Qt Centered Teaching Of Usability Theory And GUI Design At The Texas State University

Dan Tamir
Department of Computer Science
Texas State University-San Marcos
Academic Activities

- Florida Tech
  - Assistant / Associate Professor
- Texas State
  - Assistant / Associate Professor
- TAU, BGU, Haifa, FSU, FAMU
  - Adjunct Professor
High Tech

- **Motorola New Enterprises – Consultant**
  - Introduced video processing instructions to a new 64-processor SIMD processor

- **Motorola Semiconductors / Freescale**
  - Israel / Atlanta / Austin
  - Video compression
  - Developed next generation DSP core (StarCore)
    - Introduced application specific instructions
  - Introduced new DSP instructions to an embedded PPC Signal Processing Unit
Teaching

- Computer graphics / Graphical User Interface
- Multi-media / Image processing / Data compression
- Digital logic / Assembly language / Computer architecture
Research Projects

- Image processing / data compression
  - TX-DOT - Automating bridge inspection
  - NSF MRI - Laser lithography research project

- Computer Architecture
  - NSF TUES HPC - Curriculum Development
  - Semiconductor Research Institute  Power aware scheduling research project

- Usability
  - Emerson - Pinpoint analysis project
Extra Curricular

- NASA – KSC Fellow
  - A system for automatic inspection of the space shuttle tiles
- Member of the Israel delegation to the MPEG committee
  - Active member of the performance evaluation sub-committee
The Graphical User Interface Graduate level Course

1. Usability, Usability, Usability
   - Teaching usability to CS students

2. Platform/Applications
   - Stand alone, Web based, mobile,
   - OS, language, library

3. Practice
   - Teaching Qt
   - Projects and assignments
1. Special topics
   - Computer graphics,
   - Visualization,
   - Animation,
   - Networking / Multithreading
   - Time, emphasis, libraries (e.g., OpenGL)
Measuring Usability
Observations

- Usability is inversely proportional to effort
  - User effort is related to manual effort – e.g., number of mouse clicks, number of key-board clicks, mouse path traversed.

- A set of identical independent (“iid”) experiments on a single scenario can be used to measure learnability and operability

- Eye tracking can be used to provide additional measures of physical and manual effort
Time-on-Task = $\alpha X^\beta + \epsilon$
Effort–based Usability Model

*Based on ISO/IEC 9126–1:2001 Standard*
Eye Tracker Hardware

- Eye tracker
- Chin rest
Fixations and Saccades

- When performing a task, fixations and saccades can reflect effort expended.
  - Greater effort =
    - Longer fixation duration
    - More fixations
    - Longer saccade length
    - More saccades
Measurements

- Time on Task
- Number of Mouse/Keyboard clicks
- Total mouse path traversed
- Average fixation duration
- Average pupil diameter
- Number of fixations
- Average saccade amplitude
- Number of saccades
- Total mouse path traversed
Research Projects

- Evaluation of Web-based travel reservation systems
- Evaluation of Usability of two versions of Emerson Process Control software
- Pinpoint analysis using Emerson software modules
Travel Reservation Experiment
Time on Task

System A

System B

Power (System A)

$y = 622.65x^{-0.217}$

$R^2 = 0.7221$

Power (System B)

$y = 309.1x^{-0.278}$

$R^2 = 0.8792$
Our test consisted of 15 repetitive tasks.
Each task followed the same general workflow, but the function blocks, parameters, and properties being worked on were varied.
The task instructions were written in general terms such as “Add an AI block”, but did not specify how to carry out the work.
Mean Time to Complete a Task in System A

![Mean Time-On-Task Graph]

- **Task Number**
- **Minutes**

- **System A**

---

**Legend:**
- Grey bars represent System A.
Standard Deviation for a Task in System A

Mean Time-On-Task

Minutes

Task Number

System A
Power Curve Matching Tasks of System A

Mean Time-On-Task

y = 3.8080x^{-0.2690}
R^2 = 0.8219
Mean Time to Complete a Task in System B

Mean Time-On-Task

Minutes

Task Number

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

System B
Standard Deviation for a Task in System B

Mean Time-On-Task

Minutes

Task Number

System B
Power Curve Matching Tasks of System B

**Mean Time-On-Task**

System B:
\[ y = 3.9822x^{-0.2777} \]
\[ R^2 = 0.8609 \]
Overall Learnability

**Mean Time-On-Task**

System A:
\[ y = 3.8294x^{-0.2320} \]
\[ R^2 = 0.8008 \]

System B:
\[ y = 3.9822x^{-0.2777} \]
\[ R^2 = 0.8609 \]
Physical Effort

Mean Eye Path Traversed

System A:
\[ y = 1,446.7236x^{-0.2387} \]
\[ R^2 = 0.7327 \]

System B:
\[ y = 1,962.7388x^{-0.3365} \]
\[ R^2 = 0.8332 \]
Physical Effort

**Mean Mouse-Path Traversed**

- **System A:**
  
  \[ y = 25,694.2029x^{-0.1227} \]
  
  \[ R^2 = 0.5831 \]

- **System B:**
  
  \[ y = 34,390.1019x^{-0.2426} \]
  
  \[ R^2 = 0.6913 \]
A methodology involving eye tracking is a viable tool for objectively measuring usability.

After Learning point is reached, both System A and B have very similar usability characteristics.

[After moderate training] student performance is close to “real user’s” performance.
The GUI Course
Teaching Usability

- Traditional usability
  - Using standards and “classical” textbooks; e.g., Schneiderman’s book
  - User experience and HCD
  - Do / do–not–do / recipes
  - Practice
- Effort based usability
  - Presentations
  - Practice
Platform/Applications

- **Stand-alone**
  - Becoming less commonly used / less challenging
  - Qt’s “home-field”

- **Web based interface**
  - Dull HTML basis
  - Alternatives such as JavaScript, PHP
  - QtWebkit like library is missing

- **Mobile computing**
  - QtWebkit is a viable option;
    - But it is not “main-stream”
    - What is the role of mobile computing in a GUI class?

Concentrating on stand alone applications
Covering web based application
Windows, Linux, OS–X, iOS, Android,
- “Default” OS is Linux. Students can choose other OS

Language (Java, C++)
- We use C++

Java, Visual C (or similar), Qt UI–Library
- The Qt frame–work is used

A download–site (or DVD) with VMware Linux–VM, QT (4.8 and/or 4.7), OpenGl 2.x, and QtSPIM is available for students’ use.
SPIM: A MIPS32 Simulator

James Larus
spim@larusstone.org

Contents

- Older Versions of SPIM
- Further Information
- Changes to SPIM
- Copyright

Spim is a self-contained simulator that runs MIPS32 programs. It reads and executes assembly language programs written for this processor. Spim also provides a simple debugger and minimal set of operating system services. Spim does not execute binary (compiled) programs.

Spim implements almost the entire MIPS32 assembler-extended instruction set. (It omits most floating point comparisons and rounding modes and the memory system page tables.) The MIPS architecture has several variants that differ in various ways (e.g., the MIPS64 architecture supports 64-bit integers and addresses), which means that Spim will not run programs for all MIPS processors.

Spim comes with complete source code and documentation.

Spim implements both a terminal and windows interfaces. On Microsoft Windows, Linux, and Mac OS X, the spim program offers a simple terminal interface and the QtSpim program provides the windowing interface. The older programs xspim and PSpim provide window interfaces for these systems as well.

Download SPIM

QtSpim

The newest version of Spim is called QtSpim, and unlike all of the other version, it runs on Microsoft Windows, Mac OS X, and Linux—the same source code and the same user interface on all three platforms! QtSpim is the version of Spim

What's New?

QtSpim is a new user interface for Spim built on the Qt UI framework. Qt is cross-platform, so the same user interface and same code will run on Windows, Linux, and Mac OS X (yeah!). Moreover, the interface is clean and up-to-date (unlike the archaic X windows interface).

Spim has moved to SourceForge! The source code for all version of Spim are in an SVN repository and compiled version are available for download. There is also a bug tracker and discussion forum. Spim is an open source project, so please join in and contribute.
Why Qt?

- You all know the answer
Why Not Qt?

- Less known in academia (default would be Java–UI)
- Many students are familiar with the Microsoft based UI development environment (Visual–X)
  - Misunderstandings about open software
- It is a fast moving target and might be a bit overwhelming
  - Often, identifying the best specialized alternative requires a long learning time (learning curve)
  - Teaching the teachers
Why Qt (revisited)

- The signals and slots implementation of event driven programming
- Great example of good use of software engineering, OO design, and development principles
  - The GUI class is a required course for SE majors
- The rich library can be a real source of reuse
- Abundance of [self]-study aids
  - [free] Books
  - Demos
  - Videos (e.g., ICS)
- Qt designer (pros and cons)
- Inclusion of OpenGL
- Additional utilities and tools
  - Networking, IPC, Multithreading, etc.
Teaching Qt

- An intro lecture (About three hours)
  - Documentation/ study–material roadmap
  - First few tutorial sections
    - Emphasis on event driven programming / signal and slots

- Rest of the Qt learning process is mostly self-study through an “incremental project”
  - A short roadmap / demo before each increment
    - Graduate students should know how to read user manuals

- A few additional “directed” lectures

- Restricted use of Qt Designer
Assignments and Projects
Music-Box
Music-Box
Music-Box
Music-Box
Cache Simulation
Cache Simulation

Simulation Summary:
- # hits: 11
- Comp Misses: 76
- Gen Misses: 2
- T Cache: 167
- T No Cache: 89
- Miss Ratio: 38
Using Designer

DEPARTMENT OF COMPUTER SCIENCE

Welcome to the Computer Science Department

Events Calendar

<table>
<thead>
<tr>
<th>Sun</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Sat</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>19</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>20</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>21</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
</tr>
<tr>
<td>22</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
</tr>
</tbody>
</table>

[Image of computer science department website]

[Quick Links]
- Undergraduate Programs
- Graduate Programs
- Certificate Programs
- Calendar and Schedules
- Faculty Directory

[Details Area]
WELCOME TO COMPUTER SCIENCE

The Department of Computer Science is one of the oldest computer science departments in the state. It is also a fast-growing academic entity at Texas State University-San Marcos and the hub of computing related education and research activities on the campus.

The department offers the BS and BA degrees in computer science and a certificate program at the undergraduate level. At the graduate level, the department offers the MS degree in computer science and the MS degree in software engineering. We are in the process of establishing a PhD degree program. Our BS degree program is ABET-accredited. ABET, Inc. is recognized by the Council for Higher Education Accreditation. More...

Our classes are taught by a combination of permanent faculty members and a smaller number of adjunct faculty members from industry who provide student access to cutting-edge knowledge. All of our faculty members are available to students outside of the classroom. Our students have access to an amazing array of hardware, system software, and applications in our first-class computing facilities. We are rightfully proud of our degree programs.

The department currently has 21 tenured or tenure-track faculty members actively pursuing research in artificial intelligence, computer communication and networking, computer forensics, computer security, database and information systems, distributed and parallel computing, human computer interaction, multimedia computing, sensor networks, software engineering, and Web technology. We engage other academic units on campus and local industry in a variety of research collaborations.

— Hongchi Shi, PhD
Professor and Chair
Department of Computer Science

SPOTLIGHT

Introduction to Digital Media & Multimedia Programming

Digital Multimedia, a new course offering, will be taught by Dr. Dan Tamir. Dr. Tamir was a member of the Israeli delegation to the MPEG Standard Committee from 1996-1999. Students in this course will be able to explore advanced multimedia equipment and software. The material will cover the principles of digital acquisition, representation, processing, and storage of the major multimedia data types: image, graphics, audio, and video. It will include a review of compression techniques for multimedia, multimedia standards, and storage strategies. Particular emphasis will be placed on multimedia data compression. More...
Teacher’s Lounge

...a place for educators and researchers to connect, share, and learn.
A few additional “directed” lectures

- Computer graphics via Qt
  - Emphasis on Qt-GL
  - Contrasted with GLUT
- Web, HTML, CSS, PHP based GUI
- Art design view of usability (Guest lecture)
- Visualization
- Networking / multi-threading
Non-destructive UI
QtGL
QtGL (Animation)
GA–TSP (1)
GA–TSP (20)
GA–TSP (1000)
Laser Lithography Interface
Additional Assignments and Projects

- Data Base Management
- Travel Reservation
  - MS Queue
  - Qt Queue
[Free] Books

C++ GUI Programming with Qt 4

JASMIN BLANCHETTE  MARK SUMMERFIELD
FOREWORD BY MATTHIAS ETTRICH
[Free] Books
Books

Designing the User Interface: Strategies for Effective Human-Computer Interaction, 7th Edition

Ben Shneiderman & Catherine Plaisant
Conclusions

- Decisions Decisions