
MOBILEHCIOS
APPLICATION DEVELOPMENT FOR MOBILE DEVICES
Enrico Rukzio
Michael Rohs
Daniel Wagner
John Hamard


MOBILEHCIOS
APPLICATION DEVELOPMENT FOR MOBILE DEVICES
Enrico Rukzio
Michael Rohs – Symbian
Daniel Wagner
John Hamard

Michael Rohs
michael.rohs@telekom.de

Goals

Learn enough about Symbian
- to judge whether is the right system for a project
- to quickly start application development in C++

Understand
- basic concepts of the Symbian operating system and the application framework
- how to set up a C++ development environment
- how to create a basic "Hello World" application

No mobile HCI course

Michael Rohs
michael.rohs@telekom.de

Symbian Characteristics

Symbian OS from the developer' s point of view =
operating system for handheld devices with limited resources
+ user interface framework
+ APIs (C++)
+ tools

Operating system
- pre-emptive multitasking, multithreading, memory protection, event-based
- special features: conserving memory, reliability, CPU switched off when applications are not dealing with events

Symbian OS from the user' s view
- consistent user interface, comprehensive set of PIM applications, extensible by 3rd party applications

Michael Rohs
michael.rohs@telekom.de

Outline

Symbian overview
Symbian OS structure
Setting up a development environment
Creating a "Hello World" example
Basic Symbian OS concepts
Application framework

Michael Rohs
michael.rohs@telekom.de

Outline

- Symbian overview
  - History
  - Market share
  - Device classes
- Symbian OS structure
- Setting up a development environment
- Creating a "Hello World" example
- Basic Symbian OS concepts
- Application framework

Michael Rohs
michael.rohs@telekom.de

History

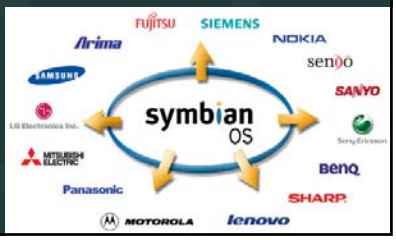
1997: Psion's EPOC OS

1998: Symbian consortium and Symbian OS

- independent of Psion
- Ericsson, Nokia, Motorola, Psion (founders)
- Panasonic, Sony Ericsson, Siemens, Samsung (shareholders)

2000: First Symbian OS phone

- Ericsson R380



Michael Rohs
michael.rohs@telekom.de

Symbian Shipments in 2004

Full year 2004: 14.38 million phones

- Q3 2004: 3.70 million phones
- Q4 2004: 5.68 million phones
- 41 phones models from 8 licensees

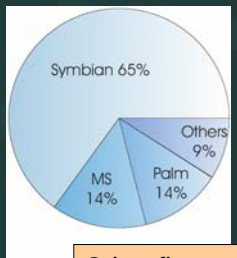
2005 shipments

- Q1 2005: 6.75 million phones
- 48 models from 9 licensees shipping

Installed base of almost 32 million phones

All mobile phone shipments full year 2004:
- 664.5 million (Source: IDC, January 27, 2005)
- 683.5 million (Source: Strategy Analytics)

Market share smartphones



Other figures:
Symbian: 80%
Microsoft: 7%

Sources: Symbian Ltd., Small Devices Ltd.

Michael Rohs
michael.rohs@telekom.de

Symbian Device Classes and OS Variants

Same basis, different look & feel

Series 60 

- smartphones with numeric keypad, "phone centric"

Series 80 

- organizers with full keyboard, "information centric"

UIQ 

- smartphones with pen input, "information centric"

Michael Rohs
michael.rohs@telekom.de

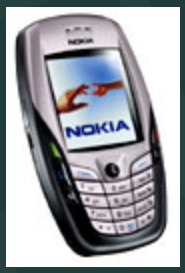
Series 60 Example Device: Nokia 6600

OS: Symbian OS 7.0s
CPU: ARM-9, 104 Mhz
Memory
- heap size: 3 MB
- storage: 6 MB + MMC

Display
- 65536 colors
- 176 x 208 pixels

Interaction
- numeric keypad
- 2 labeled soft keys
- 5-way direction key

Infrared, Bluetooth
Camera: 640 x 480 pixels



- Java: CLDC 1.0, MIDP 2.0
  - Wireless Messaging API (JSR-120)
  - Mobile Media API (JSR-135)
  - Bluetooth API (JSR-82 No OBEX)

Michael Rohs
michael.rohs@telekom.de

Series 80 Example Device: Nokia 9300

OS: Symbian OS 7.0s
CPU: ARM-based, ~150 MHz
Memory
- heap size: 20 MB
- storage: 80 MB + MMC

Display
- 65536 colors
- 128 x 128 pixels (outside)
- 640 x 200 pixels (inside)

Interaction
- numeric and qwerty keypads
- 3 labeled soft keys
- 9-way direction key

Infrared, Bluetooth

- Java CLDC 1.1, MIDP 2.0
  - Wireless Messaging API (JSR-120)
  - Mobile Media API (JSR-135)
  - Bluetooth API (JSR-82 No OBEX)
  - FileConnection and PIM API (JSR-75)
- Java CDC 1.0 (JSR-36)
  - Personal Profile (JSR-62)
  - Foundation Profile (JSR-46)

Michael Rohs
michael.rohs@telekom.de

UIQ Example Device: Sony Ericsson P910

OS: Symbian OS 7.0
CPU: ARM-9, 156 MHz
Memory
- heap size: 32 MB
- storage: 64 MB + Memory Stick

Display (touch sensitive)
- 262000 colors
- 208 x 320 pixels

Interaction
- stylus on touchscreen
- numeric and qwerty keypad
- 5-way direction key

- Java
  - CLDC 1.1
  - MIDP 2.0
- Infrared, Bluetooth
- Camera: 640 x 480 pixels

Michael Rohs
michael.rohs@telekom.de

Outline

Symbian overview
Symbian OS structure
- Hardware resources
- Software components
- Processes, threads, and context switching
- Kernel and user library
- Event handling and active objects
- Client-server framework

Setting up a development environment
Creating a "Hello World" example
Basic Symbian OS concepts
Application framework

Michael Rohs
michael.rohs@telekom.de

Hardware Resources

CPU: ARM, 32-bit, 36-220 MHz
ROM: OS, middleware, applications, Z: drive
RAM: working memory, "disk" space, C: drive
I/O: keypad, pen input, CF card slot, CCD camera
Communication: GSM/GPRS/UMTS, infrared, Bluetooth
Power source: rechargable battery 2000-3000 mWh (600-900 mAh, 3.5 V)
Display: 176x208 pixels, 65536 colors for smartphones

Michael Rohs
michael.rohs@telekom.de

Memory Organization and Disk Drives

RAM
- runtime memory (stack, heap)

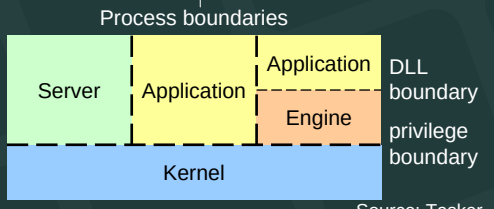
C: Flash-RAM
- application files, user files

D, E, ...: compact flash cards, memory sticks
- application files, user files

Z: ROM
- operating system files

Michael Rohs
michael.rohs@telekom.de

Software Components



Source: Tasker

Kernel
- manage and controls access to hardware resources
- hardware-supported privileges, kernel mode

Application
- program with a user interface
- runs in user mode in its own process

Server
- program without a user interface
- manages resources, provides interface to clients

Engine
- application part that manipulates data, often separate DLL

Michael Rohs
michael.rohs@telekom.de

Processes, Threads, and Context Switching

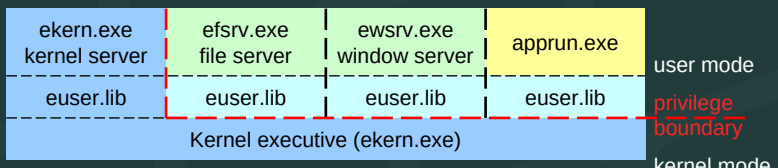
Process: fundamental unit of protection
- own address space
- virtual addresses –MMU→ physical addresses

Thread: fundamental unit of execution
- one or more per process
- preemptively scheduled by the kernel

Context switching: changing threads
- expensive between two threads in different processes
- typically just one thread per process → active objects

Michael Rohs
michael.rohs@telekom.de

Kernel and User Library



Source: Tasker

Kernel runs in privileged mode
Kernel server: highest-priority thread in system
User library (euser.lib): basic library and kernel access
- entirely user-side (e.g. descriptors, arrays)
- cross into kernel executive (e.g. time checking)
- request services of the kernel server

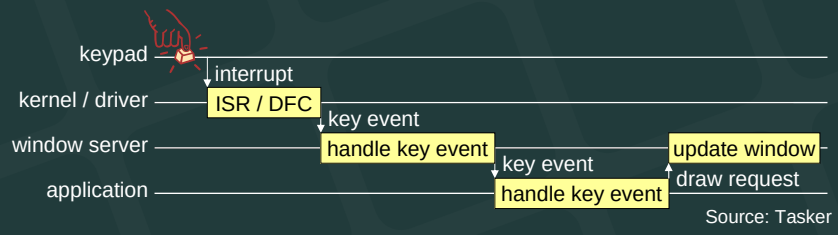
Michael Rohs
michael.rohs@telekom.de

Event Handling and Active Objects

Symbian optimized for efficient event handling

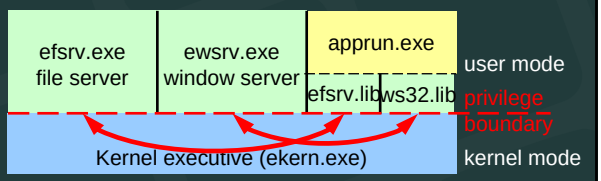
Active objects and active schedulers

- non-preemptive event handling within a single thread
- well-suited for GUI systems
- active objects call asynchronous methods or register for events
- active scheduler calls RunL method of active object



Michael Rohs
michael.rohs@telekom.de

Client-Server Framework



Most important servers

- **file server:** operations on files
- **window server:** user input, screen drawing
- communications, databases, schedule, contacts, etc.

Client-server interface

- server proxy in client's address space (e.g. **R**File, **R**Window)
- kernel-supported message passing
- inter-thread read / write: server can access client's address space

Michael Rohs
michael.rohs@telekom.de

Outline

- Symbian overview
- Symbian OS structure
- Setting up a development environment
  - Development tools
  - Installation issues
  - Application wizard
  - Build process
- Creating a "Hello World" example
- Basic Symbian OS concepts
- Application framework

Michael Rohs
michael.rohs@telekom.de

Symbian Series 60 and UIQ Development Tools

Microsoft Visual Studio – C++ IDE (MSVS)
Metrowerks Code Warrior – C++ IDE (alternative)
[Makmake – makefiles for Windows and ARM, project files for MSVS]
Bldmake – generates abld.bat from bld.inf
Abld – Perl tool, creates makefiles, compiles project
Makesis – creates installation file ".sis" from ".pkg" file

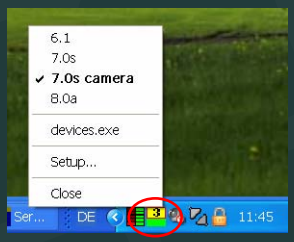
Michael Rohs
michael.rohs@telekom.de

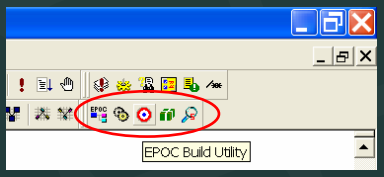
Symbian Series 60 and UIQ Development Tools

Emulator
- debugging on Windows ("epoc.exe")

Environment Switch
- choose SDK

EpocToolbar
- Plugin for MSVS





Michael Rohs
michael.rohs@telekom.de

Installation of the Symbian 7.0s SDK on Windows using Visual C++ 6.0

General hints
- avoid spaces in path and file names
  - especially in the installation path
- the installation directory has to be on volume C:
- installation requires "Administrator" privileges
- you need lots of disk space for Symbian
  - (about 720 MB for Nokia 6600 with camera API plugin)
- SDKs are device specific and version specific
  - see: http://www.forum.nokia.com/main/1,6566,034-4,00.html

Michael Rohs
michael.rohs@telekom.de

Installation of the Symbian 7.0s SDK on Windows using Visual C++ 6.0

Installation order

- install MS Visual C++ 6.0 (and latest service pack)
  - http://msdn.microsoft.com/vstudio/previous/vs6/downloads/default.aspx
- install latest Java J2SE
  - http://java.sun.com/j2se/
- install latest Active Perl
  - http://www.activestate.com/Products/ActivePerl/
- install Series 60 SDK 2.0 for Symbian OS
  - http://www.forum.nokia.com/main/1,6566,034-4,00.html

Michael Rohs
michael.rohs@telekom.de

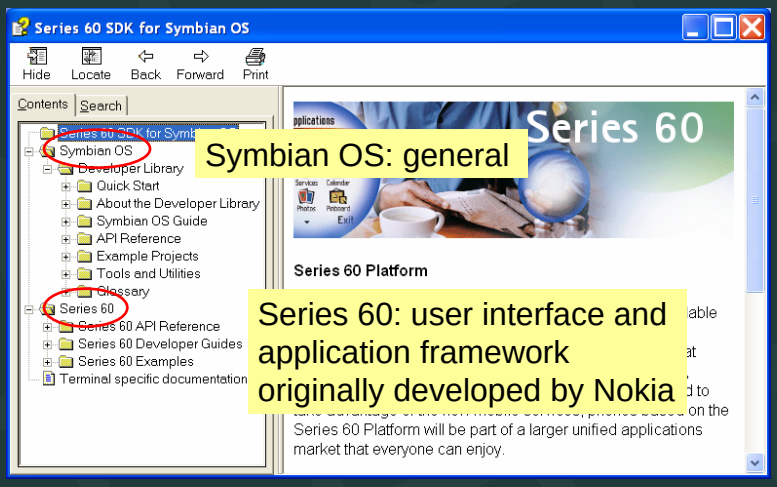
Installation directories

Root: C:\dev\Symbian\7.0s\Series60_v20

- Epoc32 – gcc compiler, emulator, include files, etc.
- Examples – SDK examples
- Series60Doc – SDK documentation
- Series60Ex – Series 60 specific examples
- Series60Tools – tools

Michael Rohs
michael.rohs@telekom.de

Platform SDK Documentation



Location: C:\dev\Symbian\7.0s\Series60_v20\Series60Doc\Start.chm

Michael Rohs
michael.rohs@telekom.de

Installing the Application Wizard Plug-in for Visual C++ 6.0

Location

- C:\dev\Symbian\7.0s\Series60_v20\Series60Tools\applicationwizard

Copy "avkonappwiz_v20.awx" and "avkonappwiz_v20.hlp" to

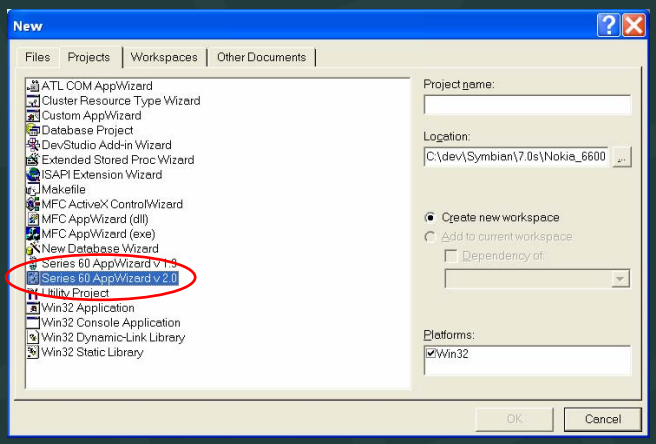
- C:\Program Files\Microsoft Visual Studio\Common\MSDev98\Template

Start MSVS

Standalone application wizard for Symbian 8.0a

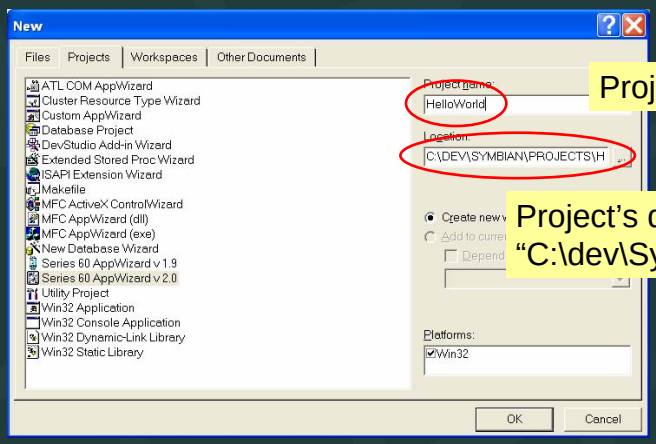
Michael Rohs
michael.rohs@telekom.de

Using the Application Wizard



Michael Rohs
michael.rohs@telekom.de

Using the Application Wizard

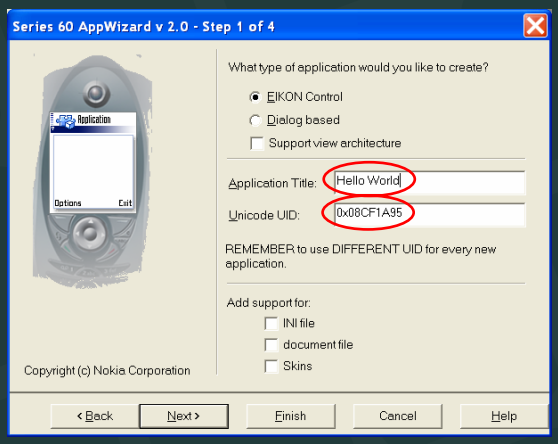


Project “HelloWorld”

Project’s directory
“C:\dev\Symbian\Projects”

Michael Rohs
michael.rohs@telekom.de

Using the Application Wizard

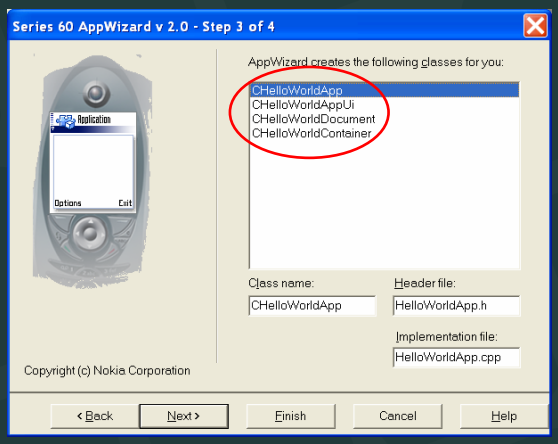


Application title will appear on the device

Each application has a unique identifier

Michael Rohs
michael.rohs@telekom.de

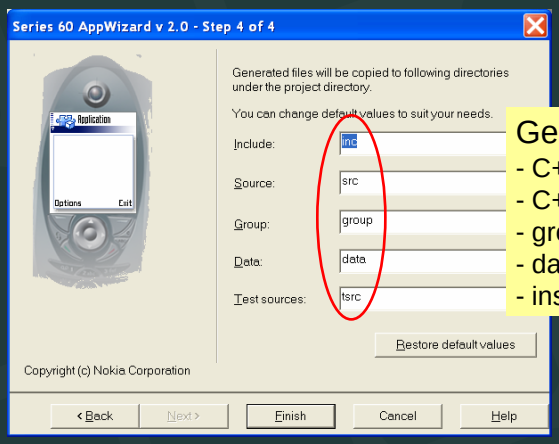
Using the Application Wizard



Generated classes:
- Application
- Document
- User interface
- Container

Michael Rohs
michael.rohs@telekom.de

Using the Application Wizard



Generated directories:
- C++ include files
- C++ source files
- group: Symbian project files
- data, aif: resource files
- install: installation files

Michael Rohs
michael.rohs@telekom.de

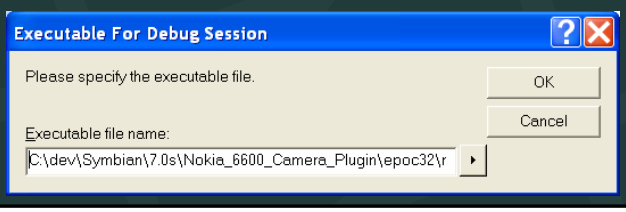
Debugging Programs in the Emulator

Compile (F7)

Execute (Ctrl + F5)

Enter file name of
emulator executable (epoc.exe)

- ...\epoc32\release\wins\udeb\epoc.exe



Michael Rohs
michael.rohs@telekom.de

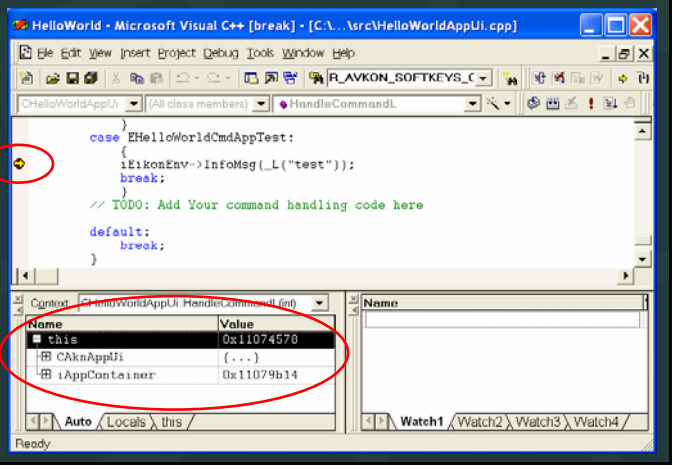
Debugging Programs in the Emulator

Set break-points (F9)

Debug (F5)

Break-point

Inspect variables

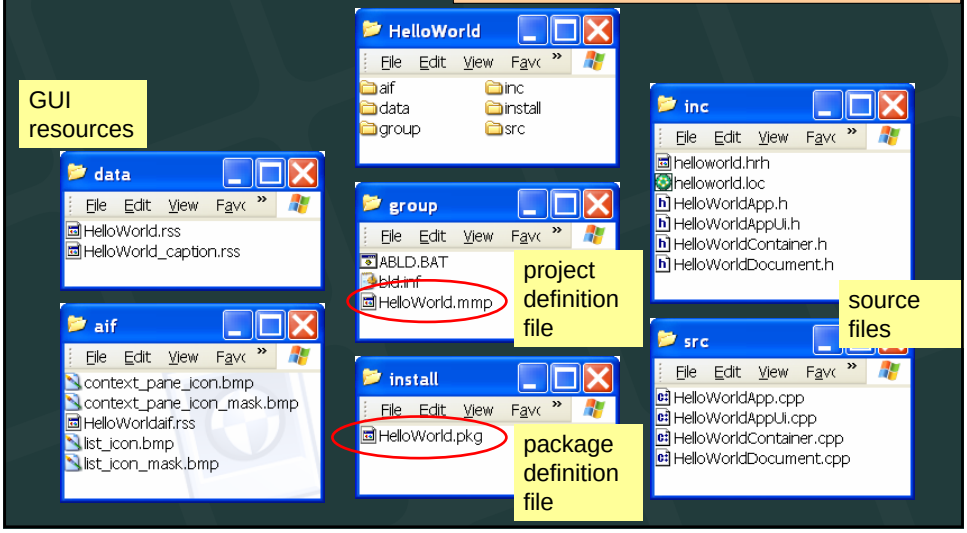


Michael Rohs
michael.rohs@telekom.de

Generated Files

Visual C++ specific files in:
C:\dev\Symbian\7.0s\Series60_v20\Epoc32\BUILD\DEV\SYMBIAN\PROJECTS\HELLOWORLD

GUI resources



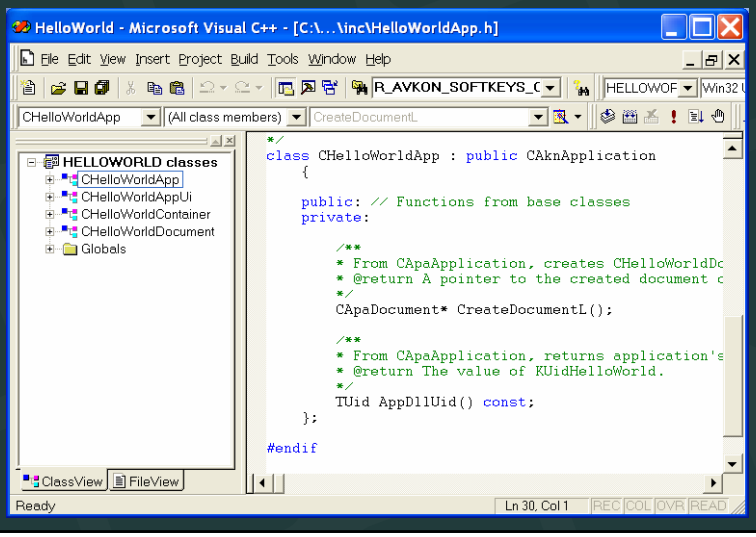
project definition file

source files

package definition file

Michael Rohs
michael.rohs@telekom.de

Generated C++ Source Code



Michael Rohs
michael.rohs@telekom.de

Project Definition File HelloWorld.mmp

```
TARGET  HelloWorld.app
TARGETTYPE  app
UID  0x100039CE 0x08CF1A95
TARGETPATH \system\apps\HelloWorld

SOURCEPATH ..\src
SOURCE  HelloWorldApp.cpp
SOURCE  HelloWorldAppUi.cpp
SOURCE  HelloWorldDocument.cpp
SOURCE  HelloWorldContainer.cpp

SOURCEPATH ..\data
RESOURCE HelloWorld.rss
RESOURCE HelloWorld_caption.rss
LANG     SC

USERINCLUDE .
USERINCLUDE ..\inc

SYSTEMINCLUDE .
SYSTEMINCLUDE \epoc32\include
```

```
LIBRARY euser.lib
LIBRARY apparc.lib
LIBRARY cone.lib
LIBRARY eikcore.lib
LIBRARY eikcoctl.lib
LIBRARY avkon.lib

AIF HelloWorld.aif ..\aif
   HelloWorldaif.rss c8
   context_pane_icon.bmp
   context_pane_icon_mask.bmp
   list_icon.bmp list_icon_mask.bmp
```

mak**m**ake **p**roject

After each change to the mmp file you need to update the Visual C++ project with **`abld makefile vc6`**

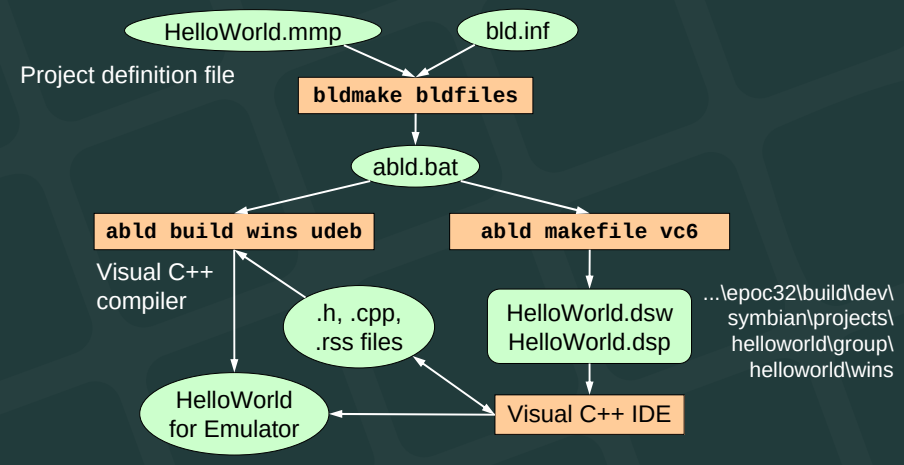
Michael Rohs
michael.rohs@telekom.de

Package Definition File HelloWorld.pkg

```
;Languages
&EN

; UID
#{"Hello World"},(0x08CF1A95),1,0,0

;Supports Series 60 v 2.0
(0x101F7960), 0, 0, 0, {"Series60ProductID"}

; Four files to install
"\dev\Symbian\7.0s\Nokia_6600_Camera_Plugin\Epoc32\release\thumb\urel\Hell
   oWorld.app" - "!:\system\apps\HelloWorld\HelloWorld.app"
"\dev\Symbian\7.0s\Nokia_6600_Camera_Plugin\Epoc32\data\z\system\apps\Hell
   oWorld\HelloWorld.rsc" - "!:\system\apps\HelloWorld\HelloWorld.rsc"
"\dev\Symbian\7.0s\Nokia_6600_Camera_Plugin\Epoc32\data\z\system\apps\Hell
   oWorld\HelloWorld_caption.rsc" -
   "!:\system\apps\HelloWorld\HelloWorld_caption.rsc"
"\dev\Symbian\7.0s\Nokia_6600_Camera_Plugin\Epoc32\data\z\system\apps\Hell
   oWorld\HelloWorld.aif" - "!:\system\apps\HelloWorld\HelloWorld.aif"
```

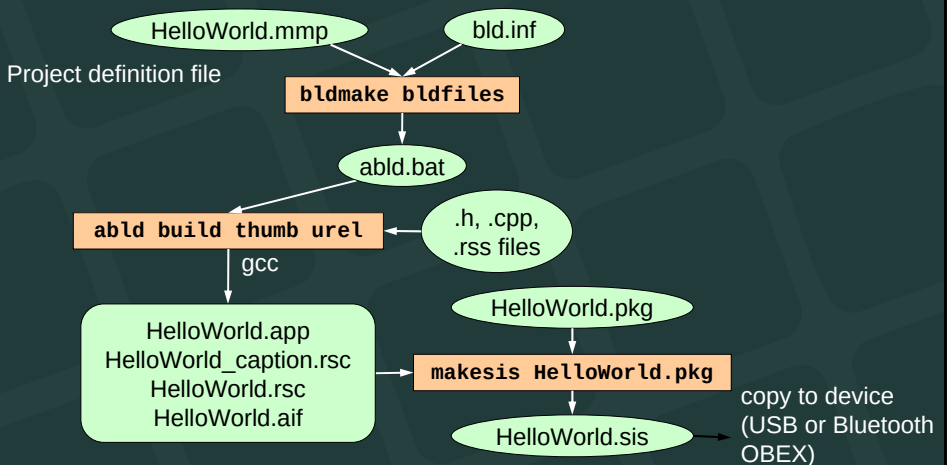

Michael Rohs
michael.rohs@telekom.de

Build Process for Emulator (Intel / Windows)



Michael Rohs
michael.rohs@telekom.de

Build Process for Target Device (Arm / Symbian)



Michael Rohs
michael.rohs@telekom.de

Using Visual Studio .NET with Symbian OS SDKs (officially unsupported)

"abld makefile vc6" creates .dsw and .dsp files
open .dsp file in Visual Studio .NET
change project properties:

- Code Generation: Enable C++
- Exceptions: No
- Buffer Security Check: No
- Command Line
- Additional Options: /QIfist

set PATH environment variable to point to the new Visual Studio .NET
add the /QIfist flag to \epoc32\tools\cl_win.pm
http://www3.symbian.com/faq.nsf/0/30398B3E9500A24D80256C7F00693A91?OpenDocument

Michael Rohs
michael.rohs@telekom.de

Outline

- Symbian overview
- Symbian OS structure
- Setting up a development environment
- Creating a "Hello World" example
- Basic Symbian OS and Symbian C++ concepts
  - Naming conventions
  - Basic types
  - Exception handling
  - Resource management
  - Descriptors
- Application framework

Michael Rohs
michael.rohs@telekom.de

Naming Conventions for Types

One character prefix for consistency, readability

T ("type") classes: no pointers, no resources — **T**Point

C ("cleanup") classes: reside on the heap, derived from CBase, virtual destructor — **C**AknApplication

R ("resource") classes: client-side handle to an OS resource (file server, message server, etc.) — **R**Socket

M ("mixin") classes: interface, no implementation, pure virtual functions, observer design pattern — **M**EikMenuObserver

E ("enumeration") types: enumerations — **E**KeyOK

Michael Rohs
michael.rohs@telekom.de

Naming Conventions for Variables and Constants

One character prefix for consistency, readability

i: member (instance) variables — **i**Document

a: method arguments — **a**Index

no prefix: local variables on the stack — size

K: constants — **K**Pi

Method names: capitalized — **S**etName

Michael Rohs
michael.rohs@telekom.de

Basic Types – Integers, Void, Boolean

Signed integer

Name	Size (bytes)
TInt	≥4
TInt8	1
TInt16	2
TInt32	4
TInt64	8

Unsigned integer

Name	Size (bytes)
TUint	≥4
TUint8	1
TUint16	2
TUint32	4

Void

Name	Type
TAny	void

Boolean

Name	Size
TBool	4

Michael Rohs
michael.rohs@telekom.de

Basic Types – Floating Point, Characters

Floating point

Name	Size (bytes)
TReal	8
TReal32	4
TReal64	8
TRealX	12

Character

Name	Size (bytes)
TText8	1
TText	2
TText16	2
TChar	4

Hint: Minimize the use of floating point operations, since typical Symbian machines don't include hardware FPUs.

Michael Rohs
michael.rohs@telekom.de

Method Parameter Passing

	by value	by & reference	by * reference
input	`X`	`const X&`	`const X*`
output		`X&`	`X*`

"small" types
- by value

"larger" types
- by **&** reference preferred
- by ***** reference if **NULL** is possible or for transferring ownership

Michael Rohs
michael.rohs@telekom.de

Error Handling and Cleanup

Reliability on continuously running resource constrained devices

- Use memory efficiently and safely
- Release resources as early as possible
- Cope with out-of-memory and other errors and roll back to a consistent state when they occur

Michael Rohs
michael.rohs@telekom.de

Exception Handling

Symbian's own variant of C++ try, catch, throw

C++ throw → Symbian User::Leave()

C++ try, catch → Symbian TRAP macro

Cleanup stack

- clean up heap when exception occurs

Two-phase construction of complex objects

- 1st phase constructors never leave
- 2nd phase constructors can

Michael Rohs
michael.rohs@telekom.de

Exceptions, Panics, Leaves, and Traps

- **Exception**: runtime error that **is not** the programmer's fault
- **Panic**: programmatic error that **is** the programmer's fault
- **Leave**: execution leaves the method and jumps up the call stack to the closest exception handler
- **Trap** / **trap harness**: central exception handler

Michael Rohs
michael.rohs@telekom.de

Throw Exceptions: User::Leave()

L indicates that method may leave (naming convention)

```
void DoExampleL()
{
    CExample* example = new CExample;

    if (example == NULL) {
        User::Leave(KErrNoMemory);
    }

    // do something with example

    delete example;
}
```

throw exception with reason code

Michael Rohs
michael.rohs@telekom.de

Catch Exceptions: TRAP Macro

```
TInt r = 0; // leave
   variable

TRAP(r, DoExampleL());

if (r != KErrNone) {
    // handle error
}
```

trap harness will catch a leave in **DoExampleL**

error can be handled here, new leaves can be thrown

- L-functions do not need to be called from a trap harness. There must be one somewhere up the call-stack.
- traps may be nested

Michael Rohs
michael.rohs@telekom.de

Cleanup Stack: Avoid Memory Leaks

L indicates that method may leave (naming convention)

can be checked using leavescan <filename.cpp>

```
void DoExampleL()
{
    CExample* example = new (ELeave) CExample;

    // cannot leave: no protection needed
    example->iInt = 5;

    // can leave: use cleanup stack
    CleanupStack::PushL(example);
    example->DoSomethingL();
    CleanupStack::Pop();

    delete example;
}
```

overwritten operator **new**, may leave, **example** is never **NULL**

- method may leave,
- **example** would become orphaned,
- **CleanupStack** deletes **example** in this case

Michael Rohs
michael.rohs@telekom.de

Cleanup Stack: Avoid Memory Leaks

```
void DoExampleL()
{
    CExample* example = new (ELeave) CExample;

    // cannot leave: no protection needed
    example->iInt = 5;

    // can leave: use cleanup stack
    CleanupStack::PushL(example);
    example->DoSomethingL();
    CleanupStack::Pop();

    delete example;
}
```

can be replaced by:
`CleanupStack::PopAndDestroy();`

or give number of objects to pop and destroy:
`CleanupStack::PopAndDestroy(1);`

Michael Rohs
michael.rohs@telekom.de

Two-Phase Construction of Objects: The C++ Constructor Must Never Leave

```
class CExample : public CBase {
    CExample();
    CElement* iElement;
};

CExample::CExample() {
    iElement = new (ELeave) CElement();
}
```

```
...
CExample example = new (ELeave) CExample();
...
```

Michael Rohs
michael.rohs@telekom.de

Two-Phase Construction of Objects: The 2nd-Phase Constructor May Leave

```
class CExample : public CBase {
    CExample() {
        iElement = NULL; // CBase takes care of zero-initialization
    }
    void ConstructL() {
        iElement = new (ELeave) CElement();
    }
    CElement* iElement;
};
```

```
...
CExample example = new (ELeave) CExample();
CleanupStack::PushL(example);
example->ConstructL();
CleanupStack::Pop();
...
```

can be hidden inside a static factory method: **NewL** or **NewLC**

Michael Rohs
michael.rohs@telekom.de

Two-Phase Construction of Objects: Static Factory Method NewL

```
class CExample : public CBase {
    CExample() { iElement = NULL; }
    void ConstructL() {
        iElement = new (ELeave) CElement();
    }
    static void NewL() {
        CExample self = new (ELeave) CExample();
        CleanupStack::PushL(self);
        self->ConstructL();
        CleanupStack::Pop();
    }
    CElement* iElement;
};
```

naming convention: C denotes that object remains on cleanup stack

```
static void NewLC() {
    CExample self =
      new (ELeave) CExample();
    CleanupStack::PushL(self);
    self->ConstructL();
}
```

```
...
CExample example = CExample::NewL();
...
```

call static factory method

Michael Rohs
michael.rohs@telekom.de

Two-Phase Construction of Objects: Use Protected Constructors with NewL

```
class CExample : public CBase {
public:
    static void NewL() {
        CExample self = new (ELeave) CExample();
        CleanupStack::PushL(self);
        self->ConstructL();
        CleanupStack::Pop();
    }
    virtual ~CExample() { delete iElement; }
protected:
    CExample() { iElement = NULL; }
    void ConstructL() {
        iElement = new (ELeave) CElement();
    }
    CElement* iElement;
};
```


Michael Rohs
michael.rohs@telekom.de

Exception Handling Quiz: What's the problem with this code?

```
class CExample : public CBase {
public:
    ...
    void SetNameL(const TDesC& aName) {
        delete iName;
        iName = aName.AllocL();
    }
    ...
private:
   HBufC* iName;
};
```

What if **AllocL** leaves?

Michael Rohs
michael.rohs@telekom.de

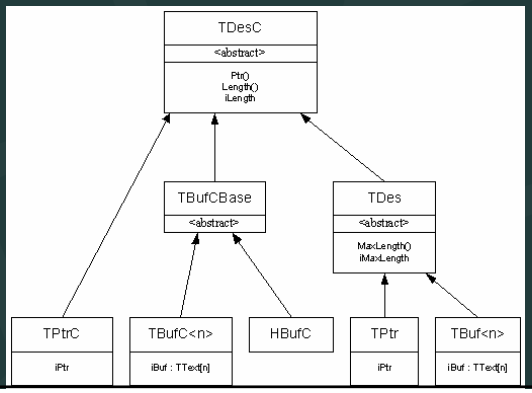
Exception Handling Quiz: What's the problem with this code?

```
class CExample : public CBase {
public:
    ...
    void SetNameL(const TDesC& aName) {
        delete iName;
        iName = NULL;
        iName = aName.AllocL();
    }
    ...
private:
   HBufC* iName;
};
```

Michael Rohs
michael.rohs@telekom.de

Descriptors: Containers for (16-bit) Text and (8-bit) Binary Data

- Runtime bounds checking
- API for buffer modification
- Used extensively in Symbian OS APIs
- Modifiable and nonmodifiable



Michael Rohs
michael.rohs@telekom.de

Descriptor Classes

Name	Instatiable	Modifiable	Description
TDesC	no	no	abstract base class; argument passing and return values (**const TDesC&**)
TDes	no	yes	abstract base class; argument passing return values (**const TDes&**)
TBufC	yes	no	templated buffer descriptor: (size, data) stored together
TPtrC	yes	no	pointer descriptor: (size, pointer) separate from data, data not owned
TBuf	yes	yes	templated buffer descriptor: (size, maximum size, data) stored together
TPtr	yes	yes	pointer descriptor: (size, maximum size, pointer) separate from data, data not owned
HBufC	yes	no	heap descriptor: (size, maximum size, data) stored together on the heap

Michael Rohs
michael.rohs@telekom.de

Descriptor API: Search, Extract, Insert, Delete, Replace, Format

Nonmodifiable

- TInt **Length**()
- TInt **Size**()
- const TUint* **Ptr**()
- HBufC* **Alloc**()
- TPtrC **Left**(TInt aLength)
- TPtrC **Mid**(TInt aPos, TInt aLength)
- TInt **Find**(const TDesC& aDes)
- const TUint16& **operator[]**()

- Modifiable
  - TInt **MaxLength**()
  - TInt **MaxSize**()
  - void **SetLength**(TInt aLength)
  - void **Append**(const TDesC& aDes)
  - void **Insert**(TInt aPos, const TDesC& aDes)
  - void **Delete**(TInt aPos,TInt aLength);

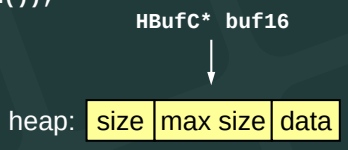
Michael Rohs
michael.rohs@telekom.de

Heap Buffer Descriptors HBufC: heap: (size, max size, data)

Used if descriptor size unknown at compile time

Change to pointer descriptor to modify

```
HBufC* buf16 = HBufC::NewLC(buf8.Length());
TPtr ptr16 = buf16->Des();
ptr16.Copy(buf8);

buf16 = buf16->ReAlloc(4 * buf8.Length());
CleanupStack::Pop();
CleanupStack::Push(buf16);
...
CleanupStack::PopAndDestroy(buf16);
```



Michael Rohs
michael.rohs@telekom.de

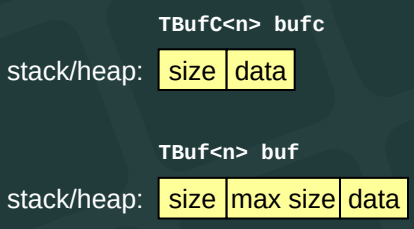
Templated Buffer Descriptors: TBufC (size, data) and TBuf (size, max size, data)

Templated with a size known at compile time

Modifiable (TBuf) and Non-modifiable (TBufC)

```
_LIT(KText, "this is a text");

TBuf<32> buf;
buf.Append(KText);
buf.AppendNum(123);
...
TBufC<32> bufc(KText);

bufc = buf;
```

assignment operator overloaded, replaces data

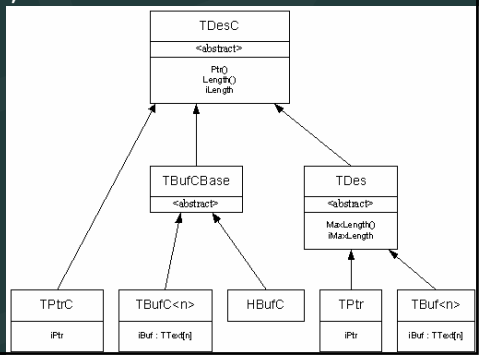


Michael Rohs
michael.rohs@telekom.de

Descriptors as Arguments and Return Types

Abstract base classes TDes and TDesC as formal parameters for flexibility

```
void SetName(const TDesC& aName);
const TDesC& Symbol() const;
```



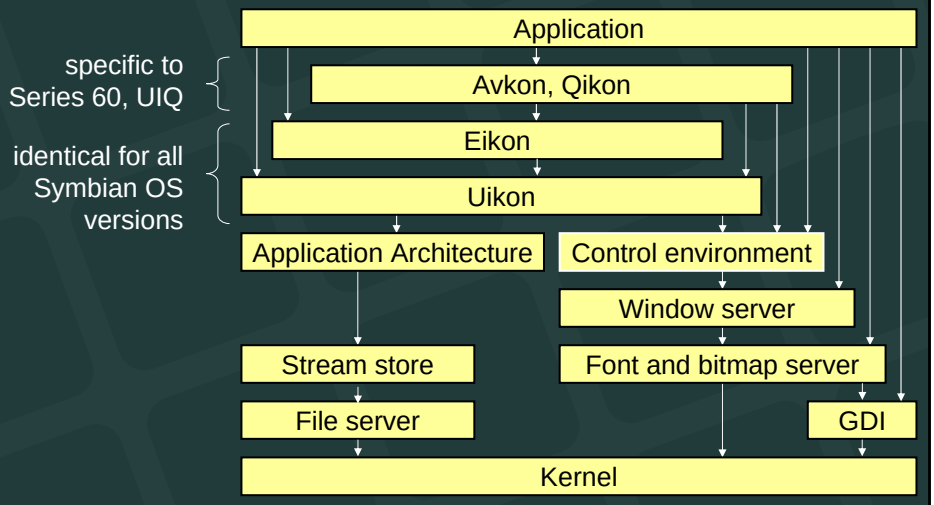
Michael Rohs
michael.rohs@telekom.de

Outline

Symbian overview
Symbian OS structure
Setting up a development environment
Creating a "Hello World" example
Basic Symbian OS and Symbian C++ concepts
Application Framework
- Application architecture
- GUI controls
- Resource files
- View architecture
- Event handling

Michael Rohs
michael.rohs@telekom.de

User Interface Architecture



Source: Gerlicher, Rupp

Michael Rohs
michael.rohs@telekom.de

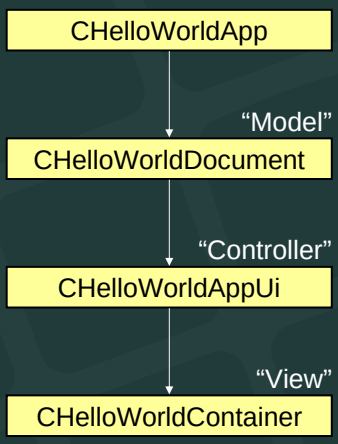
Application Architecture: Anatomy of HelloWorld

Entry point, information to framework (UID, icons, etc.)

Persistent application data

Handles or dispatches events from framework, manage controls

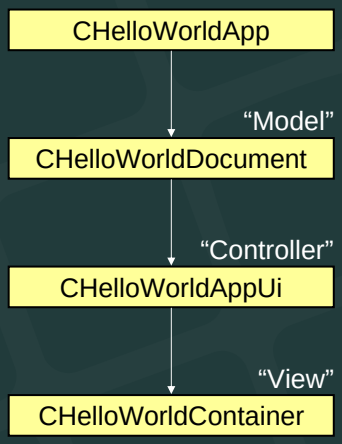
Container for controls (labels, list boxes, etc.)



Michael Rohs
michael.rohs@telekom.de

Application Architecture: Instantiation of HelloWorld by the Framework

```
E32Dll()
app = NewApplication()
uid = app->AppDllUid()
doc = app->CreateDocumentL()
ui = doc->CreateAppUiL()
[con = ui->ConstructL()]
con->Draw()
```



Michael Rohs
michael.rohs@telekom.de

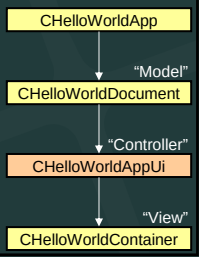
AppUi Methods for Event Handling, Called by the Framework

`HandleCommandL()`
- handles commands defined in resource files

`HandleKeyEventL()`
- key events

`HandleForegroundEventL(TBool aForeground)`
- application switched to foreground or background

and many more...



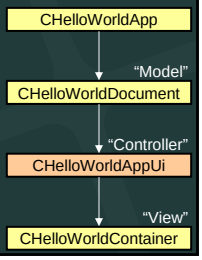
Michael Rohs
michael.rohs@telekom.de

AppUi Class Implementation (1/3): Construction

```
void CHelloWorldAppUi::ConstructL()
{
    BaseConstructL();
    iAppContainer = new (ELeave) CHelloWorldContainer;
    iAppContainer->SetMopParent(this);
    iAppContainer->ConstructL(ClientRect());
    AddToStackL(iAppContainer);
}

CHelloWorldAppUi::~CHelloWorldAppUi()
{
    if (iAppContainer) {
        RemoveFromStack(iAppContainer);
        delete iAppContainer;
    }
}
```

for interaction with controls that are not in this hierarchy

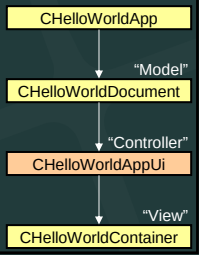
enables dispatching key events to controls



Michael Rohs
michael.rohs@telekom.de

AppUi Class Implementation (2/3): Handling Key Events

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

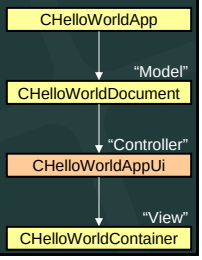


Michael Rohs
michael.rohs@telekom.de

AppUi Class Implementation (3/3): Handling Menu Selection Events

```
void CHelloWorldAppUi::HandleCommandL(TInt aCommand)
{
    switch (aCommand) {
        case EAknSoftkeyBack:
        case EEikCmdExit:
            Exit();
            break;
        case EHelloWorldCmdAppTest:
            iEikonEnv->InfoMsg(_L("test"));
            break;
        default:
        break;
    }
}
```

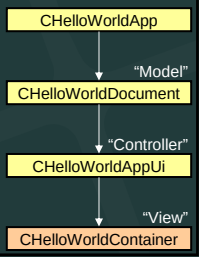
defined as a menu item in HelloWorld.rss



Michael Rohs
michael.rohs@telekom.de

AppUI Focus Events: Switching between Background and Foreground

```
void CHelloWorldAppUi::HandleForegroundEventL(TBool aForeground) {
    if (aForeground) iAppContainer->NewColor();
    CAknAppUi::HandleForegroundEventL(aForeground);
}

void CHelloWorldContainer::NewColor() {
   iCurrentColor++;
   if (iCurrentColor >= 4) iCurrentColor = 0;
}

void CHelloWorldContainer::Draw(const TRect& aRect) const {
    CWindowGc& gc = SystemGc();
    gc.SetPenStyle(CGraphicsContext::ENullPen);
    TRgb colors[] = { KRgbRed, KRgbGreen,
                      KRgbYellow, KRgbBlue };
    gc.SetBrushColor(colors[iCurrentColor]);
    gc.SetBrushStyle(CGraphicsContext::ESolidBrush);
    gc.DrawRect(aRect);
}
```



Michael Rohs
michael.rohs@telekom.de

Resource Definition File: HelloWorld.rss

```
RESOURCE EIK_APP_INFO
  {
  cba = R_AVKON_SOFTKEYS_OPTIONS_BACK;
  menubar = r_helloworld_menubar;
  }

RESOURCE MENU_BAR r_helloworld_menubar
  {
  titles =
    {
    MENU_TITLE { menu_pane = r_helloworld_menu; }
    };
  }

RESOURCE MENU_PANE r_helloworld_menu
  {
  items =
    {
    MENU_ITEM { command = EHelloWorldCmdAppTest; txt = "Test"; }
    MENU_ITEM { command = EAknCmdExit; txt = "Exit"; }
    };
  }
```

cba = command button area

Michael Rohs
michael.rohs@telekom.de

Container Class Declaration

```
class CHelloWorldContainer : public CCoeControl, MCoeControlObserver
{
public:
   void ConstructL(const TRect& aRect);
   ~CHelloWorldContainer();
private:
   void SizeChanged();
   TInt CountComponentControls() const;
   CCoeControl* ComponentControl(TInt aIndex) const;
   void Draw(const TRect& aRect) const;

   void HandleControlEventL(
   CCoeControl* aControl, TCoeEvent aEventType);
private:
   CEikLabel* iLabel;
   CEikLabel* iToDoLabel;
};
```

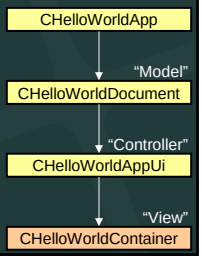
a blank canvas:
- to draw in
- to put controls into
- can be nested in a hierarchy
- can receive key events

Michael Rohs
michael.rohs@telekom.de

Container Class Implementation (1/3)

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

    iLabel = new (ELeave) CEikLabel;
    iLabel->SetContainerWindowL(*this);
    iLabel->SetTextL(_L("Example View"));

    iToDoLabel = new (ELeave) CEikLabel;
    iToDoLabel->SetContainerWindowL(*this);
    iToDoLabel->SetTextL(
        _L("Add Your controls\n here"));
    SetRect(aRect);
    ActivateL();
}

CHelloWorldContainer::~CHelloWorldContainer() {
    delete iLabel;
    delete iToDoLabel;
}
```

Sets control as ready to be drawn.

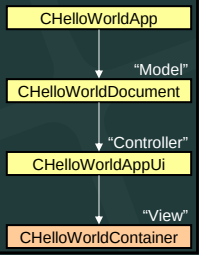


Michael Rohs
michael.rohs@telekom.de

Container Class Implementation (2/3)

```
void CHelloWorldContainer::SizeChanged()
{
    iLabel->SetExtent(TPoint(10,10), iLabel->MinimumSize());
    iToDoLabel->SetExtent(TPoint(10,100), iToDoLabel->MinimumSize());
}

TInt CHelloWorldContainer::CountComponentControls() const {
    return 2;
}

CCoeControl* CHelloWorldContainer::
ComponentControl(TInt aIndex) const
{
    switch (aIndex) {
    case 0: return iLabel;
    case 1: return iToDoLabel;
    default: return NULL;
    }
}
```

All methods called by the framework



Michael Rohs
michael.rohs@telekom.de

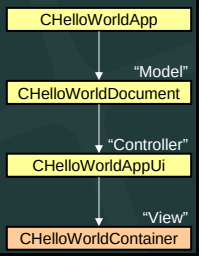
Container Class Implementation (3/3)

```
void CHelloWorldContainer::Draw(const TRect& aRect) const
{
    CWindowGc& gc = SystemGc();
    gc.SetPenStyle(CGraphicsContext::ENullPen);
    gc.SetBrushColor(KRgbGray);
    gc.SetBrushStyle(CGraphicsContext::ESolidBrush);
    gc.DrawRect(aRect);
}
```

Called by the framework

Redraw Events

- system-initiated redraws
  - framework calls Draw()
- application-initiated redraws
  - application calls DrawNow() or DrawDeferred()



Michael Rohs
michael.rohs@telekom.de

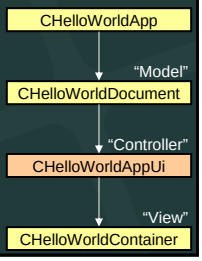
Handling Key Events in the Container Class

Window server routes key events to AppUi

Control stack controls order of key event delivery

Overwrite CCoeControl::OfferKeyEventL() for key handling

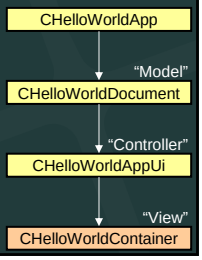
```
void CHelloWorldAppUi::ConstructL()
{
    BaseConstructL();
    iAppContainer = new (ELeave) CHelloWorldContainer;
    iAppContainer->SetMopParent(this);
    iAppContainer->ConstructL(ClientRect());
    AddToStackL(iAppContainer);
}
```



Michael Rohs
michael.rohs@telekom.de

Handling Key Events in the Container Class

```
TKeyResponse CHelloWorldContainer::OfferKeyEventL(
    const TKeyEvent& aKeyEvent, TEventCode aType)
{
   if (aType == EEventKey) {
        TBuf<8> text;
        text.AppendNum(aKeyEvent.iCode);
        iToDoLabel->SetTextL(text);
        DrawNow();
        return EKeyWasConsumed;
   }
   return EKeyWasNotConsumed;
}
```



Michael Rohs
michael.rohs@telekom.de

Adding a Menu Item to the Options Menu

HelloWorld.hrh:

```
enum THelloWorldCommandIds {
    EHelloWorldCmdAppTest = 1, EHelloWorldCmdNewColor = 2
};
```

HelloWorld.rss:

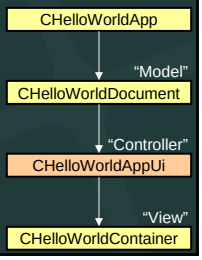
```
RESOURCE MENU_PANE r_helloworld_menu
{
  items =
  {
    MENU_ITEM { command = EHelloWorldCmdNewColor; txt = "New color";},
    MENU_ITEM { command = EHelloWorldCmdAppTest; txt = "Test"; },
    MENU_ITEM { command = EAknCmdExit; txt = "Exit"; }
  };
}
```

Michael Rohs
michael.rohs@telekom.de

Adding a Menu Item to the Options Menu

HelloWorldAppUi.cpp:

```
void CHelloWorldAppUi::HandleCommandL(TInt aCommand) {
    switch (aCommand) {
        case EHelloWorldCmdNewColor:
            iAppContainer->NewColor();
            break;
        ...
    }
}
```



Michael Rohs
michael.rohs@telekom.de

Outline

- Symbian overview
- Symbian OS structure
- Setting up a development environment
- Creating a "Hello World" example
- Basic Symbian OS concepts
- Application framework

Michael Rohs
michael.rohs@telekom.de

References

Ansgar Gerlicher, Stefan Rupp: Symbian OS – Eine Einführung in die Anwendungsentwicklung, dpunkt-Verlag, 2004

Martin Tasker: Professional Symbian Programming, Wrox Press Ltd., 2000

Leigh Edwards, Richard Berker, et al.: Developing Series 60 Applications – A Guide for Symbian OS Developers, Addison-Wesley, 2004

Jo Stichbury: Symbian OS Explained – Effective C++ Programming for Smartphones, Wiley, 2004

Richard Harrison: Symbian OS C++ for Mobile Phones, 2 Volumes, Wiley, 2003/2004

More: www.symbian.com/books/index.html

Michael Rohs
michael.rohs@telekom.de

Thank you for your endurance!