A Context for Pen-Based Mathematical Computing

Elena Smirnova and Stephen Watt

Ontario Research Centre for Computer Algebra, University of Western Ontario

www.orcca.on.ca/PenMath

Maple Summer Conference 2005
Introduction

• Growing popularity of pen-enabled devices such as the *Pocket PC*, *Tablet PC* and *interactive whiteboards* implies the need for handwriting recognition tools, including not only text, but mathematics too.

• Math input on pen-enable devices goes way beyond ordinary hand-written math on paper or chalkboard, because it can enjoy rich functionality of the software standing behind ink-capturing hardware.

• This may provide pen-entered math with useful features
  o Editing
  o On-spot validation
  o Direct manipulation
# Specifics Of Pen-Based Math Approach

- **larger alphabet**
  
  $A, A, A, \tilde{A}, a, \alpha, \infty, \infty, \ldots$

- **no fixed vocabulary**

  $LambertW(k,x), WrightOmega(z), \ldots$
• larger alphabet
  A, A, A, Ñ, a, α, ∞, ∞,…

• no fixed vocabulary
  LambertW(k,x), WrightOmega(z),…

• 2-dimensional structures
  \[ x^n, \frac{p}{q}, \sum_{i}^{N} a_i, \begin{bmatrix} a & b \\ c & d \end{bmatrix} \]

• large symbols for grouping
  \[ \langle \alpha \rangle, \langle \beta \rangle, \langle \gamma \rangle, \sqrt{x^2 + y^2 + z^2} \]
Specifics Of Pen-Based Math Approach

- larger alphabet
  A, A, A, Ñ, a, α, ∞, ∞,…

- no fixed vocabulary
  LambertW(k,x), WrightOmega(z),…

- 2-dimensional structures
  \[ x^n, \frac{p}{q}, \sum_{i=1}^{N} a_i, \begin{bmatrix} a & b \\ c & d \end{bmatrix} \]

- large symbols for grouping
  \[ \langle \alpha \rangle, \sqrt{x^2 + y^2 + z^2} \]

- multiple notations
  \[ C_n^m \text{ vs. } \binom{m}{n}, \tan x \text{ vs. } \tan x \]

- ambiguous notations
  \[ \sin^{-1}x \text{ (arcsin } x \text{ or } \frac{1}{\sin x} \text{)} \]
  \[ \lg x \text{ (log}_2 x, \log_{10} x \text{ or ln } x) \]
• The above issues require a new approach for pen-based software solutions for handling handwritten mathematics.

• Ultimately we wish to have a pen-based platform for:
  - mathematical expression entry,
  - mathematical editing and
  - calculation.

\[
\frac{LambertW(z)}{z} = e^{-x}
\]
• The above issues require a new approach for pen-based software solutions for handling handwritten mathematics.

• Ultimately we wish to have a pen-based platform for
  o mathematical expression entry,
  o mathematical editing and
  o calculation.

\[ \text{LambertW}(z) = e^{-x} \cdot z, \quad z \neq 0 \]
Interface for Pen-based Mathematics

Math Expression Recognizer

Math Expression Renderer

Math Expression Validation Tool

Math Expression Manipulation Tool

\[ \mathcal{J}_{nk} \]

\[ f(\omega) d\omega \]

mathematical back engine
In this talk

- **We will not**

  address the subject of developing specific software for ink-aware math application

- **We will**

  - Investigate the topic of an *interface* to pen-enable math software
  - Suggest an architectural solution to enable such an interface.
Question we explore:
If a pen-based interface for math is widely acceptable, how should its architecture be organized?

Key to the decision:
Define the target audience that will use this interface:
- We do not restrict the audience only to math systems users
- We also include uses of rich text editors and document processors
- We do not restrict the audience to one hardware/software platform
The remainder of this presentation is organized as follows:

- **Overview of existing Ink technologies**
- **Portability objectives for pen-based frameworks**
- **The large-scale aspects of the architecture designed**
- **Some of the implementation issues**
- **Report on current and future work**
Recently both math software packages and document processing applications have started to comprise ink-enabled features.

Maple 10 and Word 2003 are good examples of software with basic pen-aware features.
Ink Features In Maple 10 and MS Word 2003

overlay inking and ink comments in MS Word 2003

scratchpad and character selector in Maple
In both cases the attempts to enable ink are extremely handy for brief handwritten notes.

- Both solutions are specific to the software product: they cannot be easily exported and reused in other applications
- Neither provides full ink support for handwritten mathematics
Available Technologies

• Digitizer Device Drivers (such as WACOM)
  
  + easy to use interface (C++)
  + accessible from Java through JNI adaptors

  - hardware-specific
  - provides too primitive ink handling functionality
Available Technologies (2)

- .NET / C#
  
  + fully compatible with Tablet SDK API
  + native to Windows platforms

  - cannot be exported as an ActiveX control to run inside MS Office applications
  - is not portable across platforms
  - cannot be directly use within Maple architecture
Available Technologies (3)

- Tablet PC SDK
  - provides high-level support for ink management on Tablet PC
  - supported by .NET framework
  - is not portable across platforms
  - not directly available from Java
  - not available from Maple
Available Technologies (4)

- Maple

  - An interface for pen-based mathematics will be required to perform non-trivial transformations on its input.
  - It is inevitable that a pen-based mathematical framework will make use of a computer algebra system.
  - For this we find Maple to be a suitable choice.
Combining Available Technologies

- Even though these technologies individually provide high-levels of functionality useful in ink applications, none is completely suitable for our needs.

- In particular these elements to not all work together.

- Our solution will need to combine them in such a way that the final architecture
  - can provide high-quality ink capabilities
  - remains portable across platforms
  - provides easy connection with applications.
Our approach must meet the requirement of **portability**

- **two-dimensional platform portability** of pen-based interface frameworks:
  - across platforms and applications
  - *over time* for any given (evolving) platform/application
Portability Criteria

Our approach must meet the requirement of **portability**

- **two-dimensional platform portability** of pen-based interface frameworks:
  - across platforms and applications
  - over time for any given (evolving) platform/application

- **digital ink portability**
  - can be achieved with InkML (universal ink format)
  - wrappers for device-specific ink interfaces
Portability Criteria

Our approach must meet the requirement of **portability**

- **two-dimensional platform portability** of pen-based interface frameworks:
  - across platforms and applications
  - over time for any given (evolving) platform/application

- **digital ink portability**
  - can be achieved with InkML (universal ink format)
  - wrappers for device-specific ink interfaces

- **mathematical data portability**
  - OpenMath
  - MathML
Our Architectural Approach

Invariant Solution With Replaceable “Glue”

- Parts remaining invariant:
  A High-level math object manipulation code
  B Low-level digital ink analysis code

- Parts, depending on hosting system:
  ① Basic ink collecting software
    - to support abstract ink representation
  ② “Glue”: Inter-Component Interface
    - to link (A) and (B) with (1) and (3),
  ③ Interface code
    - to embed pen-based math input in hosting application
Framework Components

③ Interface to Host Application

② “Glue”: Inter-Component Interface

① Basic Ink collecting software

A
High-level math object manipulation code
Java

B
Low-level digital ink analysis code
C++
Implementation Languages

- **C#**
  - **assignment**: ink collecting and processing,
  - **example of use**: connecting to Tablet SDK

- **C++**
  - **assignment**: low-level intensive computations
  - **example of use**: character recognizer, glyph feature determiner

- **Java**
  - **assignment**: high-level code for connecting with mathematical engine
  - **example of use**: math expression manipulation
Instantiating The Architecture

We have instantiated the architecture for Tablet PC as follows:

① For **basic ink software**
   we used .NET-based Tablet PC SDK

② Specially designed **linkage mechanism** included
   - a number of .NET technologies (C#, managed C++),
   - COM interoperability features and
   - Java Native Interface (as described further)

③ **Interface to the hosting application**
   vary depending on the application
To test the Tablet PC version of our architecture we use
  - for mathematical computing: *Waterloo Maple*
  - for document processing: *Microsoft Word*

Then our framework components look like
Linkage For The Test Framework

1: TABLET PC SDK

2: Ink Component

A: JAVA

B: C++

INTERFACE TO HOST APP.

JNI

JNI + JAWT NI + COM Interop

COM Interop + PInvoke

.NET
A solution for the host interface is as follows:

- **interface to Maple**
  - Java library, accessing .NET control through JNI

- **interface to MS Word**
  - ActiveX control, accessing .NET control via Win 32 C++ Wrapper
Example:
Suppose again that we have two tasks, $T_1$ and $T_2$, with

$$p_i(t) = a_i \lambda_i e^{-\lambda_i t}$$
$$q_i(t) = (1 - a_i) \lambda_i e^{-\lambda_i t}$$

For $0 \leq a_i \leq 1$, $\lambda_i > 0$.

For both tasks let the time allotment function be

$$\nu_i(t) = \frac{t}{2}$$

Then we have

$$P_i(t) = a_i \left(1 - e^{-\left(-\frac{1}{2} \lambda_i t\right)}\right)$$

$$Q_i(t) = (1 - a_i) \left(1 - e^{-\left(-\frac{1}{2} \lambda_i t\right)}\right)$$

Which implies

$$P_A(t) + Q_A(t) = 1 - \left(1 - a_1\right) \left(1 - e^{-\left(-\frac{1}{2} \lambda_2 t\right)}\right) - \left(1 - a_2\right) e^{-\left(-\frac{1}{2} \lambda_1 t\right)}$$

$$+ \left(1 - a_1 - a_2\right) e^{\left(t \cdot \left(\frac{1}{2} \lambda_1 + \frac{1}{2} \lambda_2\right)\right)}$$
Example:

Suppose again that we have two tasks, $T_1$ and $T_2$, with

$$p_i(t) = a_i \cdot \lambda_i^{\frac{A_t}{\tau}} \cdot q_i(t) = (1 - a_i) \cdot \lambda_i^{\frac{A_t}{\tau}}$$

for $0 \leq a_i \leq 1$, $\lambda_i > 0$. For both tasks let the time allotment function be $v_i(t) = t/2$.

Then we have

$$P_i(t) = a_i \cdot (1 - e^{-\lambda_i t/2})$$  
$$Q_i(t) = (1 - a_i) \cdot (1 - e^{\lambda_i t/2})$$

Which implies

$$P_i(t) + Q_i(t) = 1 - (1 - a_i) e^{-\lambda_i t/2} - (1 - a_2) e^{-\lambda_2 t/2} + (1 - a_1 - a_2) e^{(\lambda_1 + \lambda_2) t/2}$$

Using (3.17), the above expression yields

$$\langle t \rangle_{par} = 2 \left[ \frac{1 - a_2}{\lambda_1} + \frac{1 - a_1}{\lambda_2} - \frac{1 - (a_1 + a_2)}{\lambda_1 + \lambda_2} \right]$$
Current results

- We have developed a software solution to enable a pen-based math interface on Tablet PC platforms.
- Successfully plugged a recognizer for a wide variety of math characters to the framework.
- We tested its compatibility with Maple 10, MS Office (2000, XP and 2003).
Future Work

- Ongoing work in
  - plugging tools to determine structures of math expression
  - enabling math engine features
    - to validate math expressions
    - to allow direct manipulation on math formulae
  - instantiating our solution on other platforms
    - for handheld devices
    - other operating systems.
Ink Component

\[
\begin{align*}
\text{Ink} & \frac{2}{3} \quad \sqrt{2} \\
\text{Ink} & \frac{2}{3} - \sqrt{2}
\end{align*}
\]

Application

Ink-Enabled Platform

Math Expression

Render Tool

Math Expression

Direct Manipulation Tool

Glyph Feature Detector

Math characters samples DB

Math Char Candidates Selector

Math Expressions Pattern Matching

Math expressions patterns DB

Math Expression Candidates Selector

Legend

- C++
- JAVA
- DATA

Character Recognizer

abstract ink

ink

ink + reco

ink + reco

ink + reco

ink + reco

ink + reco

ink + reco

ink + reco

ink + reco

ink + reco

ink + reco

ink + reco

ink + reco

ink + reco

ink + reco

MathML (C+P)

MathML

MathML

MathML

MathML

MathML

MAPLE Math Engine

Le

nend

DATA

JAVA

C++
Conclusions

• Our goal was
  o design a framework to allow wide use of pen-based math interfaces
Conclusions

• Our goal was
  o design a framework to allow wide use of pen-based math interfaces

• Our requirements were that
  o these interfaces be suitable for both math computing packages and document processing applications
  o the framework
    ▪ provides high-quality ink capturing and handling
    ▪ allows easy access to mathematical engine
    ▪ ensures future portability across and along platforms and applications
Conclusions

- **Our goal** was
  - design a framework to allow wide use of pen-based math interfaces

- **Our requirements** were that
  - these interfaces be suitable for both math computing packages and document processing applications
  - the framework
    - provides high-quality ink capturing and handling
    - allows easy access to mathematical engine
    - ensures future portability across and along platforms and applications

- **Our believes** are
  - we have achieved these objectives
  - our solution can allow a more natural interface for mathematics in a variety of settings