Presenting Large Graphical Contents on Mobile Devices – Performance Issues

René Rosenbaum, Heidrun Schumann, Christian Tominski
University of Rostock, Germany

14th IRMA International Conference, New Orleans, 25/05/2004
Outline

- Introduction to mobile devices
- Graphics on mobile devices
  - Raster approach
  - Vector approach
- Performance measures
  - Quality
  - Update time
- Conclusions
Introduction to Mobile Devices

- Mobile devices: variety of small wearable computing devices
  - Mobiles,
  - Handhelds,
  - Tablet PCs
- Many fields of applications
  - Personal Information Management
  - Location Based Services
  - Mobile Data Collection
- Limitations due to device size
  - Display (size, resolution, color)
  - Computational power
Introduction to Mobile Devices

- Current handheld properties concerning graphics:
  - Display size: about 3.5 inch
  - Display resolution
    - Varies from 160 x 160 to 240 x 320 pixels
    - High end handhelds 640 x 480 pixels
    - Most spread handhelds still low resolution
  - Display colors
    - 12bit to 16bit in general
    - 4bit devices still widespread
- Computational power
  - Various RISC processors with 16MHz to 400MHz
- Memory
  - 4MB to 64MB
  - Extensible with memory cards
Graphics on Mobile Devices

- Presentation of much information by means of graphics / images
- Graphical information easily comprehensible
- Two general approaches
  - Raster graphics
  - Vector graphics
- Which approach should be used on mobile devices?

Much information: databases, services, personal information, etc.

Graphical representation

Raster graphics  Vector graphics?
Graphics on Mobile Devices

Raster approach
- Pixels carry graphical information (i.e. color)
- Pixels arranged on regular, usually rectangular grid
- Variety of file formats, e.g. uncompressed BMP, compressed JPEG
- Rendering pipeline
Vector approach

- Graphics described by geometric primitives and their attributes (e.g. stroke-width, fill-color, ...)
- File formats for mobile devices: proprietary Flash, standardized SVG, ...
- Rendering pipeline

```
<line x1="100" y1="300" x2="300" y2="100" stroke-width="5" />
<line x1="300" y1="300" x2="500" y2="100" stroke-width="10" />
<line x1="500" y1="300" x2="700" y2="100" stroke-width="15" />
<line x1="700" y1="300" x2="900" y2="100" stroke-width="20" />
<line x1="900" y1="300" x2="1100" y2="100" stroke-width="25" />
```
Performance Measures

- Scenario: Interactive exploration of large graphics with basic **zoom & pan** functionality
- Used device: Toshiba e740
- Operating system: Pocket PC 2002
- Testbed for measuring:
  - Presentation quality & presentation time
  - Raster & vector approach
  - Native code (MFC), .Net Compact Framework, & Java (Jeode)
High presentation quality achieved if only little modifications to the graphical information occur throughout rendering pipeline

- **Raster approach**
  - On zooming (i.e. image scaling)
    - Poor quality for build-in system scaling functions (omitting or multiplying rows and columns of an image), unless filtered scaling is used (interpolating each new pixel’s value depending on neighboring pixels)
  - On panning (i.e. drawing different parts of image)
    - No changes in quality

- **Vector approach**
  - On zooming and panning (i.e. applying the transformation matrix)
    - Highest quality, since no information gets lost
    - Slight decrease in quality by rasterization, unless anti-aliasing is used
Presentation Time

- Presentation time includes
  - Loading time (only once at start-up)
    - Reading time (reading from file)
    - Decoding time (creating the IMR)
  - Update time (on each interaction)
    - Transformation time (scaling for raster graphics; matrix multiplication for vector graphics)
    - Drawing time (rendering IMR to display)
- Interactive scenario:
  - No waiting longer than few seconds at start-up
  - Smooth interaction at about 15 refreshes per second
  - Thus, preferred update times < 67 ms
Loading Time

- Raster approach
  - BMP: slow reading, fast decoding
  - JPG: fast reading, slow decoding
- Reading time raster vs. vector approach

![Graph showing loading time vs. number of primitives for different file formats: BMP, JPG, SVG.](image)

- Decoding time depends heavily on used file format
Update time preferably low for interactive response rates

Raster approach
- Interactive zooming and panning is possible, low quality

Vector approach
- Integer vector transformation fast due to RISC architecture
Drawing Time

- Raster graphics drawing
  - Very fast ca. 50 ms
  - Independent from image size
- Vector graphics drawing (rasterization)
  - Interactive rates only for up to about 100 primitives
  - Depends on complexity of primitives

![Time vs. Number of Primitives Graph](image)
Indirect Rendering

- **Speed-up for vector graphics**
  - On zooming
    - Render transformed vector primitives to a new memory buffer bitmap
  - On panning
    - Do NOT perform matrix multiplication and re-rendering
    - Instead, draw different part of memory buffer bitmap

- **Advantage**
  - High response rates during panning without loss in quality

- **Disadvantage**
  - Memory requirements for buffer bitmap
Conclusion

- **Graphics loading**
  - Loading time depends strongly on the used file format.
  - Loading raster data is generally fast and requires additional computational efforts if content must be decompressed.
  - Loading vector data is fast for up to about 100 primitives.

- **Graphics rendering**
  - Drawing raster data is fast on mobile devices.
  - Rendering vector primitives directly to display is generally slow and depends on primitive number and complexity.
  - Rendering vector primitives indirectly using an IMR-bitmap achieves faster update times.
Conclusion

- **Quality**
  - A simple scaling of raster data is fast but leads to low quality presentations.
  - Scaling vector graphics by integer matrix multiplications is very fast and achieves high quality.

- **Development environment**
  - Implementations based on MFC are fastest.
  - .Net has a slight performance advantage over Java on the used mobile device.

- **Conclusion**
  - Few primitives at high quality, use Vector approach
  - Large number of primitives at acceptable quality, use Raster approach
Future

- Today’s devices contain more performance, operating system is limiting factor
- Graphic capabilities will increase
  - OS Pocket PC 2003: performance increase approx. 35%
  - “The Microsoft Game API (GAPI) aimed at high-performance gaming and visualization applications, …”
- Last not least
  - Recent tests: SHARP Zaurus (www.zaurus.com) running on Linux and Qtopia (www.trolltech.com)
  - Zaurus outperformed the Toshiba device