Mobile STK for Symbian OS

Georg Essl, Michael Rohs
Deutsche Telekom Laboratories, TU-Berlin
{georg.essl,michael.rohs}@telekom.de

Abstract

Mobile audio-enabled technology is a commodity by now. However access to these technologies for audio research, algorithmic synthesis and interactive performance is currently limited. We describe an effort to port the Synthesis Toolkit (STK) developed by Perry Cook and Gary Scavone to the Symbian operating system, which is used by a number of mobile phone manufacturers.

1 Introduction

Mobile phones and mobile audio playback devices, like mp3 players are now a mass market commodity. All of these devices have audio playback capacities inherent in their design, yet most of this capacity is either used for very basic audio functions like streamed speech communication, stream or file audio playback, either for passive entertainment or for alerting functions.

In order to support an on-going project of using mobile phones to interactive performance (Rohs, Essl, and Roth 2006), we sought to bring algorithmic sound synthesis methods to these mobile technologies.

Currently there are two main operating systems for mobile devices in wide use. These are the Symbian operating system and Windows CE (which under various revisions is also called Pocket PC or Windows Mobile). While Windows CE is prominent among personal digital assistants (PDA), Symbian OS is widely used in mobile phones. Our current focus is on mobile phone technologies, hence prioritizing the Symbian OS portation.

The Synthesis Toolkit (STK) by Perry Cook and Gary Scavone (Cook and Scavone 1999; Cook 2002; Scavone and Cook 2005) is a cross-platform set of audio signal processing and algorithmic synthesis classes. Its stated goal is "to facilitate rapid development of music synthesis and audio processing software" (Scavone and Cook 2005). STK has already been ported to various environments, including Max/MSP and PD (Trueman and DuBois 2000), ChucK (Wang and Cook 2003) and SuperCollider (Lansky 2006).

At current, sophisticated synthesis algorithm implementations for mobile operating systems are lacking, and STK provides an attractive choice for filling this gap due to its design with an eye towards cross-platform portability and due to its rich set of parametric synthesis methods. Our current port is based on STK 4.2.1.

We plan to also port STK for Windows CE, however this should be less technically involved, as Windows CE supports Direct X and shares the basic architecture of the already supported Windows environment.

2 Architectural Limitations of Symbian OS

One of the main obstacles to porting STK to Symbian OS was the lack of global static memory. This means that applications are barred from sharing information through the mechanism of global static variables in C++ (Stichbury 2004).

In STK global variables are used in a few contexts:

- The first one is to store the currently active sampling rate, which is then accessed through `Stk::sampleRate()`. Making this information local would have meant adding substantial amount of argument passing to most STK classes. Hence we opted to solve this problem by making the sampling rate a global compile-time defined constant again, called `SRATE`. This mechanism used to be in place in earlier versions of STK.

- The second instance is error message support. The mechanisms to give debug messages are rather limited and different from desktop devices. Hence we have decided to disable this class altogether.

- Shaker index number was stored in a local static variable. It was made a member variable of tt Shakers serving the same purpose.

3 Audio Support of Symbian OS

Symbian OS offers various ways to play audio data (Nokia 2005b; Nokia 2005a). For the purpose of porting STK, the streaming of data buffers is needed. Symbian OS provides such streaming capability, which in its interface architecture closely mimicks the interface of RTAudio (Scavone and Cook 2005).

3.1 Streaming Audio Output

The class responsible for real-time audio output streaming is called `CMdaAudioOutputStream` and it needs to be used from a class which implements the `MMdaAudioOutputStreamCallback` interface. The callback interface will be called when the playback stream enters various states, including (1) finishing opening the stream, (2) when the latest data buffer has been copied to the playback hardware, and (3) when the stream closes. The second callback closely corresponds to the callback that RTAudio uses to pull data for playback.

A client class using Mobile STK for Symbian can call sample generating functions of STK classes (the so-called "tick"-methods (Scavone and Cook 2005)) or combination of such classes. We implemented double buffering as Symbian does not copy the user supplied buffer nor does it lock the memory for write access while playing back. Hence ticking will fill one buffer after the one currently being copied, introducing an additional latency of one buffer length on top of the latency introduced by the buffer copying and inherent audio streaming implementation.

3.2 Prospects for RTAudio and RTMidi ports

Gary Scavone's RTAudio and RTMidi (Scavone 2002; Scavone and Cook 2005) is closely tied with STK, though in principle stand-alone. At the current time the port for Symbian OS does not include a port of RTAudio and RTMidi. However at least a partial port of this project into RTAudio is planned to make the audio interfacing that now is available for STK also independently available for other audio application efforts while keeping the unified interface that RTAudio provides.

The authors are unaware of any availability of a hardware MIDI interface for Symbian-enabled mobile phones, hence there are no plans to port RTMidi.

4 Supported features of STK

All signal processing classes of STK have been fully ported. Also all instrument type classes have been ported. This also includes instruments which utilize sound files. In addition all sound file input and output classes have been ported.

4.1 Modifications to standard STK

Version 4.2.1 of STK assumes ANSI-C/POSIX type environment but also uses additional C++ standard library features like vector templates, `std::string` or `iostream`. While Symbian features ANSI-C/POSIX features, the C++ standard libraries are missing.

In standard STK `std::string` is used for file names and for error message passing. We have changed file name string to `const char *`. We have exluded also error message display because it uses `iostream`, which is again a C++ standard library feature and hence does not port easily.

Vector templates are used in many classes to provide structured dynamic array allocation and access in standard STK. These templates do not exist in Symbian C++. To replace the template instantiations of the array, custom vector classes with the same interface have been implemented. These and other Symbian specific porting functions are implemented in `symbmath.h` and `symbmath.cpp`.

The specific classes substituting standard vector templates are:

- `FloatVector`: Float vectors.
- `IntVector`: Integer vectors.
- `ADSRVector`: Vector of pointers to ADSR instantiations.
- `WaveLoopVector`: Vector of pointers to WaveLoop instantiations.
- `FileWvInVector`: Vector of pointers to FileWvIn instantiations.
- `GrainVector`: Vectors to Grain class instantiations. `GrainState` and `Grain` structure definition had to be moved to allow the global definition of this class.

4.2 Exclusions from the Standard STK

A number of classes have been explicitly excluded from being ported to Mobile STK for Symbian. The reason is simply that some class functionality provided by by STK assumes a ANSI-C/POSIX-style desktop computer oriented environment. Primary examples of such functionality are socket connectivity functions. In STK they are used to allow STK applications to interface with other applications streaming the STK native MIDI-like language SKINI. In demo applications

this is used to interface with graphical user interfaces implemented in Tcl/tk. Tcl/tk is not currently available for Symbian OS hence there was no current need to port this functionality.

While multi-threading does exist in principle in Symbian OS it is generally not recommended for performance reasons. Hence we have replaced STKs multi-threading support with a single application framework.

Moreover currently `MidiFileIn` support and SKINI support is not ported. A port of these two features is planned.

Finally, support for realtime sound input and full-duplex input/output streaming is not yet supported. Support for realtime sound input is planned. If it is currently possible to implement full-duplex input/output streaming is currently being investigated.

5 Embedding STK in a Symbian application

A typical Symbian application will consist of a number of skeleton classes that, when fully implemented, constitute a running application. In our port these classes are called `STKapp.cpp`, `STKappui.cpp`, `STKcontainer.cpp` and `STKdocument.cpp`. Only `STKappui.cpp` and `STKcontainer.cpp` have been modified to give a core idea how to write STK applications in Symbian OS. `STKappui.cpp` allows one to define user interface features like pull-up menues and handle user generated events like selecting a menu item. In our core implementation we allow for selecting various instruments and for triggering NoteOn events. More sophisticated interactions can be in principle incorporated into this file. The core of the implementation can be found in `STKcontainer.cpp`. STKcontainer contains both the streaming audio playback functionality as well as the basic interface to connecting this playback functionality with STK synthesis classes. The core methods provided are `CSTKContainer::Excite()` which is used to trigger excitation events of active synthesis classes, and `CSTKContainer::FillBuffer()` which should call the tick method of active synthesis classes. The `FillBuffer()` method also contains the core of the double buffering implementation. A simple example using the `Saxofony` class can be seen below. An instance of the class has been created in the constructor like this `sax = new (ELeave) Saxony(220.0);`:

```
void CSTKContainer::Excite()
{ // NoteOn for a Saxofony
  sax->noteOn(220.0,1.0);
}

TUint8 *CSTKContainer::FillBuffer()
{
  int i=0,j=0;
  TReal in,out;

  TUint8 *aData;
  if(dBuffer==0) { // Double Buffer
    aData = aData1;
    dBuffer = 1;
  }
  else {
    aData = aData2;
    dBuffer = 0;
  }

  for(i=0;i<KBufferSize; i++)
  { // Fill Buffer with Saxofony samples
    aData[i]= 127 * sax->tick();
  } // Note: Scaled to 8 bit resolution
  return aData;
}
```

6 Performance Indicators

We used a Nokia 6630 mobile phone to design and test the Mobile STK for Symbian. In order to assess the processing capability of the phones we ran performance tests exciting increasing numbers of `Plucked` instances, and also `SineWave` instances. At 8000 Hz and with a buffer size of 1024, 5 plucked strings and 9 sinewave oscillators could be computed without introducing distortions. Increasing the buffer size to 2048 and 8192 did not change this behavior hence we conclude that the bottle-neck is purely of computational power. Also, doubling the sampling rate to 16000 Hz does reduce the number of simultaneous Plucked instances to 2 and sinewave to 4, which is half the number, again indicating that pure computational power is the limit to more parallel sources.

One likely core reason for the performance limits with these phones is their lack of the hardware floating point unit. Personal experience of the second author indicates that floating point operations lead to a significant deterioration of performance. For this reason we want to investigate fixed-point arithmatic substitutions to increase performance in the future.

It should be noted that the Nokia 6630 is designed as a mobile camera phone and not as an mp3 player. It is to be expected that mp3 players show better audio performance. Also performance of mobile phones can be expected to increase with technological advances.

7 Uses of Mobile STK for Symbian

We see the main application of Mobile STK for Symbian in interactive applications. In fact, the motivation to port STK to Symbian comes from a related project where Symbian-enabled camera-phones are used for interactive musical performance (Rohs, Essl, and Roth 2006). In the current form of this project the sound is generated externally on a computer which connects to the Symbian phone via Bluetooth.

Further mobile interactive entertainment in form of games may soon become very popular. Interactive audio support of high fidelity, but low memory and computational requirements may proof to be very helpful in this case. Perry Cook (Cook 2002) and others have long stressed the usefulness of parametric synthesis models for interactive applications like games and this argument certainly holds for mobile applications. In fact the argument may be stronger, as memory and computational power are more limited compared to desktop computers or special-purpose audio hardware.

A Appendix: Retrieving and installing Mobile STK for Symbian

Mobile STK for Symbian is available for download at `http://mobilestk.sourceforge.net`. It follows the same licensing spirit of the original STK[1], which essentially states that it is "mostly" free, specifically for academic purposes.

B Appendix: Not ported classes of the STK 4.2.1 distribution

Socket-related functionality:

- `InetWvIn, InetWvOut`
- `TcpClient, TcpServer`
- `UdpSocket`

Process/thread related functionality:

- `Mutex, Thread`

MidiFile, SKINI and message passing functionality:

- `MidiFileIn, Messager, Skini`

RTAudio/RTMidi functionality:

- `RTAudio, RTDuplex, RtWvIn, RtWvOut`
- `RTMidi`

Also none of the demo applications have been ported.

[1] `http://ccrma.stanford.edu/software/stk/information.html`

References

Cook, P. and G. Scavone (1999). The Synthesis ToolKit (STK). In *Proceedings of the International Computer Music Conference*, Beijing.

Cook, P. R. (2002). *Real Sound Synthesis for Interactive Applications*. A K Peters, Ltd.

Lansky, P. (2006). STK for SuperCollider. Available online at `http://www.music.princeton.edu/paul/stkugens.tar.gz`.

Nokia (2005a). S60 platform: Music application developer's guide v1.0. Available online at `http://www.forum.nokia.com/info/sw.nokia.com/id/44e77a3e-e26e-4a07-a22d-ec0708024b78/S60_Platform_Music_Application_Developers_Guide_v1_0_en.pdf.html`.

Nokia (2005b). Symbian OS: Creating Audio Applications in C++. Available online at `http://www.forum.nokia.com/info/sw.nokia.com/id/829fbb19-e7fb-4578-8365-6a66d74b2572/Symbian_OS_Creating_Audio_Applications_In_Cpp_v1_0_en.pdf.html`. Published by Nokia Cooperation.

Rohs, M., G. Essl, and M. Roth (2006). CaMus: Live Music Performance using Camera Phones and Visual Grid Tracking. submitted to the International Conference on New Interfaces for Musical Expression.

Scavone, G. and P. R. Cook (2005). RTMidi, RTAudio, and a Synthesis Toolkit (STK) UPdate. In *Proceedings of the International Computer Music Conference*, Barcelona, Spain.

Scavone, G. P. (2002). RtAudio: A Cross-Platform C++ Class for Realtime Audio Input/Output. In *Proceedings of the International Computer Music Conference*, Götborg, Sweden, pp. 196–199. ICMA.

Stichbury, J. (2004). *Symbian OS Explained*. Hoboken, NJ: Wiley.

Trueman, D. and L. DuBois (2000). PeRColate. Available online at `http://www.music.columbia.edu/PeRColate/`.

Wang, G. and P. R. Cook (2003). ChucK: A Concurrent, On-the-fly Audio Programming Language. In *Proceedings of the International Computer Music Conference (ICMC),*, Singapure.