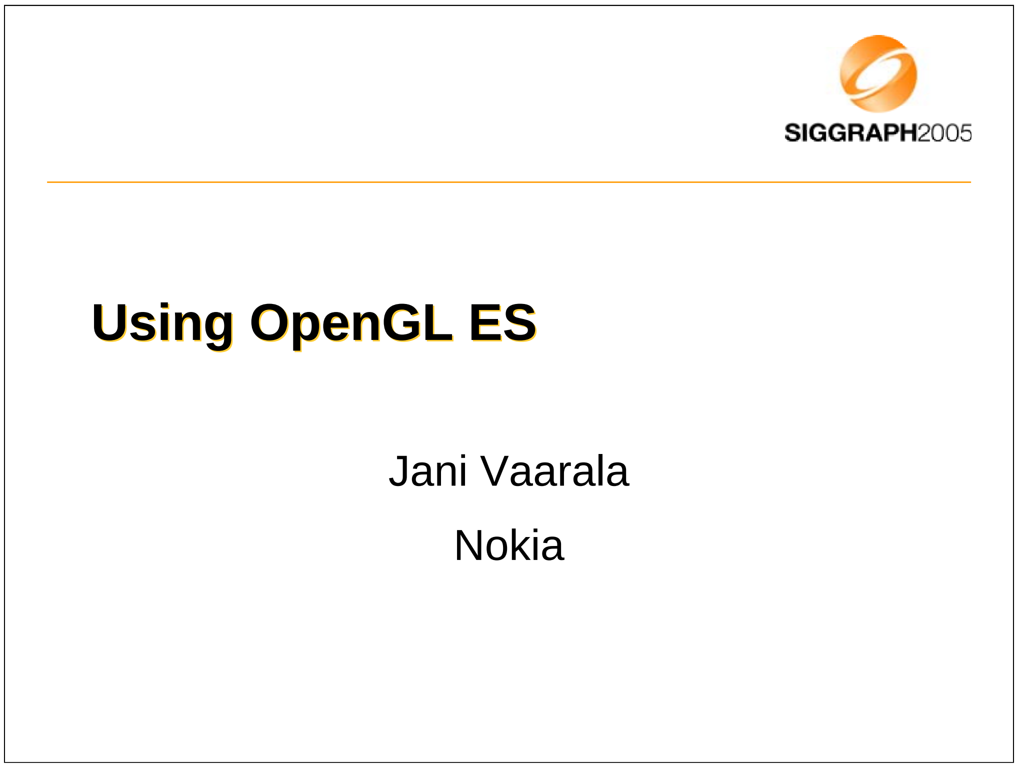
SIGGRAPH2005

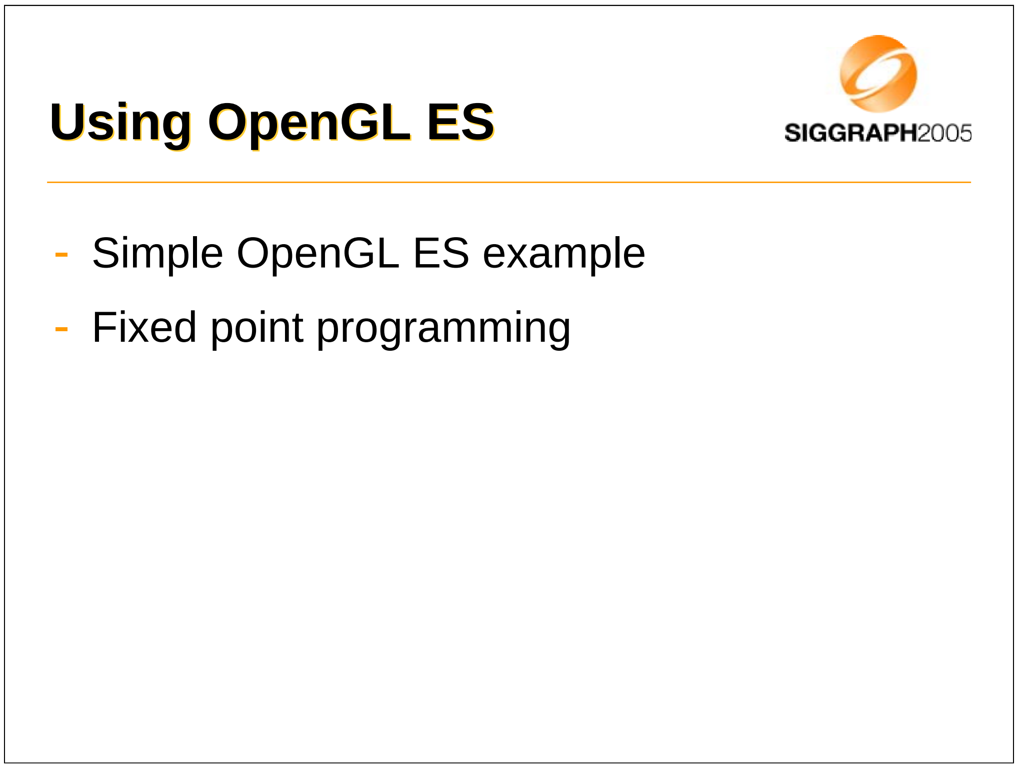


SIGGRAPH2005

Using OpenGL ES

Jani Vaarala

Nokia



Using OpenGL ES

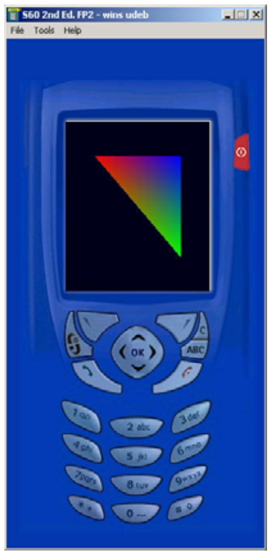
SIGGRAPH2005

- Simple OpenGL ES example
- Fixed point programming

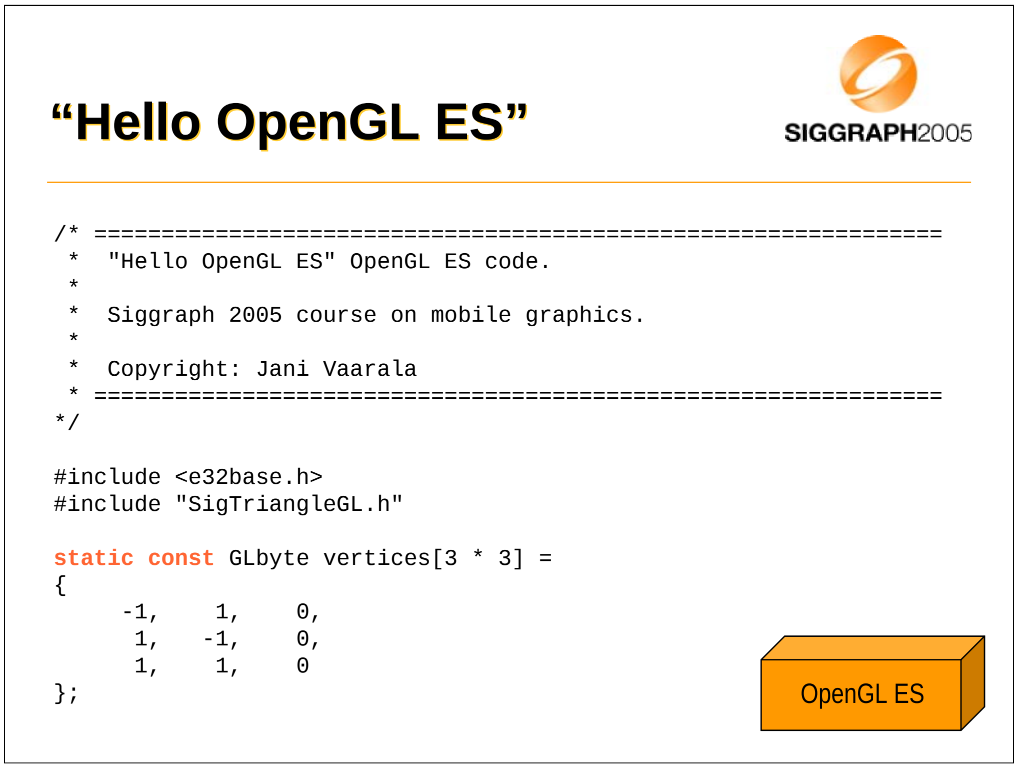
-We will use Symbian as an example, as there are already openly programmable devices out there that come with preinstalled OpenGL ES support.

“Hello OpenGL ES”

SIGGRAPH2005



-This is what we are aiming for: single smooth shaded triangle on the emulator (and on the device).



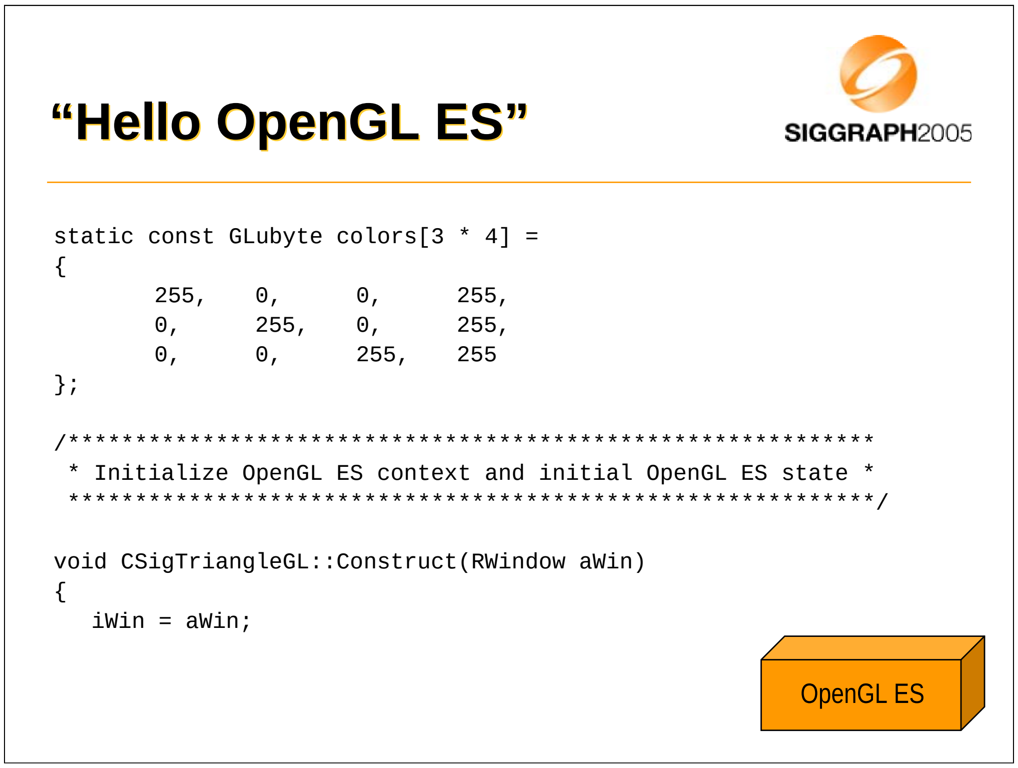
“Hello OpenGL ES”

SIGGRAPH2005

```
/* ================================================================
 *  "Hello OpenGL ES" OpenGL ES code.
 *
 *  Siggraph 2005 course on mobile graphics.
 *
 *  Copyright: Jani Vaarala
 * ================================================================
*/

#include <e32base.h>
#include "SigTriangleGL.h"

static const GLbyte vertices[3 * 3] =
{
     -1,    1,    0,
      1,   -1,    0,
      1,    1,    0
};
```

-First we define 3 vertices of a triangle.

-We use static const for two reasons: it’s a good habit to mark it as const for compiler and under Symbian global data is not allowed.



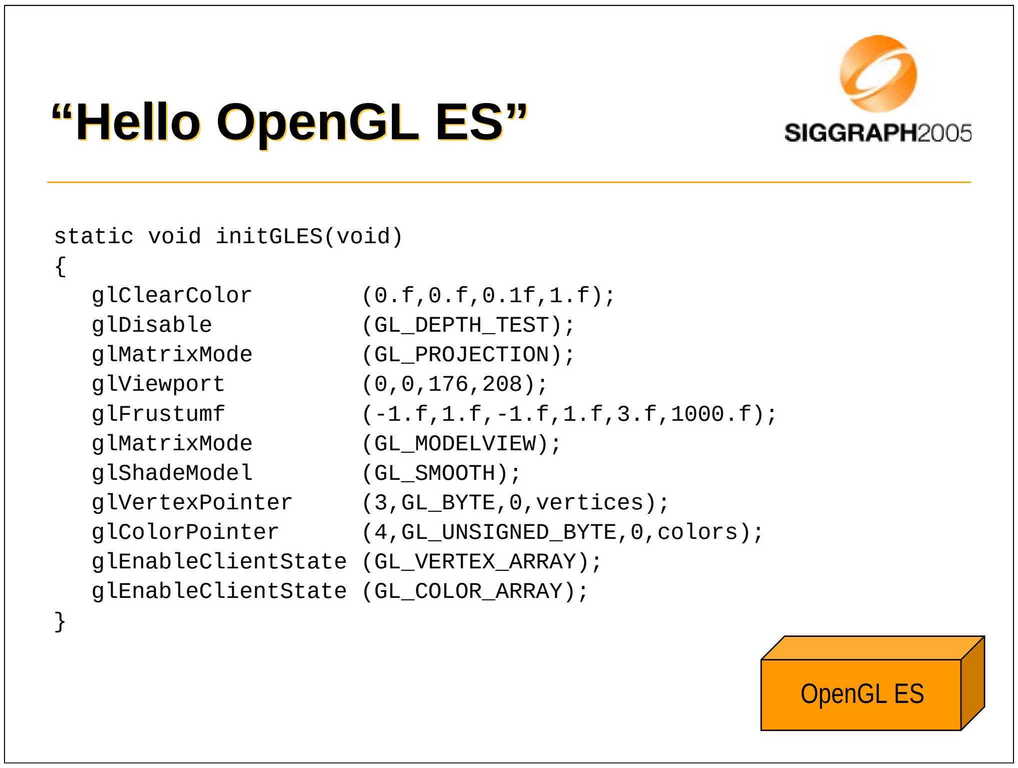
“Hello OpenGL ES”

SIGGRAPH2005

```
static const GLubyte colors[3 * 4] =
{
        255,    0,      0,      255,
        0,      255,    0,      255,
        0,      0,      255,    255
};

/************************************************************
 * Initialize OpenGL ES context and initial OpenGL ES state *
 ************************************************************/

void CSigTriangleGL::Construct(RWindow aWin)
{
   iWin = aWin;
```

OpenGL ES

-Each vertex has different color (full R, full G, full B).



“Hello OpenGL ES”

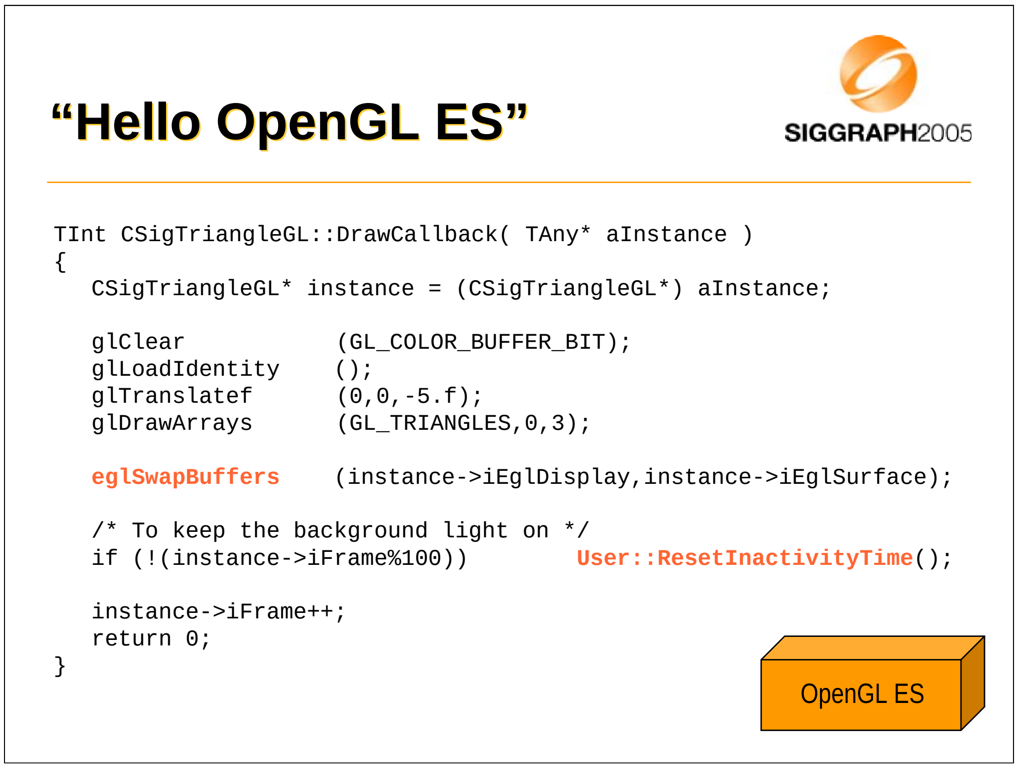
SIGGRAPH2005

```
static void initGLES(void)
{
   glClearColor        (0.f,0.f,0.1f,1.f);
   glDisable           (GL_DEPTH_TEST);
   glMatrixMode        (GL_PROJECTION);
   glViewport          (0,0,176,208);
   glFrustumf          (-1.f,1.f,-1.f,1.f,3.f,1000.f);
   glMatrixMode        (GL_MODELVIEW);
   glShadeModel        (GL_SMOOTH);
   glVertexPointer     (3,GL_BYTE,0,vertices);
   glColorPointer      (4,GL_UNSIGNED_BYTE,0,colors);
   glEnableClientState (GL_VERTEX_ARRAY);
   glEnableClientState (GL_COLOR_ARRAY);
}
```

OpenGL ES

-OpenGL ES setup code, sets up a vertex array and a color array.

-Resolution is set absolutely here (176x208 is typical Series 60 resolution), in real code, always check the resolution from system.



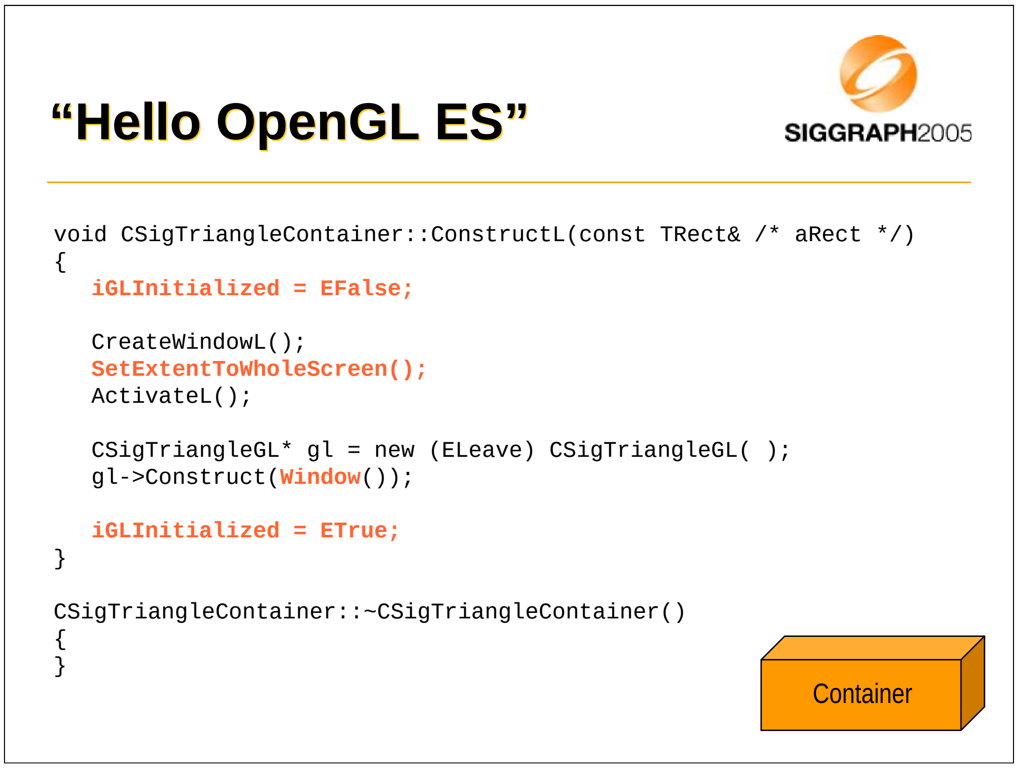
“Hello OpenGL ES”

SIGGRAPH2005

```
TInt CSigTriangleGL::DrawCallback( TAny* aInstance )
{
   CSigTriangleGL* instance = (CSigTriangleGL*) aInstance;

   glClear           (GL_COLOR_BUFFER_BIT);
   glLoadIdentity    ();
   glTranslatef      (0,0,-5.f);
   glDrawArrays      (GL_TRIANGLES,0,3);

   eglSwapBuffers    (instance->iEglDisplay,instance->iEglSurface);

   /* To keep the background light on */
   if (!(instance->iFrame%100))        User::ResetInactivityTime();

   instance->iFrame++;
   return 0;
}
```

OpenGL ES

- This is the render callback. We just clear the color buffer, translate camera a bit and draw a triangle.
- Code keeps a running frame counter. Every once in a while call is made to User::ResetInactivityTime( ) to reset the inactivity counters (to avoid dimming of display backlight).



“Hello OpenGL ES”

SIGGRAPH2005

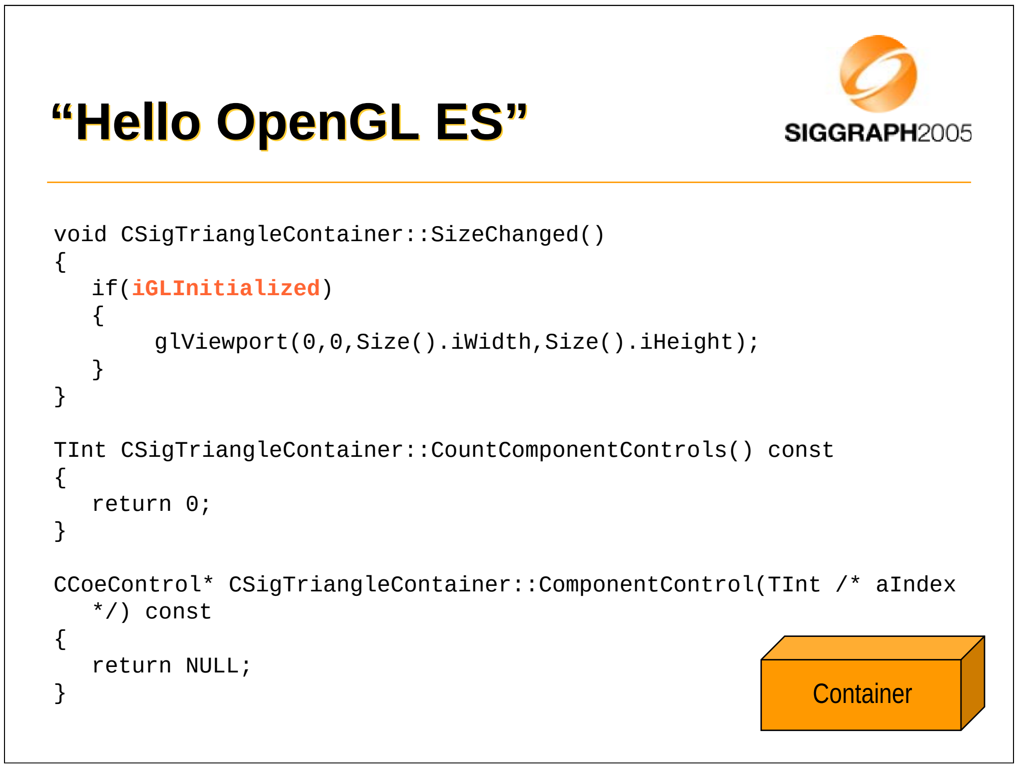
```
void CSigTriangleContainer::ConstructL(const TRect& /* aRect */)
{
   iGLInitialized = EFalse;

   CreateWindowL();
   SetExtentToWholeScreen();
   ActivateL();

   CSigTriangleGL* gl = new (ELeave) CSigTriangleGL( );
   gl->Construct(Window());

   iGLInitialized = ETrue;
}

CSigTriangleContainer::~CSigTriangleContainer()
{
}
```

Container

-ConstructL( ) will be called by the app framework to initialize the View. iGLInitialized is used to block GL calls before actual initialization is done (window operations may cause calls to SizeChanged function).

-We set the extent to fill the whole screen and call the constructor for the GL part of the application. We give in to that constructor a Symbian window class (RWindow) that we get from the Window( ) function.

-After the constructor returns, GL is in initialized state.

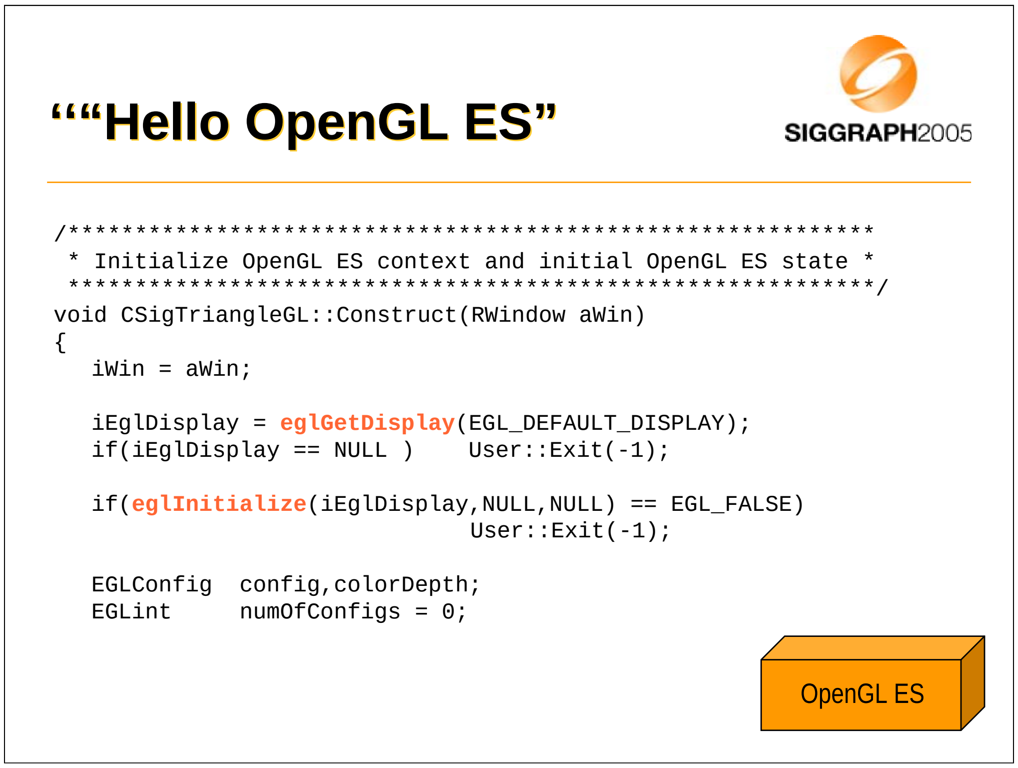


“Hello OpenGL ES”

SIGGRAPH2005

```
void CSigTriangleContainer::SizeChanged()
{
   if(iGLInitialized)
   {
        glViewport(0,0,Size().iWidth,Size().iHeight);
   }
}

TInt CSigTriangleContainer::CountComponentControls() const
{
   return 0;
}

CCoeControl* CSigTriangleContainer::ComponentControl(TInt /* aIndex
   */) const
{
   return NULL;
}
```

-SizeChanged( ) will get called when the application window changes size. If GL is not initialized we don’t change the viewport here (if context is not valid, calling GL functions may crash the application).



““Hello OpenGL ES”

SIGGRAPH2005

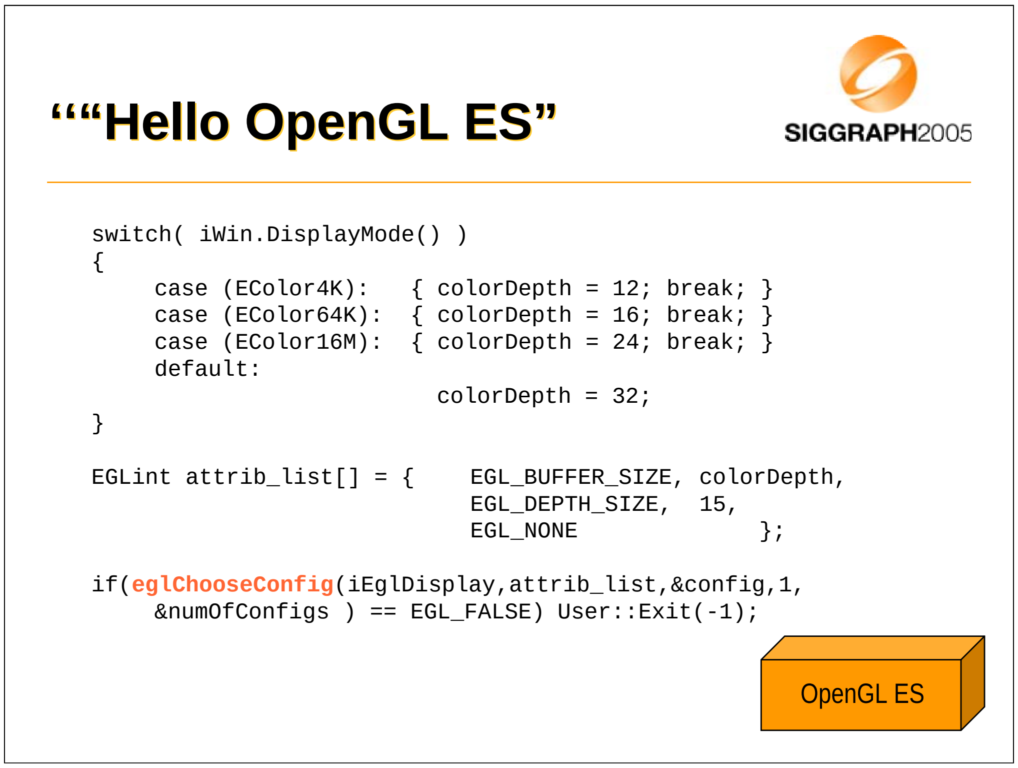
```
/***************************************************************
 * Initialize OpenGL ES context and initial OpenGL ES state *
 ***************************************************************/
void CSigTriangleGL::Construct(RWindow aWin)
{
   iWin = aWin;

   iEglDisplay = eglGetDisplay(EGL_DEFAULT_DISPLAY);
   if(iEglDisplay == NULL )    User::Exit(-1);

   if(eglInitialize(iEglDisplay,NULL,NULL) == EGL_FALSE)
                              User::Exit(-1);

   EGLConfig  config,colorDepth;
   EGLint     numOfConfigs = 0;
```

OpenGL ES

-This is our GL initialization code, called from the View.

-eglGetDisplay(EGL_DEFAULT_DISPLAY) – get the default display to render to

-eglInitialize( )
- initialize EGL on that display



““Hello OpenGL ES”

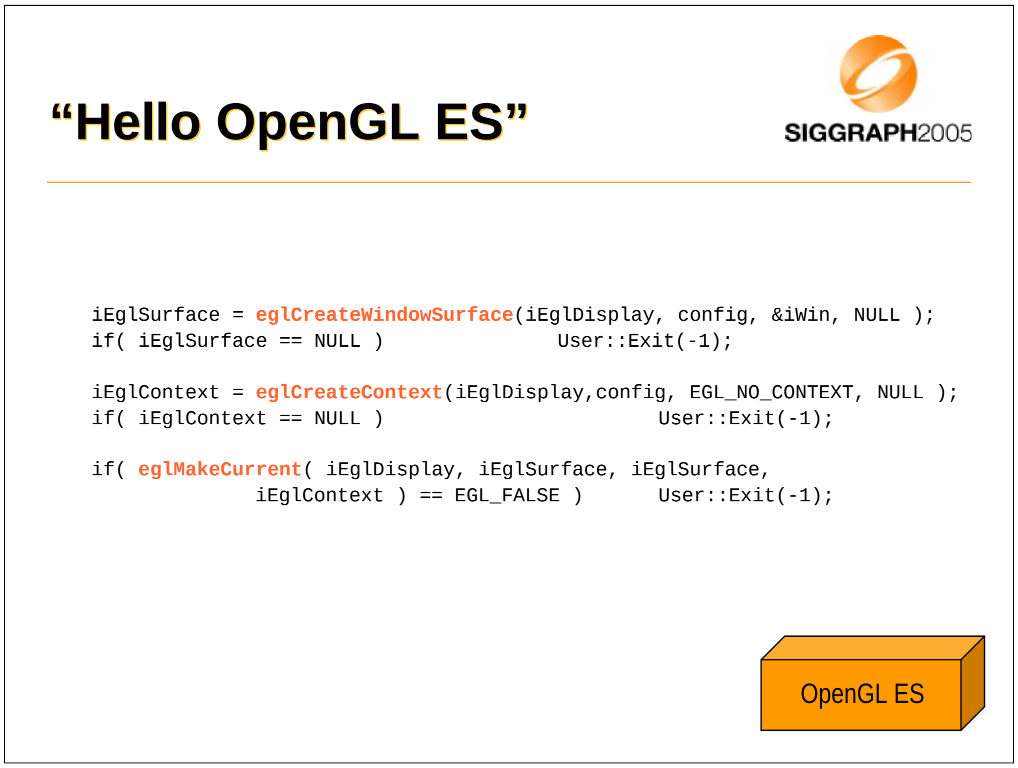
SIGGRAPH2005

```
switch( iWin.DisplayMode() )
{
     case (EColor4K):   { colorDepth = 12; break; }
     case (EColor64K):  { colorDepth = 16; break; }
     case (EColor16M):  { colorDepth = 24; break; }
     default:
                          colorDepth = 32;
}

EGLint attrib_list[] = {    EGL_BUFFER_SIZE, colorDepth,
                            EGL_DEPTH_SIZE,   15,
                            EGL_NONE               };

if(eglChooseConfig(iEglDisplay,attrib_list,&config,1,
     &numOfConfigs ) == EGL_FALSE) User::Exit(-1);
```

OpenGL ES

-iWin.DisplayMode( ) - find out the display mode of the window (match config with that)

-eglChooseConfig( ) - choose the best matching config (see EGL spec for selection criteria)



“Hello OpenGL ES”

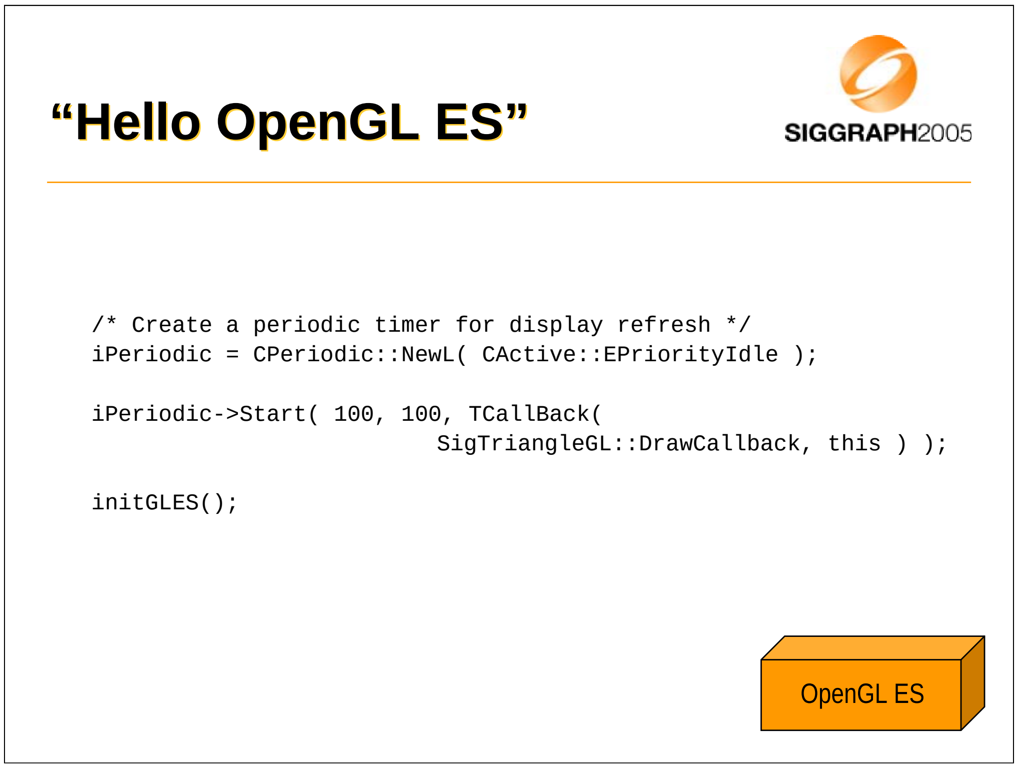
SIGGRAPH2005

```
iEglSurface = eglCreateWindowSurface(iEglDisplay, config, &iWin, NULL );
if( iEglSurface == NULL )              User::Exit(-1);

iEglContext = eglCreateContext(iEglDisplay,config, EGL_NO_CONTEXT, NULL );
if( iEglContext == NULL )                      User::Exit(-1);

if( eglMakeCurrent( iEglDisplay, iEglSurface, iEglSurface,
              iEglContext ) == EGL_FALSE )      User::Exit(-1);
```

-eglCreateWindowSurface( ) - create a window surface for rendering

-eglCreateContext( ) - create a rendering context (multiple contexts may be used, but not at the same time)

-eglMakeCurrent( ) - make surface current and context current to the display and the thread



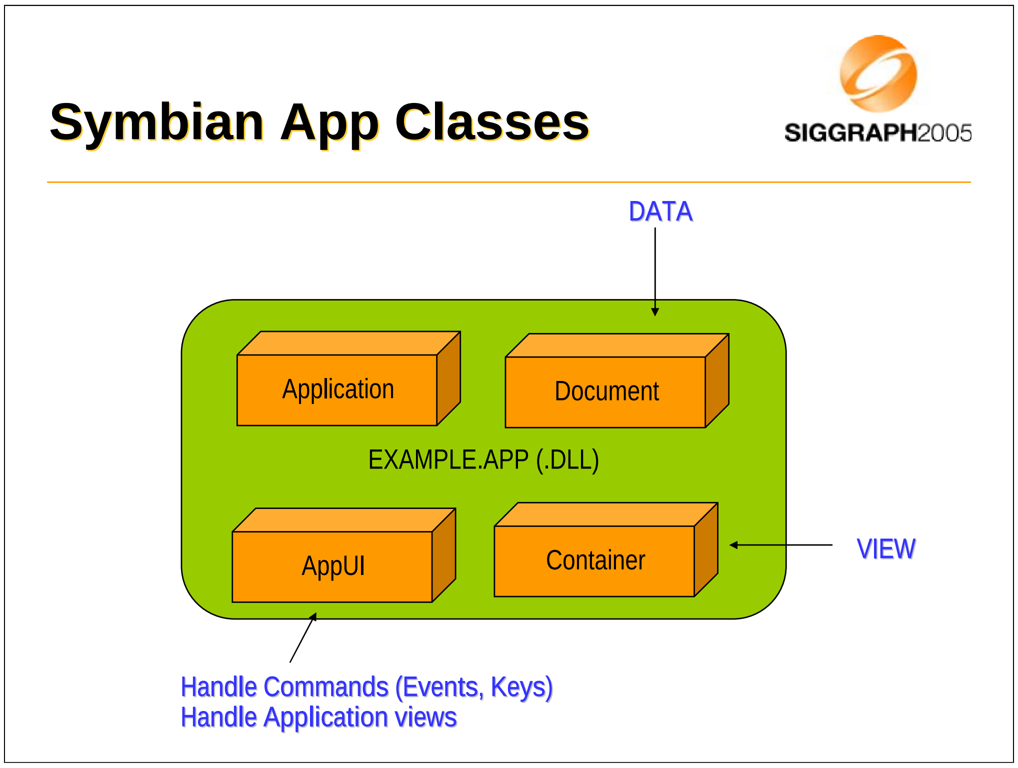
“Hello OpenGL ES”

SIGGRAPH2005

```
/* Create a periodic timer for display refresh */
iPeriodic = CPeriodic::NewL( CActive::EPriorityIdle );

iPeriodic->Start( 100, 100, TCallBack(
                          SigTriangleGL::DrawCallback, this ) );

initGLES();
```

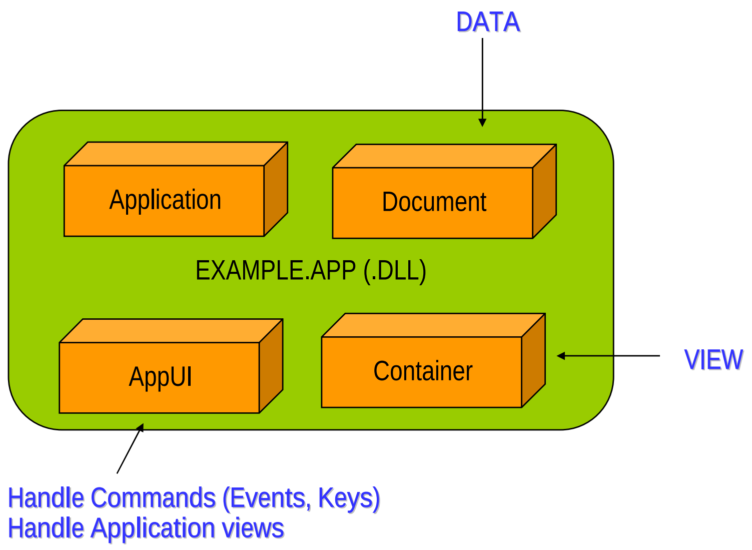
-Cperiodic::NewL( ) - create a Symbian Active Object (form of co-operative multi-tasking) for providing a timer callback

-initGLES( ) - call the GL initialization part shown before

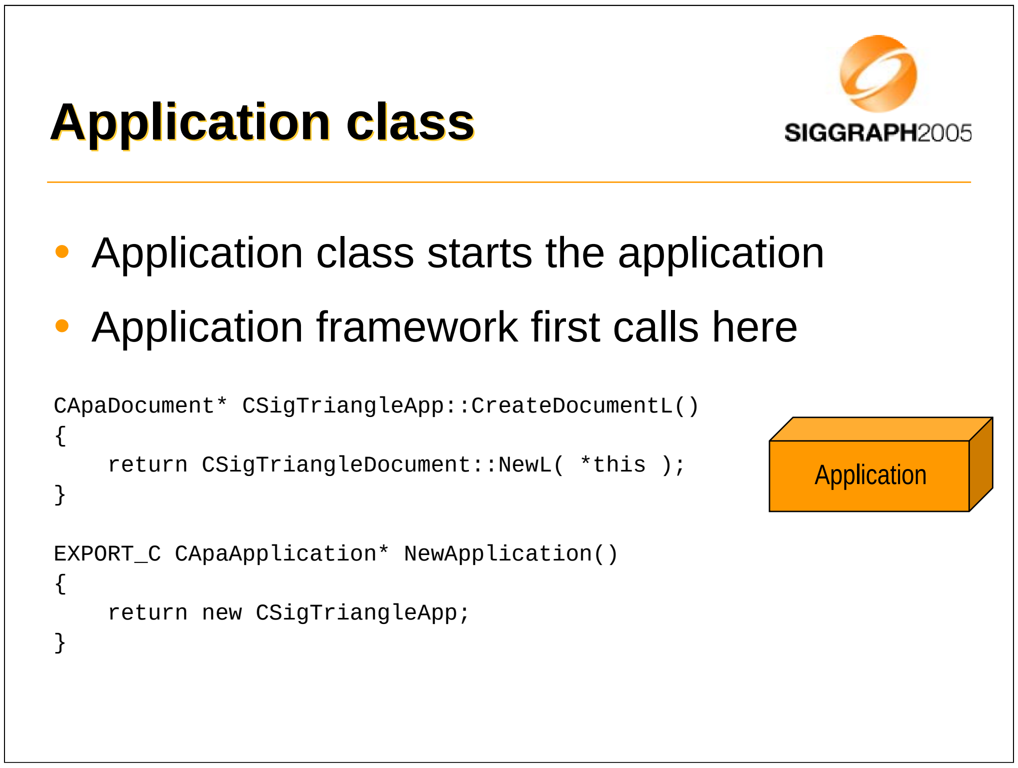


Symbian App Classes

SIGGRAPH2005



-Symbian UI framework follows Model-View-Controller model.

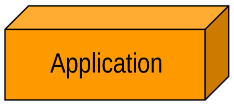


Application class

SIGGRAPH2005

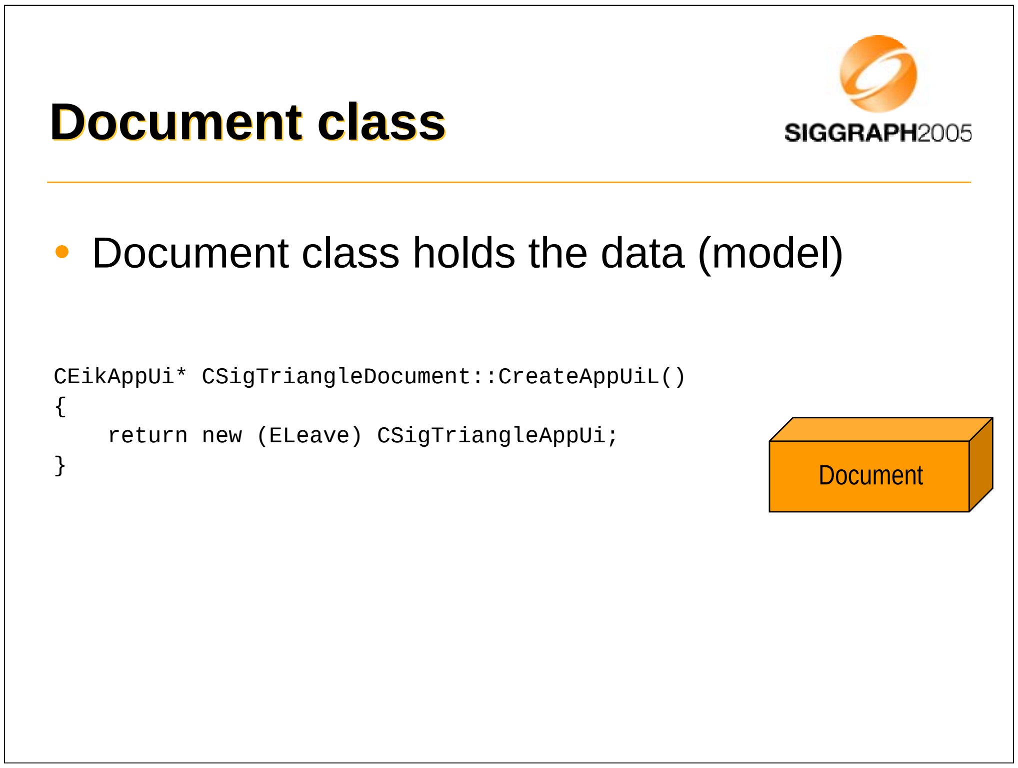
- Application class starts the application
- Application framework first calls here

```
CApaDocument* CSigTriangleApp::CreateDocumentL()
{
    return CSigTriangleDocument::NewL( *this );
}

EXPORT_C CApaApplication* NewApplication()
{
    return new CSigTriangleApp;
}
```



-NewApplication( ) is called by the application framework to create the application class.

-CreateDocumentL( ) is called by application framework to create the document class.



Document class

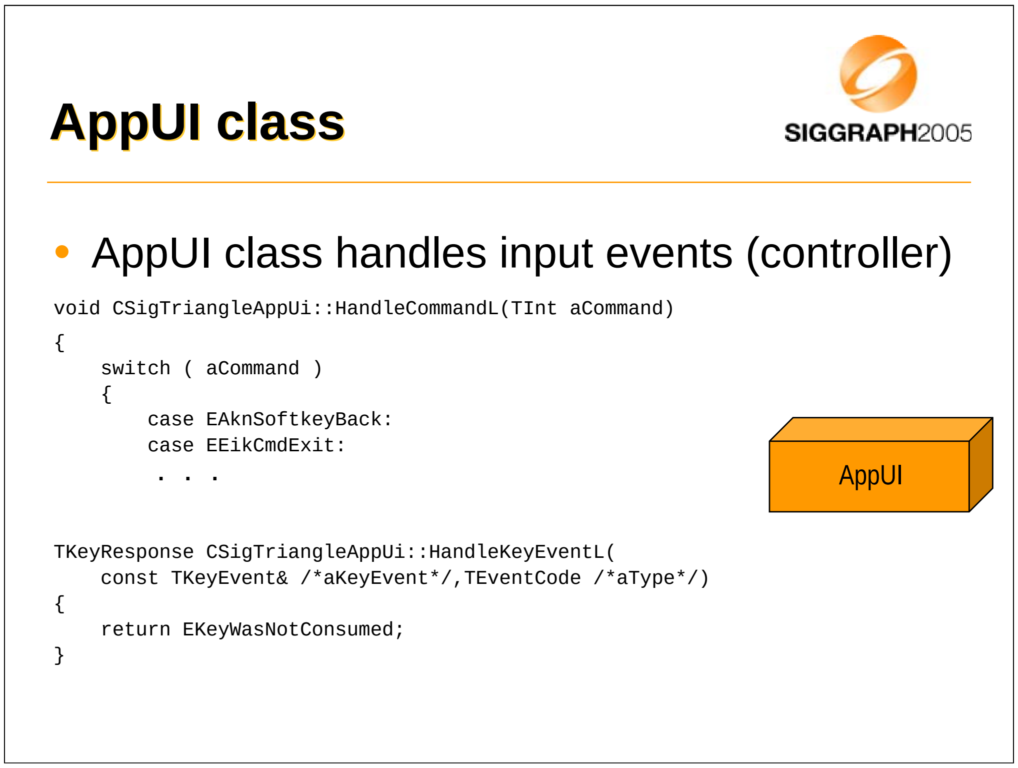
SIGGRAPH2005

- Document class holds the data (model)

```
CEikAppUi* CSigTriangleDocument::CreateAppUiL()
{
    return new (ELeave) CSigTriangleAppUi;
}
```

-CreateAppUIL( ) is called by application framework to create the application UI class (controller).

-In this example we just have the const data, so Document is not really used.

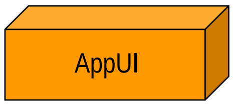


AppUI class

SIGGRAPH2005

- AppUI class handles input events (controller)

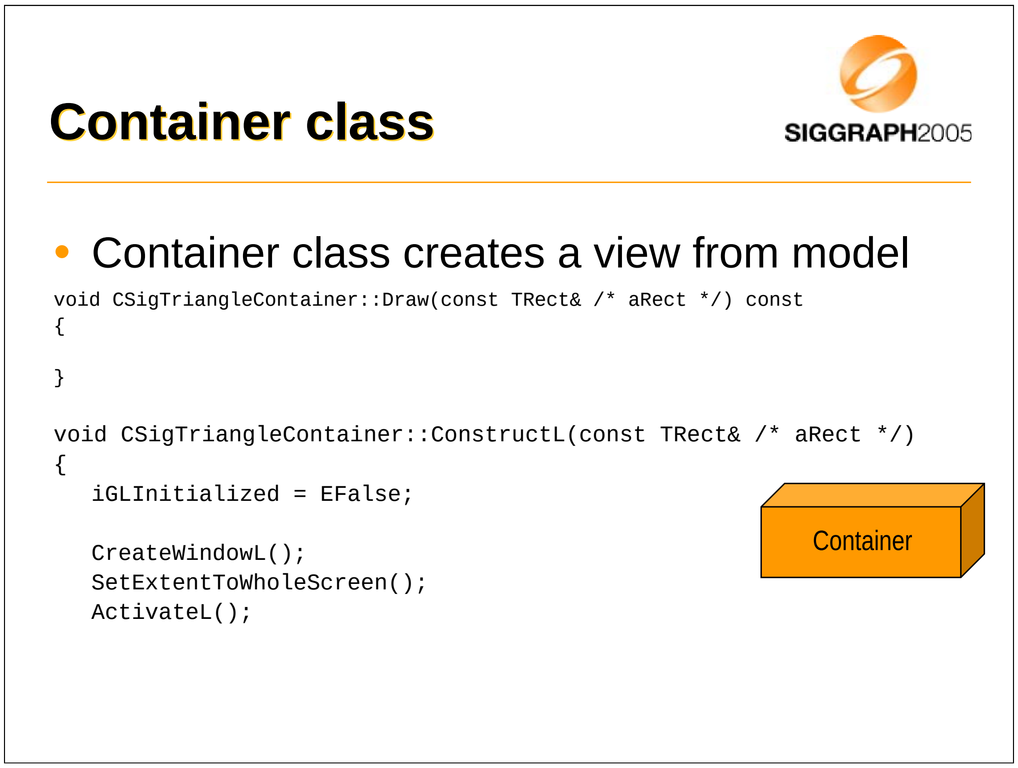
```
void CSigTriangleAppUi::HandleCommandL(TInt aCommand)
{
    switch ( aCommand )
    {
        case EAknSoftkeyBack:
        case EEikCmdExit:
         . . .


TKeyResponse CSigTriangleAppUi::HandleKeyEventL(
    const TKeyEvent& /*aKeyEvent*/,TEventCode /*aType*/)
{
    return EKeyWasNotConsumed;
}
```



-AppUI creates the container class and returns it back to the application framework.

-HandleCommandL( ) is called when input events are triggered. Here exit command is handled.

-HandleKeyEventL( ) is called when keyboard events are triggered. Example does not take keyboard input.

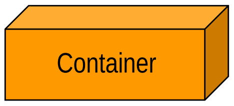


Container class

SIGGRAPH2005

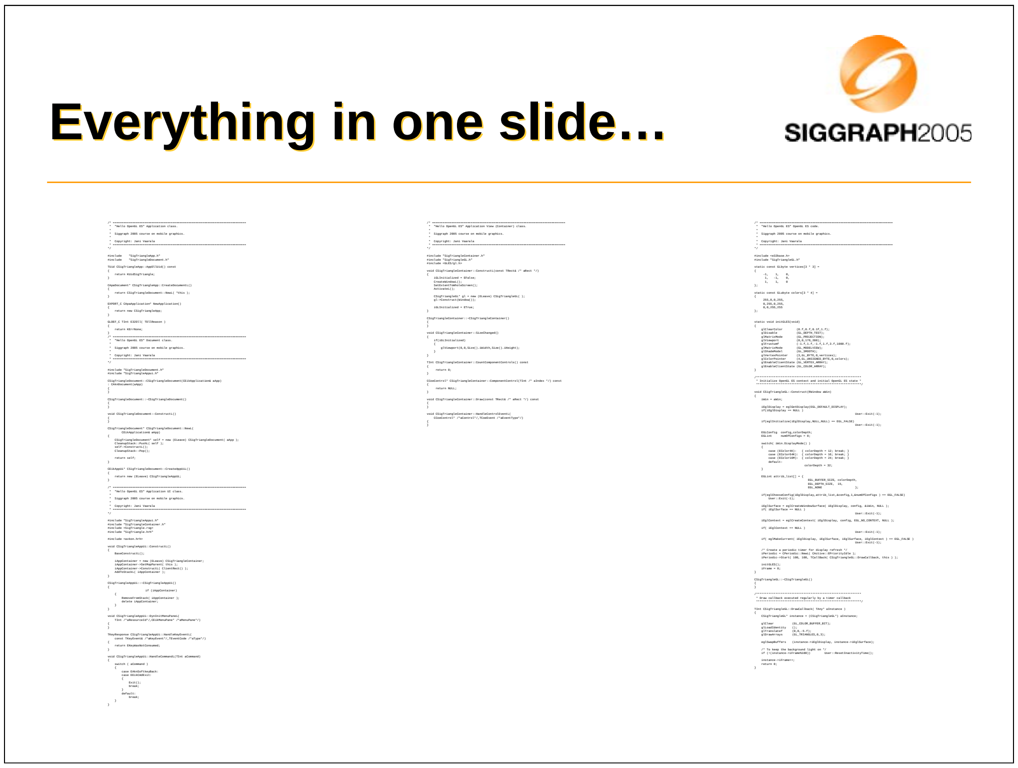
- Container class creates a view from model

```
void CSigTriangleContainer::Draw(const TRect& /* aRect */) const
{

}

void CSigTriangleContainer::ConstructL(const TRect& /* aRect */)
{
   iGLInitialized = EFalse;

   CreateWindowL();
   SetExtentToWholeScreen();
   ActivateL();
```



-Draw( ) is called by the UI framework when the container needs updating (e.g., some graphics from this container was revealed from behind a window). It is not however used with OpenGL ES, as it’s drawing is handled through EGL and not the native Symbian graphics event framework. Actually, GL applications shouldn’t do any drawing from the Draw function of a container.

-ConstructL( ) is called by the AppUI to create the Container.



Everything in one slide…

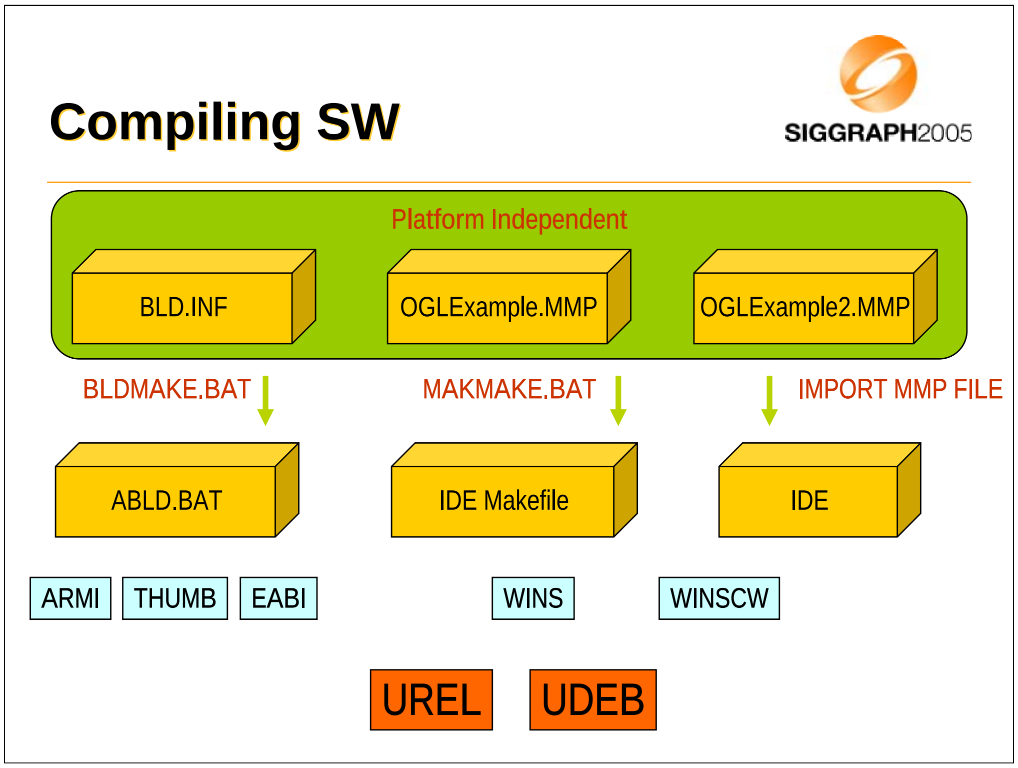
SIGGRAPH2005

```
/* ========================================================================
 *  "Hello OpenGL ES" Application class.
 *
 *  Siggraph 2005 course on mobile graphics.
 *
 *  Copyright: Jani Vaarala
 * ========================================================================
*/

#include    "SigTriangleApp.h"
#include    "SigTriangleDocument.h"

TUid CSigTriangleApp::AppDllUid() const
{
    return KUidSigTriangle;
}

CApaDocument* CSigTriangleApp::CreateDocumentL()
{
    return CSigTriangleDocument::NewL( *this );
}

EXPORT_C CApaApplication* NewApplication()
{
    return new CSigTriangleApp;
}

GLDEF_C TInt E32Dll( TDllReason )
{
    return KErrNone;
}
/* ========================================================================
 *  "Hello OpenGL ES" Document class.
 *
 *  Siggraph 2005 course on mobile graphics.
 *
 *  Copyright: Jani Vaarala
 * ========================================================================
*/

#include "SigTriangleDocument.h"
#include "SigTriangleAppui.h"

CSigTriangleDocument::CSigTriangleDocument(CEikApplication& aApp)
: CAknDocument(aApp)
{
}

CSigTriangleDocument::~CSigTriangleDocument()
{
}

void CSigTriangleDocument::ConstructL()
{
}

CSigTriangleDocument* CSigTriangleDocument::NewL(
        CEikApplication& aApp)
{
    CSigTriangleDocument* self = new (ELeave) CSigTriangleDocument( aApp );
    CleanupStack::PushL( self );
    self->ConstructL();
    CleanupStack::Pop();

    return self;
}

CEikAppUi* CSigTriangleDocument::CreateAppUiL()
{
    return new (ELeave) CSigTriangleAppUi;
}

/* ========================================================================
 *  "Hello OpenGL ES" Application UI class.
 *
 *  Siggraph 2005 course on mobile graphics.
 *
 *  Copyright: Jani Vaarala
 * ========================================================================
*/

#include "SigTriangleAppui.h"
#include "SigTriangleContainer.h"
#include <SigTriangle.rsg>
#include "SigTriangle.hrh"

#include <avkon.hrh>

void CSigTriangleAppUi::ConstructL()
{
    BaseConstructL();

    iAppContainer = new (ELeave) CSigTriangleContainer;
    iAppContainer->SetMopParent( this );
    iAppContainer->ConstructL( ClientRect() );
    AddToStackL( iAppContainer );
}

CSigTriangleAppUi::~CSigTriangleAppUi()
{
                      if (iAppContainer)
    {
        RemoveFromStack( iAppContainer );
        delete iAppContainer;
    }
}

void CSigTriangleAppUi::DynInitMenuPaneL(
    TInt /*aResourceId*/,CEikMenuPane* /*aMenuPane*/)
{
}

TKeyResponse CSigTriangleAppUi::HandleKeyEventL(
    const TKeyEvent& /*aKeyEvent*/,TEventCode /*aType*/)
{
    return EKeyWasNotConsumed;
}

void CSigTriangleAppUi::HandleCommandL(TInt aCommand)
{
    switch ( aCommand )
    {
        case EAknSoftkeyBack:
        case EEikCmdExit:
        {
            Exit();
            break;
        }
        default:
            break;
    }
}
```

```
/* ========================================================================
 *  "Hello OpenGL ES" Application View (Container) class.
 *
 *  Siggraph 2005 course on mobile graphics.
 *
 *  Copyright: Jani Vaarala
 * ========================================================================
*/

#include "SigTriangleContainer.h"
#include "SigTriangleGL.h"
#include <GLES/gl.h>

void CSigTriangleContainer::ConstructL(const TRect& /* aRect */)
{
    iGLInitialized = EFalse;
    CreateWindowL();
    SetExtentToWholeScreen();
    ActivateL();

    CSigTriangleGL* gl = new (ELeave) CSigTriangleGL( );
    gl->Construct(Window());

    iGLInitialized = ETrue;
}

CSigTriangleContainer::~CSigTriangleContainer()
{
}

void CSigTriangleContainer::SizeChanged()
{
    if(iGLInitialized)
    {
        glViewport(0,0,Size().iWidth,Size().iHeight);
    }
}

TInt CSigTriangleContainer::CountComponentControls() const
{
     return 0;
}

CCoeControl* CSigTriangleContainer::ComponentControl(TInt /* aIndex */) const
{
     return NULL;
}

void CSigTriangleContainer::Draw(const TRect& /* aRect */) const
{
}

void CSigTriangleContainer::HandleControlEventL(
    CCoeControl* /*aControl*/,TCoeEvent /*aEventType*/)
{
}
```

```
/* ========================================================================
 *  "Hello OpenGL ES" OpenGL ES code.
 *
 *  Siggraph 2005 course on mobile graphics.
 *
 *  Copyright: Jani Vaarala
 * ========================================================================
*/

#include <e32base.h>
#include "SigTriangleGL.h"

static const GLbyte vertices[3 * 3] =
{
     -1,    1,    0,
      1,   -1,    0,
      1,    1,    0
};

static const GLubyte colors[3 * 4] =
{
     255,0,0,255,
     0,255,0,255,
     0,0,255,255
};


static void initGLES(void)
{
    glClearColor        (0.f,0.f,0.1f,1.f);
    glDisable           (GL_DEPTH_TEST);
    glMatrixMode        (GL_PROJECTION);
    glViewport          (0,0,176,208);
    glFrustumf          (-1.f,1.f,-1.f,1.f,3.f,1000.f);
    glMatrixMode        (GL_MODELVIEW);
    glShadeModel        (GL_SMOOTH);
    glVertexPointer     (3,GL_BYTE,0,vertices);
    glColorPointer      (4,GL_UNSIGNED_BYTE,0,colors);
    glEnableClientState (GL_VERTEX_ARRAY);
    glEnableClientState (GL_COLOR_ARRAY);
}

/*********************************************************
 * Initialize OpenGL ES context and initial OpenGL ES state *
 *********************************************************/

void CSigTriangleGL::Construct(RWindow aWin)
{
    iWin = aWin;

    iEglDisplay = eglGetDisplay(EGL_DEFAULT_DISPLAY);
    if(iEglDisplay == NULL )
                                                  User::Exit(-1);

    if(eglInitialize(iEglDisplay,NULL,NULL) == EGL_FALSE)
                                                  User::Exit(-1);

    EGLConfig  config,colorDepth;
    EGLint     numOfConfigs = 0;

    switch( iWin.DisplayMode() )
    {
        case (EColor4K):   { colorDepth = 12; break; }
        case (EColor64K):  { colorDepth = 16; break; }
        case (EColor16M):  { colorDepth = 24; break; }
        default:
                              colorDepth = 32;
    }

    EGLint attrib_list[] = {
                                EGL_BUFFER_SIZE, colorDepth,
                                EGL_DEPTH_SIZE,  15,
                                EGL_NONE                   };

    if(eglChooseConfig(iEglDisplay,attrib_list,&config,1,&numOfConfigs ) == EGL_FALSE)
        User::Exit(-1);

    iEglSurface = eglCreateWindowSurface( iEglDisplay, config, &iWin, NULL );
    if( iEglSurface == NULL )
                                                  User::Exit(-1);

    iEglContext = eglCreateContext( iEglDisplay, config, EGL_NO_CONTEXT, NULL );

    if( iEglContext == NULL )
                                                  User::Exit(-1);

    if( eglMakeCurrent( iEglDisplay, iEglSurface, iEglSurface, iEglContext ) == EGL_FALSE )
                                                  User::Exit(-1);

    /* Create a periodic timer for display refresh */
    iPeriodic = CPeriodic::NewL( CActive::EPriorityIdle );
    iPeriodic->Start( 100, 100, TCallBack( CSigTriangleGL::DrawCallback, this ) );

    initGLES();
    iFrame = 0;
}

CSigTriangleGL::~CSigTriangleGL()
{
}

/*********************************************************
 * Draw callback executed regularly by a timer callback
 *********************************************************/

TInt CSigTriangleGL::DrawCallback( TAny* aInstance )
{
    CSigTriangleGL* instance = (CSigTriangleGL*) aInstance;

    glClear           (GL_COLOR_BUFFER_BIT);
    glLoadIdentity    ();
    glTranslatef      (0,0,-5.f);
    glDrawArrays      (GL_TRIANGLES,0,3);

    eglSwapBuffers    (instance->iEglDisplay, instance->iEglSurface);

    /* To keep the background light on */
    if (!(instance->iFrame%100))        User::ResetInactivityTime();

    instance->iFrame++;
    return 0;
}
```

-This slide includes all of the CPP files for the example. All of the code is also in the course notes with build instructions etc.



Compiling SW

-BLD.INF contains high level instructions for building, it includes links to multiple platform independent project files (.MMP).

-.MMP files are platform independent “Makefiles” which can be conterted to platform specific makefiles (e.g., Visual Studio, Metrowerks).

-ABLD.BAT can be created wth BLDMAKE.BAT. ABLD.BAT is used to compile for ARM target (“abld.bat build thumb urel”).

-MAKMAKE.BAT can be used to create IDE makefile (.DSW, .DSP for VC6 for example).

-Some IDEs supporting Symbian may also support importing directly the project from a MMP file.

-ARMI is the build target for ARM processor in 32-bit opcode mode (bigger code, but faster).

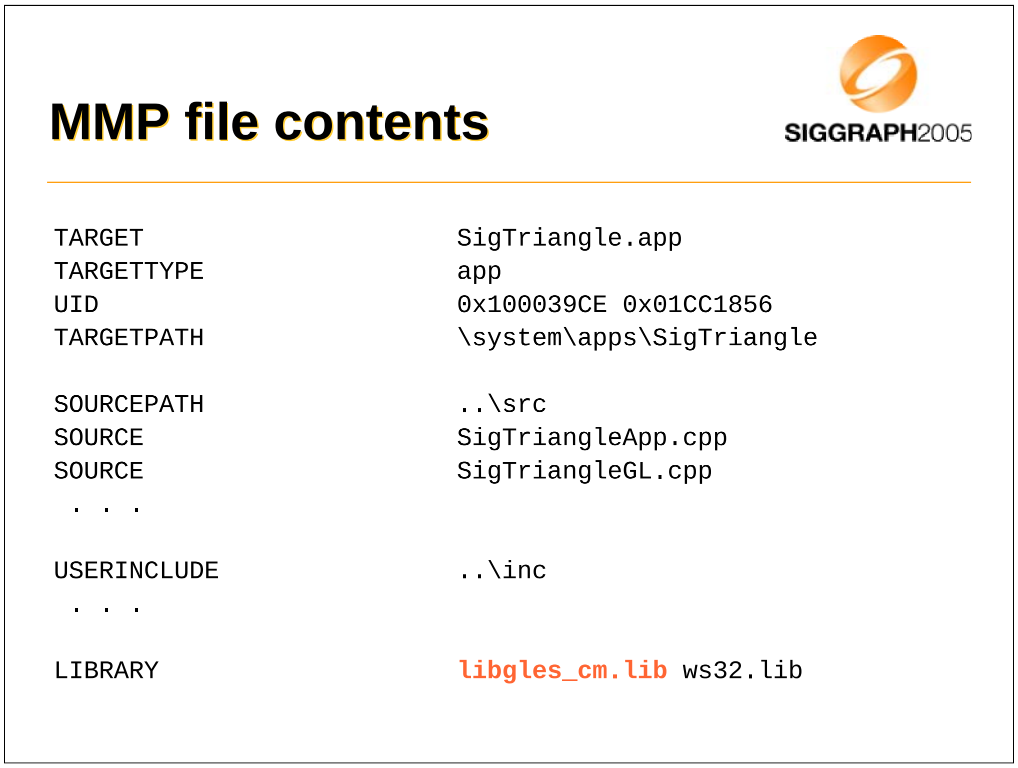
-THUMB is the build target for ARM processor in 16-bit opcode mode (smaller code, but slower).

-WINS is the build target for Windows Emulator.

-WINSCW is the build target for Windows Emulator in CodeWarrior (specify one).

-UREL is the secondary build target for release builds.

-UDEB is the secondary build target for debug builds (specify one).



MMP file contents

SIGGRAPH2005

```
TARGET                      SigTriangle.app
TARGETTYPE                  app
UID                         0x100039CE 0x01CC1856
TARGETPATH                  \system\apps\SigTriangle

SOURCEPATH                  ..\src
SOURCE                      SigTriangleApp.cpp
SOURCE                      SigTriangleGL.cpp
 . . .

USERINCLUDE                 ..\inc
 . . .

LIBRARY                     libgles_cm.lib ws32.lib
```

-MMP file format is explained in the SDK documentation in depth.

-TARGET - specifies the name of the application

-TARGETTYPE - specifies the type of the application

-UID - specifies the Unique IDentifier (each application has it's own, 0x0100 0000 – 0x01ff ffff is the test range)

-SOURCEPATH - specifies a folder (relative or absolute), where source files are stored

-SOURCE - specifies a single source (.C, .CPP, .S)

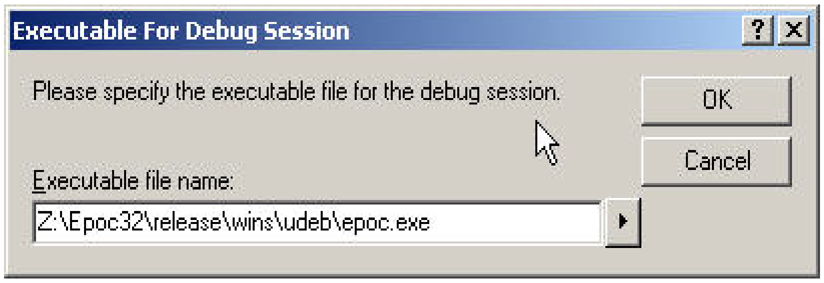
-USERINCLUDE - specifies user include paths

-LIBRARY - specifies what libraries should be linked with the application (LIBGLES_CM.LIB is required for OpenGL ES applications to link)

Compiling in Windows

SIGGRAPH2005

Just press F5 to build'n'run (Visual Studio)



Executable For Debug Session

Please specify the executable file for the debug session.

Executable file name:

Z:\Epoc32\release\wins\udeb\epoc.exe

Step-by-step guide in the course notes:

Sig2005/SigTriangle/readme.txt

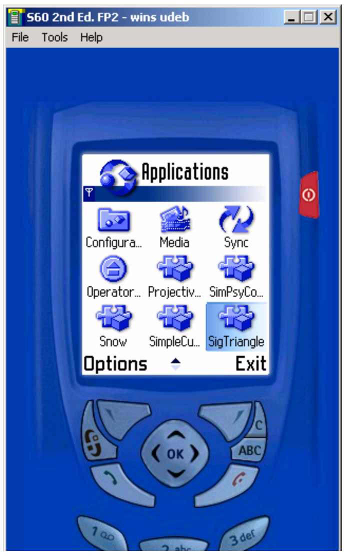
-You can use MAKMAKE.BAT to convert .MMP file to a VC6 .DSW + .DSP file.

-When you open the .DSW file, you can build'n'run the application by hitting F5.

-When following dialog is opened, you should search for right "EPOC.EXE" for the SDK that you are using (typical installation location is C:\Symbian\…). Select EPOC32\RELEASE\WINS\UDEB\EPOC.EXE for VC6 builds (debug version).

Emulator

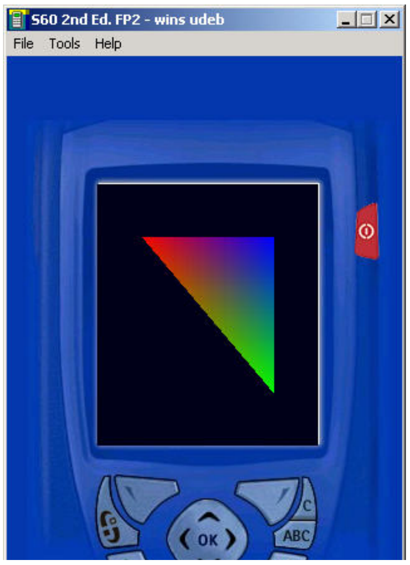
SIGGRAPH2005



-Choose the SigTriangle application from application menu (in emulator) to run our OpenGL ES application.

Emulator

SIGGRAPH2005



-This is what you should get.

Compiling for ARM Target

SIGGRAPH2005

- Target build is done in the group folder:

```
bldmake.bat bldfiles
./abld.bat build thumb urel                    OR
./abld.bat build armi urel
```

- Install file (SigTriangle.sis) is created (sis folder)

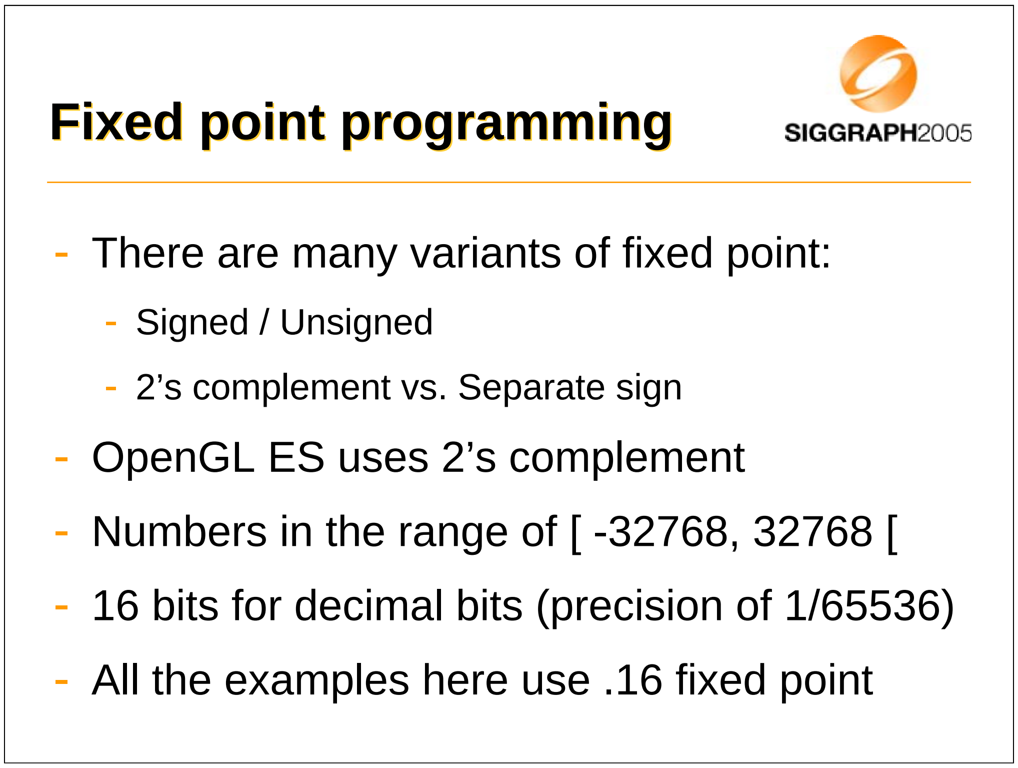
```
makesis.bat SigTriangle.pkg
```

- PKG-file is a list of files to be included in the .sis file
- Installation file can be sent to phone

Fixed point programming

SIGGRAPH2005

- Why to use it?
  - Most mobile handsets don't have a FPU
- Where does it make sense to use it?
  - Where it makes the most difference
  - For per-vertex processing: morphing, skinning, etc.
  - Per vertex data shouldn't be floating point
- OpenGL ES API supports 32-bit FP numbers



Fixed point programming

SIGGRAPH2005

- There are many variants of fixed point:
  - Signed / Unsigned
  - 2’s complement vs. Separate sign
- OpenGL ES uses 2’s complement
- Numbers in the range of [-32768, 32768 [
- 16 bits for decimal bits (precision of 1/65536)
- All the examples here use .16 fixed point

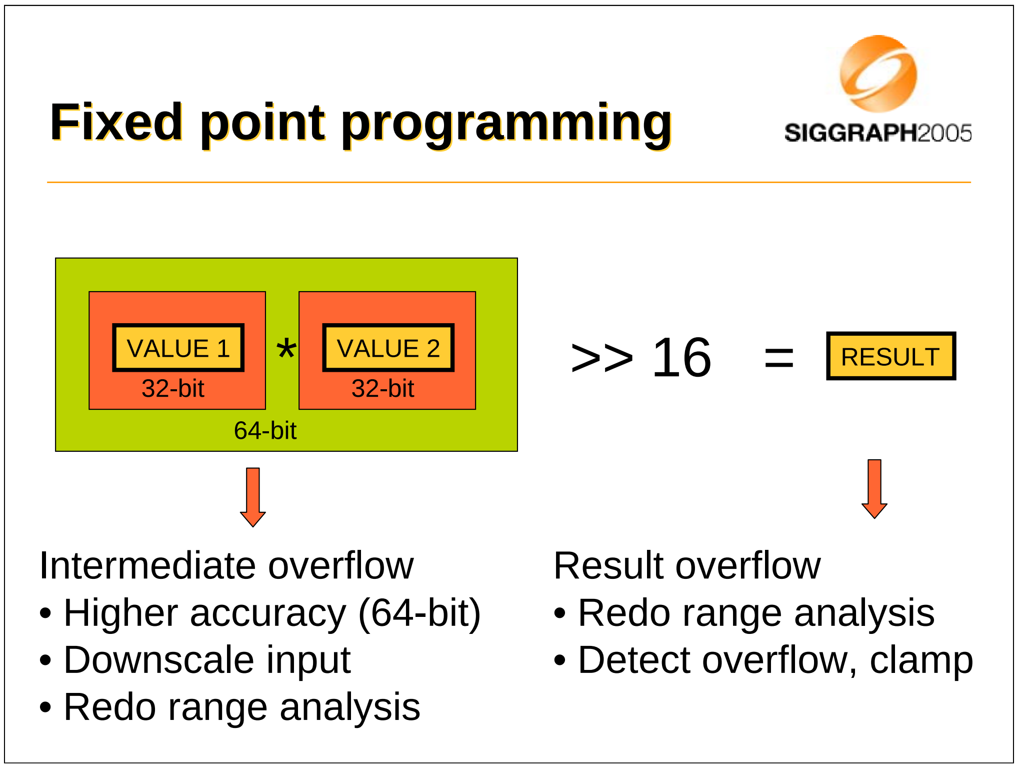
•Fixed point scale is 2^16 (65536, 0x10000).

Fixed point programming

SIGGRAPH2005

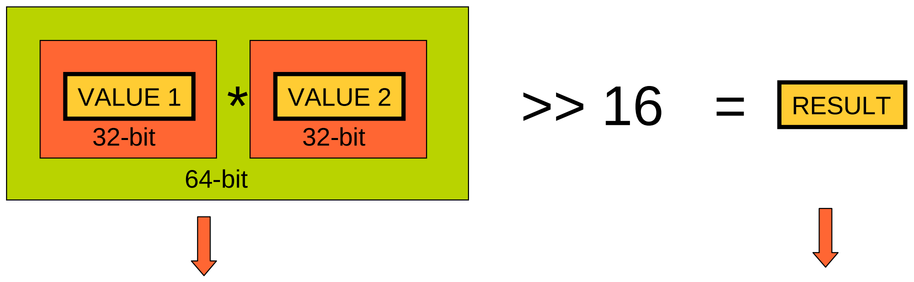
- Examples:

```
0x0001 0000 =  “1.0f”
0x0002 0000 =  “2.0f”
0x0010 0000 =  “16.0f”
0x0000 0001 =  1/0x10000(0x10000 = 2^16)
0xffff ffff = -1/0x10000(-0x0000 0001)
```



Fixed point programming

SIGGRAPH2005



Intermediate overflow
- Higher accuracy (64-bit)
- Downscale input
- Redo range analysis

Result overflow
- Redo range analysis
- Detect overflow, clamp

-Multiplying two 32-bit numbers with standard C “int” multiply gives you lower 32 bits from that multiplication.

-Intermediate value may need 64 bits (high 32-bits cannot be ignored in this case).

-This can occur for example if you multiply two fixed point numbers together (also two fixed point scales multiplied together at the same time).

-Solution 1: use 64-bit math for the intermediate, use 64-bit shifter to get the result down.

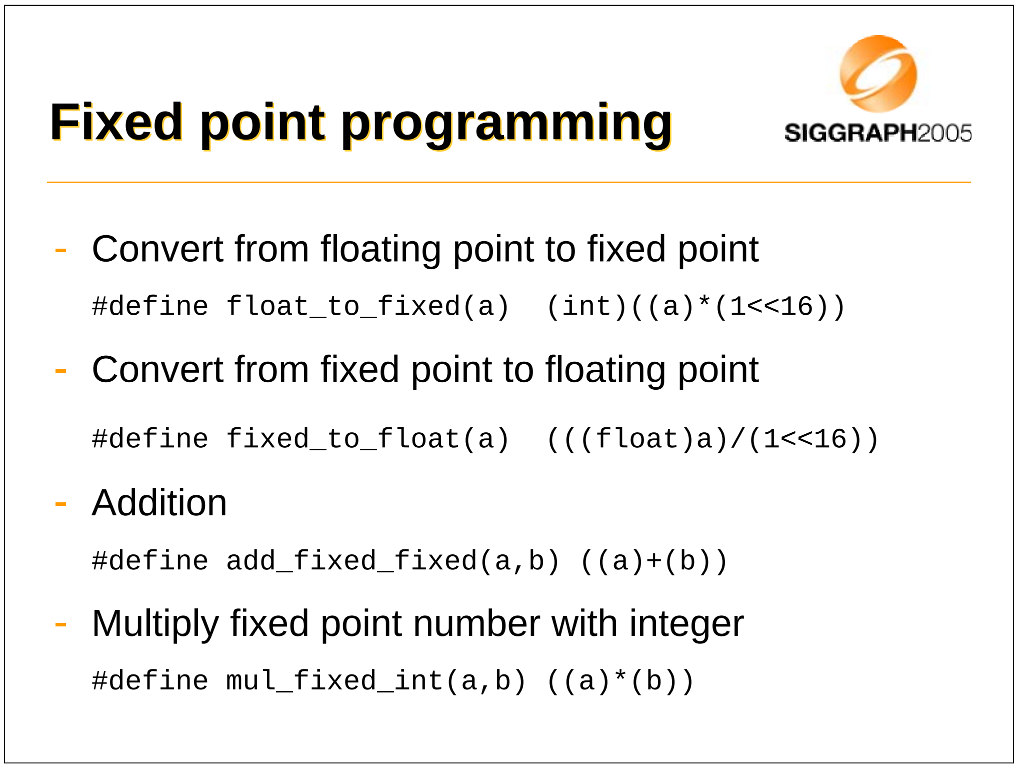
-Solution 2: downscale on the input (just for this operation), for example divide input operands by 2^4, take that into account in result.

-Solution 3: redo the range analysis.

-Also the result may overflow (even if internal precision of 64-bit would be used for intermediate calculation).

-Solution 1: redo the ranges.

-Solution 2: clamp the results (it’s better to clamp than just overflow. Clamping limits the resulting error, with ignored overflow the errors easily become very large).



Fixed point programming

SIGGRAPH2005

- Convert from floating point to fixed point

```
#define float_to_fixed(a)  (int)((a)*(1<<16))
```

- Convert from fixed point to floating point

```
#define fixed_to_float(a)  (((float)a)/(1<<16))
```

- Addition

```
#define add_fixed_fixed(a,b) ((a)+(b))
```

- Multiply fixed point number with integer

```
#define mul_fixed_int(a,b) ((a)*(b))
```

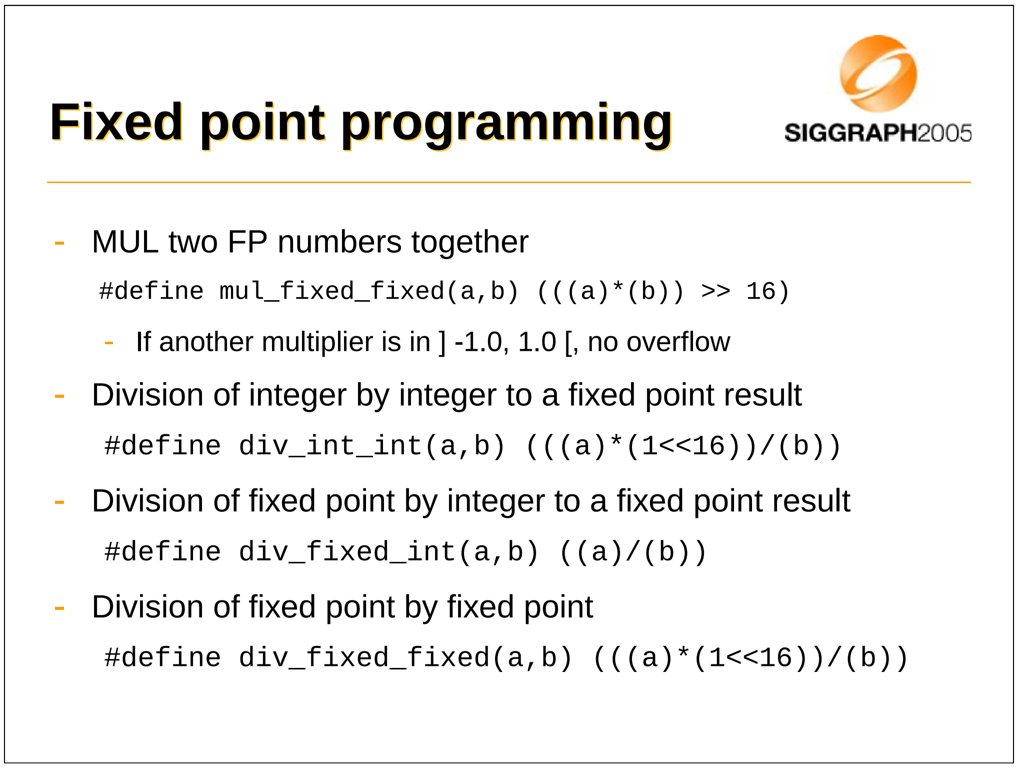
Notes about overflows:

-conversion from float is not possible if input number is not in the right range [-32768, 32768[.

-conversion from fixed reduces accuracy (float has 25 bits for mantissa and sign, whereas fixed point uses 32 bits) E.g., (32767.0 + 1/65536 = 32767.0000152). If accuracy is crucial, convert to double to preserve the result.

-add can overflow by one bit (e.g. by adding 32767.0 + 32767.0), result overflows. If you use add for averaging, you may also divide both input numbers by two and then just add them together. This doesn’t overflow in the intermediate calculations, but it loses some accuracy (lowest bit from both inputs).

-multiplying fixed point number with integer can overflow if result does not fit into 32-bit, examples: 32767.0 * 2 or 2.0 * 16384.



Fixed point programming

SIGGRAPH2005

- MUL two FP numbers together

```
#define mul_fixed_fixed(a,b) (((a)*(b)) >> 16)
```

  - If another multiplier is in] -1.0, 1.0 [, no overflow
- Division of integer by integer to a fixed point result

```
#define div_int_int(a,b) (((a)*(1<<16))/(b))
```

- Division of fixed point by integer to a fixed point result

```
#define div_fixed_int(a,b) ((a)/(b))
```

- Division of fixed point by fixed point

```
#define div_fixed_fixed(a,b) (((a)*(1<<16))/(b))
```

Notes about overflows:

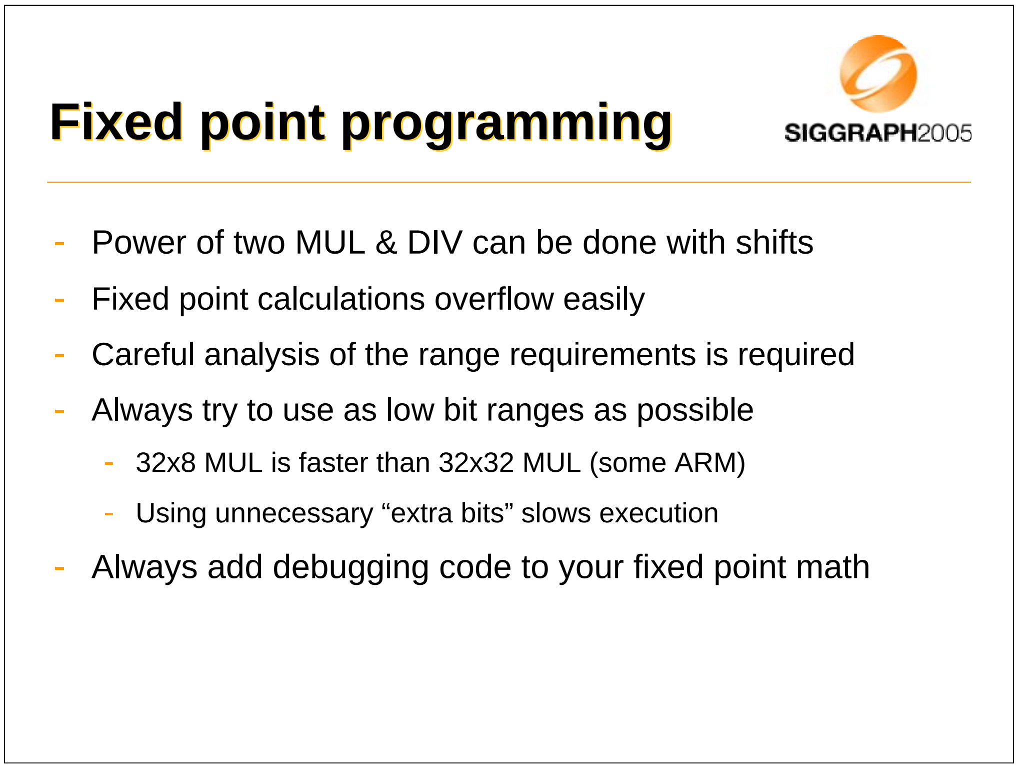
-MUL two FP numbers together can overflow in the intermediate calculation (a*b), an example: 2.0 * 2.0 (intermediate is: 2*2*1^16*1^16, requires 35 bits intermediate incl. sign bit).

-If the operation can be done with 32x32 -> 64-bit multiply, followed by 16-bit shift, overflow only occurs if the result after the shift does not fit into 32-bit (in that case either the range has to be changed or the destination should be carried over in 64-bit number).

-Division of integer by integer can overflow if a is not in the range [-32768,32767] (because multiplication of a by (1<<16) does not fit in to 32 bits).

-Division of fixed by integer cannot overflow, but results may become zero.

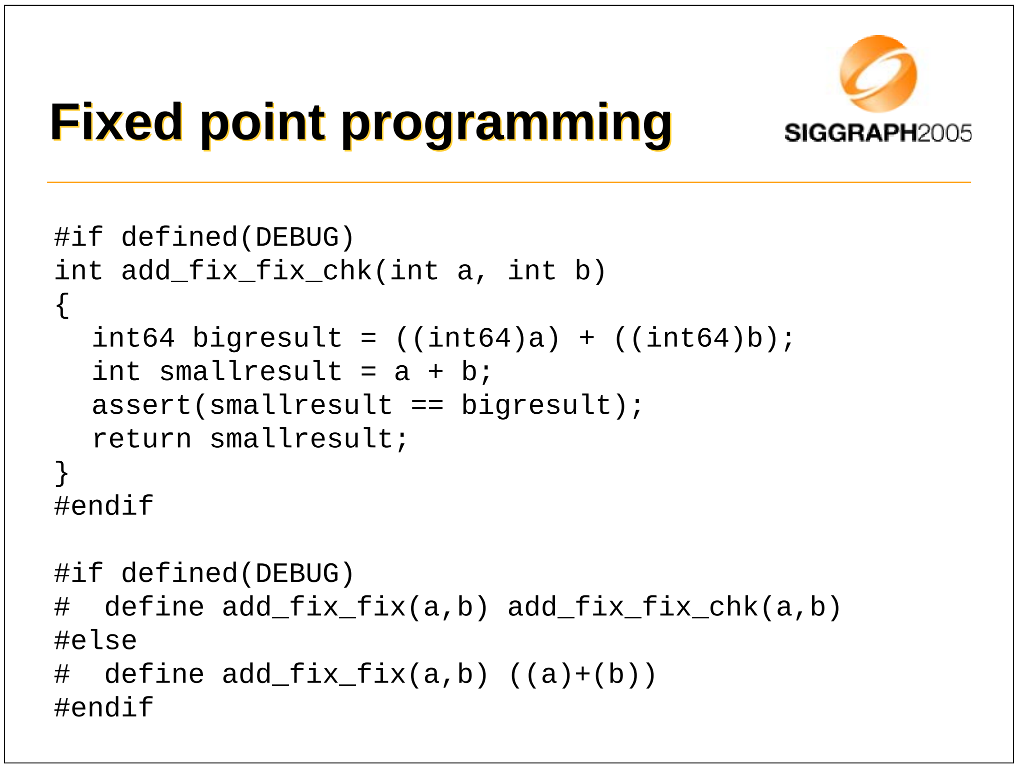
-Division of fixed by fixed may overflow if a is not in range]-1.0, 1.0[, intermediate overflow.



Fixed point programming

SIGGRAPH2005

- Power of two MUL & DIV can be done with shifts
- Fixed point calculations overflow easily
- Careful analysis of the range requirements is required
- Always try to use as low bit ranges as possible
  - 32x8 MUL is faster than 32x32 MUL (some ARM)
  - Using unnecessary “extra bits” slows execution
- Always add debugging code to your fixed point math



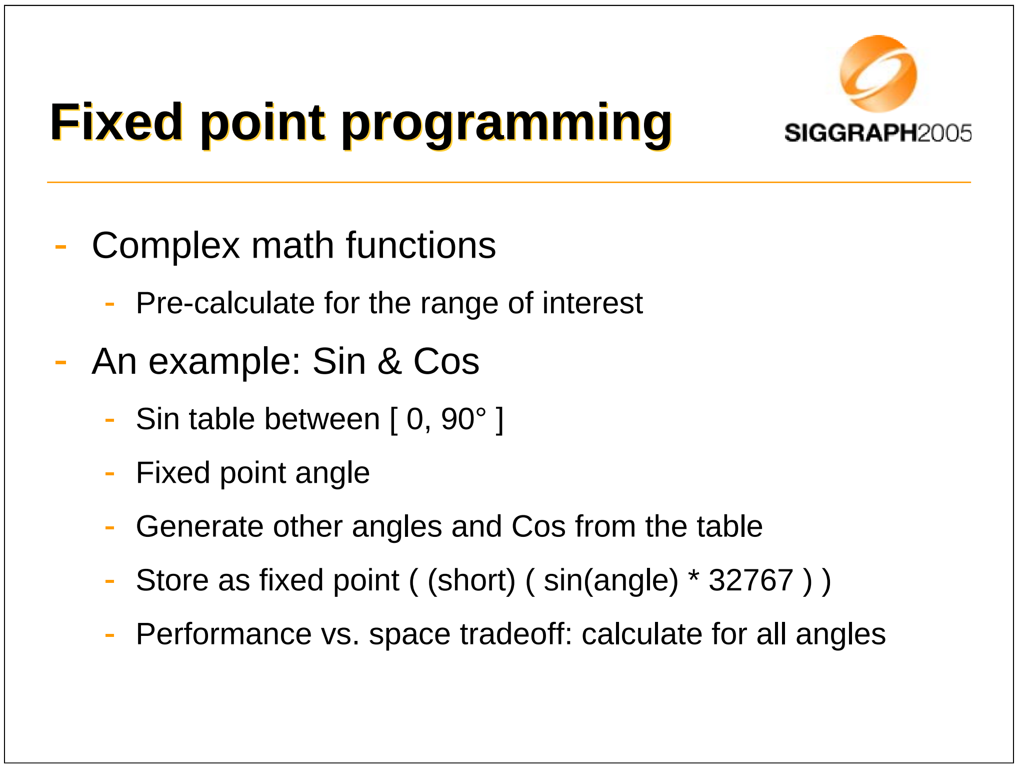
Fixed point programming

SIGGRAPH2005

```
#if defined(DEBUG)
int add_fix_fix_chk(int a, int b)
{
  int64 bigresult = ((int64)a) + ((int64)b);
  int smallresult = a + b;
  assert(smallresult == bigresult);
  return smallresult;
}
#endif

#if defined(DEBUG)
#  define add_fix_fix(a,b) add_fix_fix_chk(a,b)
#else
#  define add_fix_fix(a,b) ((a)+(b))
#endif
```

-Do all of the fixed point operations with macros and not by direct calculus.

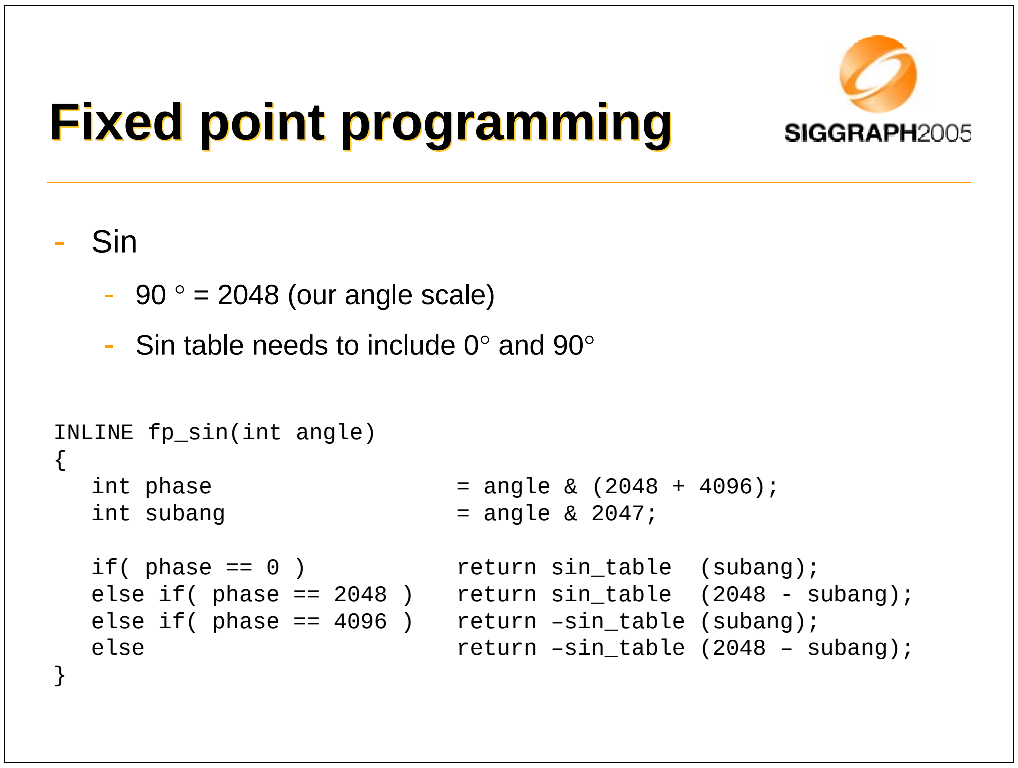
-Create DEBUG variants for every operation you do in fixed point (even simplest ADD, MUL, …). When you are compiling debug builds, all operations should assert that no overflows occur. If overflow assert is triggered, something needs to be done (ignore if not big enough visual impact, change ranges, etc.).



Fixed point programming

SIGGRAPH2005

- Complex math functions
  - Pre-calculate for the range of interest
- An example: Sin & Cos
  - Sin table between [0, 90°]
  - Fixed point angle
  - Generate other angles and Cos from the table
  - Store as fixed point ((short) (sin(angle) * 32767))
  - Performance vs. space tradeoff: calculate for all angles



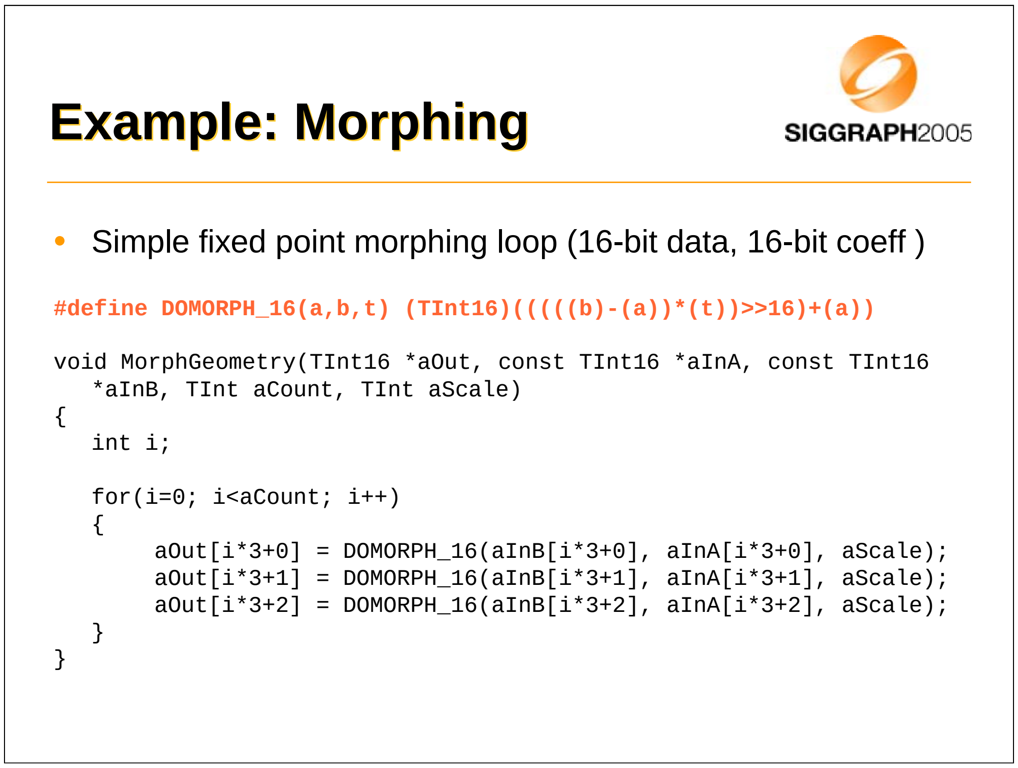
Fixed point programming

SIGGRAPH2005

- Sin
  - 90 ° = 2048 (our angle scale)
  - Sin table needs to include 0° and 90°

```
INLINE fp_sin(int angle)
{
   int phase                    = angle & (2048 + 4096);
   int subang                   = angle & 2047;

   if( phase == 0 )             return sin_table  (subang);
   else if( phase == 2048 )     return sin_table  (2048 - subang);
   else if( phase == 4096 )     return –sin_table (subang);
   else                         return –sin_table (2048 – subang);
}
```

- This function can be easily converted to be just single table lookup by precalculating SIN from 0 to 360+90 (both SIN and COS can then be referenced from the same table) if the angles are guaranteed to be between [0,360].



Example: Morphing

SIGGRAPH2005

- Simple fixed point morphing loop (16-bit data, 16-bit coeff)

```
#define DOMORPH_16(a,b,t) (TInt16)(((((b)-(a))*(t))>>16)+(a))

void MorphGeometry(TInt16 *aOut, const TInt16 *aInA, const TInt16
   *aInB, TInt aCount, TInt aScale)
{
   int i;

   for(i=0; i<aCount; i++)
   {
        aOut[i*3+0] = DOMORPH_16(aInB[i*3+0], aInA[i*3+0], aScale);
        aOut[i*3+1] = DOMORPH_16(aInB[i*3+1], aInA[i*3+1], aScale);
        aOut[i*3+2] = DOMORPH_16(aInB[i*3+2], aInA[i*3+2], aScale);
   }
}
```

-Morphing is done for 16-bit vertex data (16-bit vertices, 16-bit normals).

-This is done to make the fixed point math to fit inside of 32-bit integers.

-Standard 32-bit mul and addition is enough here.

Example: Morphing

SIGGRAPH2005

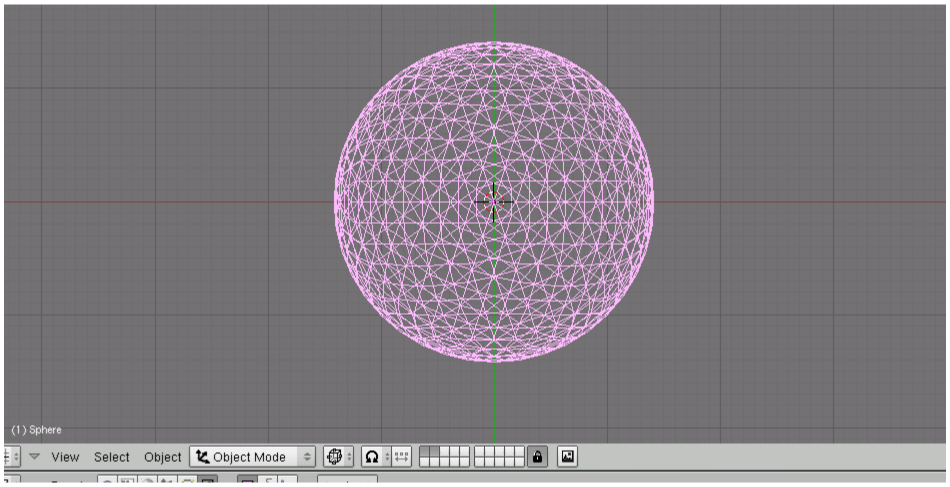


Image from Blender (http://www.blender3d.org)

-With the course notes there is a blender exporter that can export vertices and normals from the currently active object.

-Exporter is a python script that can be loaded into a script window and executed there.

-Exporter outputs directly C header files.

-In this example, we start with a tessellated sphere and export that as a C header.

Example: Morphing

SIGGRAPH2005

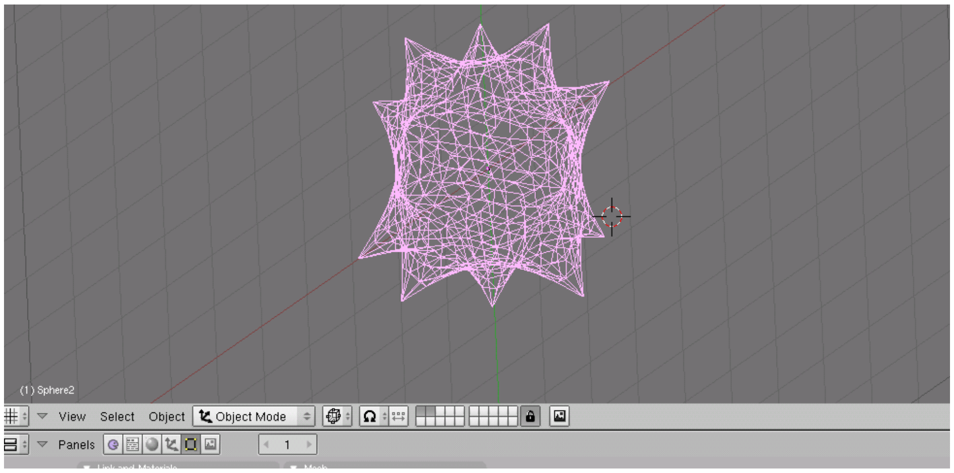


Image from Blender (http://www.blender3d.org)

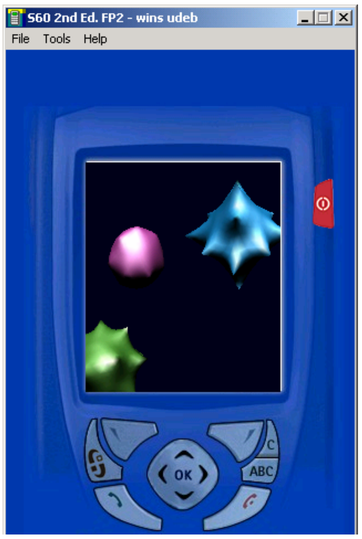
-In second phase, we modify the geometry, but do not add or remove vertices or triangles and export this geometry to a separate .H file.

-This is our second morph target.

Example: Morphing

SIGGRAPH2005

- Source code in the course notes
- Blender exporter in course notes
- 2 Blender models
- Exported vertices and normals
- Fixed point morphing for both



-After the .H files are exported we can just include them into the morphing example code and morph our vertices and normals freely.

-Check the course material.

Questions?

SIGGRAPH2005

? ? ?