CCRMA Mobile Music Making Workshop

Georg Essl
Deutsche Telekom Laboratories
TU-Berlin

Overview.

- Setup
  - Mobile Phones
  - Mobile OS
  - Carbide
  - Mobile Programming in C++
  - Preparing for installation: Key and security

Making Friends.
Connection phone and laptop.

- In order to be able to collect the mobile phone with the laptop we need to do two things.
  - Install Nokia PC Suite
  - Pair the phone with the laptop (bluetooth stack dependent)

Hello World.

- New->Symbian OS C++ Project->3rd ed. GUI Application
- HelloWorld and check path!

Yello Yorld.

- src
  - HelloWorld.cpp
  - HelloWorldApplication.cpp
  - HelloWorldAppUi.cpp (Menu UI)
  - HelloWorldAppView.cpp (Display Stuff, draw)
  - HelloWorldDocument.cpp
- Inc
  - HelloWorld.hrh (Contains menu structure definitions)
  - HelloWorld.pan
  - HelloWorldApplication.h
  - HelloWorldAppUi.h (Menu UI)
  - HelloWorldAppView.h (Display Stuff, draw)
  - HelloWorldDocument.h

Yello Yorld.

- sis
  - HelloWorld.pkg (install packaging)
  - HelloWorld.sis (program, yay)
- group
  - HelloWorld.mmp (link and dependancy file)
  - Bld.inf (overall project build definition, don’t change)
- data
  - HelloWorld.rls (localization strings)
  - HelloWorld_reg.rss (registration resource file, leave alone)
  - HelloWorld.rss (menu structures)

Jello Swirls.
Compiling.

- First: Project->Properties
- Carbide Build Configuration
- Configuration: Phone Release (GCCE) [S60_3rd_MR]
- PKG File: sis/HelloWorld_S60_3_X_v_1_0_0.pkg
- Ignore next two lines!

Jello Swirls.
Compiling.

- Signed Sis File Name: HelloWorldme.sis
- Certificate: browse to your certificate *.cer
- Key: browse to your key *.key
- Password: (your password) or password

Jello Swirls.
Compiling.

- Build Selected Target, or Build Target Only
- Finding Errors
  - Check Problems tab and scroll up
  - Check Console tab, ctrl-f and search for “error”
- Typical error: pkg doesn’t pick right path.

Jello Swirls.
Compiling.

- If all went right you should have a:
- sis\HelloWorldme.sis

- Install on phone using Nokia PC Suite (install both on PC and phone)
- Connect via USB cable (preferable) or bluetooth.
- Double-click sis-file to install and answer dialogues on both PC and phone.
- Click “yes!” like a madman.

Fellow Pearls.
Tinker Time.

- Add display stuff: Draw()

```
    TRect bRect = aRect;

    CWindowGc& gc = SystemGc();
    gc.SetBrushColor( KRgbGray );
    gc.SetPenStyle(CGraphicsContext::ENullPen);
    gc.SetBrushStyle(CGraphicsContext::ESolidBrush);
    gc.DrawRect(bRect);
    const CFont* font = CEikonEnv::Static()->TitleFont();
    gc.UseFont(font);
    gc.SetPenStyle(CGraphicsContext::ESolidPen);
    gc.SetPenColor(TRgb(0,0,0));

    TSize aSize(bRect.Width(),font->HeightInPixels()+5);
    bRect.SetRect(TPoint(0,(bRect.Height()/2)-font->HeightInPixels()-5),aSize);
    gc.DrawText(_L("Fellow Pearls for real! "),bRect,font->AscentInPixels(),CGraphicsContext::ECenter);
```

Help Debug.
Too cheap to pay NokiaPro.

- Generic Debug util:
- Utilities/src/Debug.cpp Write debug info to files.
- static void WriteInt(const TInt aInteger);
- static void WriteReal(const TReal aReal);
- static void WriteInfo(const TDesC& aInformation);
- *N version adds newline

Help Debug.
Too cheap to pay NokiaPro.

- Writes file to phone. Check Debug.cpp and phone build for location. Use Nokia PC Suite.

MobileSTK.
Install.

- Take MobileSTK.zip and install to D:\Dev\Work\Symbian-9.1\MobileStk3
- Run carbide
- Import…->Symbian bld.inf
- Find MobileStk3/group/bld.inf
- Check project settings as before
- Build

MobileSTK.
Core Parts.

- StkCore.cpp (Core part, implements Audio layer and accesses instruments)
- STKFPAppui.cpp (Menu)
- STKFpAppview.cpp (Visualization)

- Most files are almost unchanged STK 4.2.2

MobileStk.
Linking etc.

- Check out group/STKFP.mmp

StkCore.

- CSTKCore::CSTKCore() – Place to construct your instruments

- .h: Shakers *shakers;
- .cpp: shakers = new (ELeave) Shakers();

StkCore.

- CSTKCore::~CSTKCore() – Place to destruct your instruments

- .cpp: delete shakers;

StkCore.

- Put your excitations in:

- void CSTKCore::Excite()

- case 8: // This is for selection
- shakers->noteOn(0,excitation_amp);
- break;

StkCore.

- Put your ticks in:

- TUint8 *CSTKCore::FillBuffer()

- case 8:
- aData[i]= StkFloat(127)*shakers->tick();
- break;

Note that you need to scale float tick results to 8-bit signed integers (multiply 127).

StkCore.

- Warning: Sample-based STK instruments require installing the sample files on the phone. (We won’t do this!)

- OK lets pick a sample-free instrument and add it to StkCore (or better replace an existing one, for example case 3)

Symbian Specifics.

- Symbmath.h – contains de-templated array classes and optional substitues for math.h. You should never have to touch this, unless you want to add new arrays.

- StdTemplates are not yet supported (I think)

Symbian Specifics.

- Stk.h contains all main global definitions of STK.
- It contains most Symbian sensitive aspects of the port.

- #if defined(SYMBIAN)
- <symbian specific stuff here>

Fixed Point Code.

- Fixedpoint.h
- Stk.h:
- #undef USEFIXEDPOINT
- ->
- #define USEFIXEDPOINT

- Only for true hackers (yes you!)
- StkFloat (turns into fake floats)
- xStkFloat (forces real floats)

Symbian Specifics.

- Main porting and programming consideration:
- No global stack!
  - No static
  - No globally defined variable outside classes

- You may find changes to original STK to remove statics and globals.
- Won’t link (compile-time error).

Symbian Specifics.

- Main porting and programming consideration:
- Multithreading is supported but discouraged. Context switches costly.
- Current MobileSTK avoids multithreading and does not port thread-related stuff of STK.

Sampling Rate.

- Stk.h
- #define DO8kHz

- Other possibilities:
- #define DO22kHz for 22050
- #undef DO8kHz for 44100

- Tradeoff:
  - Slow frame rate means less data to compute per buffer update -> easy on computation, high latency
  - Fast frame rate means lots of data to compute per buffer update -> stutter, lower latency

MobileSTK.

- Tinker time.

Integrating MobileSTK.

- If you want to integrate MobileSTK with other programs, core interface is via StkCore:

```
void PlayL();   - start playback stream
void StopL();   - stop playback stream
void Excite ();  - excite currently active STK instrument
int setInstrument(int);   - set currenty active STK instrument
void setAmplitude(StkFloat);  - set excitation amplitude
void setPitch(int);      - set excitation pitch

void Record();    - start recording (mic access)
bool isRecording();   - check if recording is going on
```

Shake Fun.
Connect.

- One at a time, turn on and write down shake#! We don’t wanna go mess with each others sensors.
- Pair shake with laptop!

Shake Fun.

- Check your local bluetooth serial port (if you don’t have one or your bluetooth doesn’t work properly, find a buddy [Bluetooth debugging is a real pain! We won’t even try!]
- Start connection to shake unit via bluetooth stack (depends on stack, we’ll try to get that to work for some at least), write down com port!
- Start->Run->hypertrm
- Name: shake
- Bits per second: 230400
- (8, none, stopbit 1, hardware, just leave these)

Shake Fun.

- Unzip Shake Useful Config zip and look at the filenames.
- Transfer->Send Text File->[pick some, like OutputAll.txt]

Shake Calibration.

- Check Shake manual for steps to do calibration.
- Magnetometer needs calibration for compass use.
- Accelerometers tend to be well-calibrated by default.

Shake Fun.

- Shake user group:
- http://www.dcs.gla.ac.uk/research/shake/

5500 Accel Code.

- ShaMus: SensorTestAppView.cpp
- void CSensorTestAppView::AccelerometerUpdate(TInt aX, TInt aY, TInt aZ)
- {
- /* Mess around here */
- }

- (No separate utility source yet, but planned)

Shake Code.

- Shake Code acts like a BTObserver that gives you sensor data.

- class MShakeObserver
- {
- public:
- virtual void ShakeUpdate(TInt aCompassHeading, TInt aInclination) = 0;
- };

- Hack your own observer if you want other data in
- ShakeSensor.cpp/h

Shake Gestures.

```
        switch(playmode)
        {
        case PLAY_SIMPLESTRIKE:
        if(iX > 1 && oldiX < 1 && iCalibrateCount > KCalibrateCountEnd)
        {
                mystk.setInstrument(6);
                // The below doesn't work as the thing is too slow.
#ifdef USESHAKE
                mystk.setPitch((iA+1800)/9.0+24.0);
#else
                mystk.setPitch((iY+800)/3.0+24.0);
#endif

                mystk.Excite();
        }
        break;
```

BlueTooth Stuff.

- Instantiate Bluetooth code and make your stuff an observer.

- See ShaMus code for example.
- Core Bluetooth code in
- Utilities/inc/BTObserver.h (interface/callback)
- BTClient.cpp/h
- Unidirectional for now. If you need bidirectional, email me and I can get you bidirectional.