OpenOffice.org Conference 2008 Beijing

OpenOffice.org's Aqua Port

Herbert Dürr

(Sun Microsystem)

Overview

- Status of the Aqua Port
  - what has been accomplished
- Development
  - Specifics of a Productivity Suite
  - Typical Problems of a Port
  - Code Refactoring
- Contributing
  - as a normal user
  - as expert
  - as developer

What has been accomplished

- The Aqua Port was now officially released with OpenOffice.org 3.0
- Based on OpenOffice.org X11 for Mac OSX project that started about five years ago
- Functionality matches and exceeds other ports
- Good system integration
- User Interface is conceptually a cross-platform port
- Extensions work nicely

More Technical Details

- Mac OSX 10.4 and newer required
- Cocoa vs. Carbon
- 64bit vs. 32bit, x86 vs. PPC
- The port has been accomplished almost exclusively by extending OOo's cross-platform layer code
  - Application development on that platform is usually specifically for that system
  - a pragmatic approach
- Better accessibility than competitors

Text status

- Coretext vs. ATSUI
- Justified Text
- Vertical Writing
- Beyond the unicode baseplane
- BiDirectional Text
- PDF-export
- Advanced Typographic Font Features

Development

- Specifics of a Productivity Suite
  - Long-Livedness
  - Compatibility
  - View Independence
- Typical Porting Problems
  - Multi-Platform vs. Optimal Integration
- Code Refactoring
  - Why is it needed?
  - A successful recipe

Porting Approaches

- Top-Down
  - Allows a clean and modern design
  - Everybody likes rewritten code
- Bottom-Up
  - Getting things done
  - Don't impact other ports
  - Efficient code reuse
  - Less Regressions
  - Ready for stabilization branches

Careful Refactoring (1)

- Understand new requirements
- Understand existing interfaces+code
- Blackbox the obsoleted code
- Understand the existing use cases
- Sanitize the blackbox's interface
- Reuse the old code for implementing the sanitized interface
- Implement obsoleted interfaces with the sanitized ones

Careful Refactoring (2)

- Replace obsoleted code
  - make old/new codepaths easily switchable
- Extend the sanitized interface
  - can often be merged into the existing interface
- Make other layers use the sanitized interfaces
  - eventually add helper methods
- Remove the obsoleted parts of the old interface

→ The cleaner interface helps a lot with porting

Careful Refactoring: An Example

Example: Polygon Clipping via XOR trick

- XOR is very difficult to implement in Quartz

  for a good reason:

  the concept of directly messing with pixel bits has been obsolete for a long time already!
- Why was it still used?

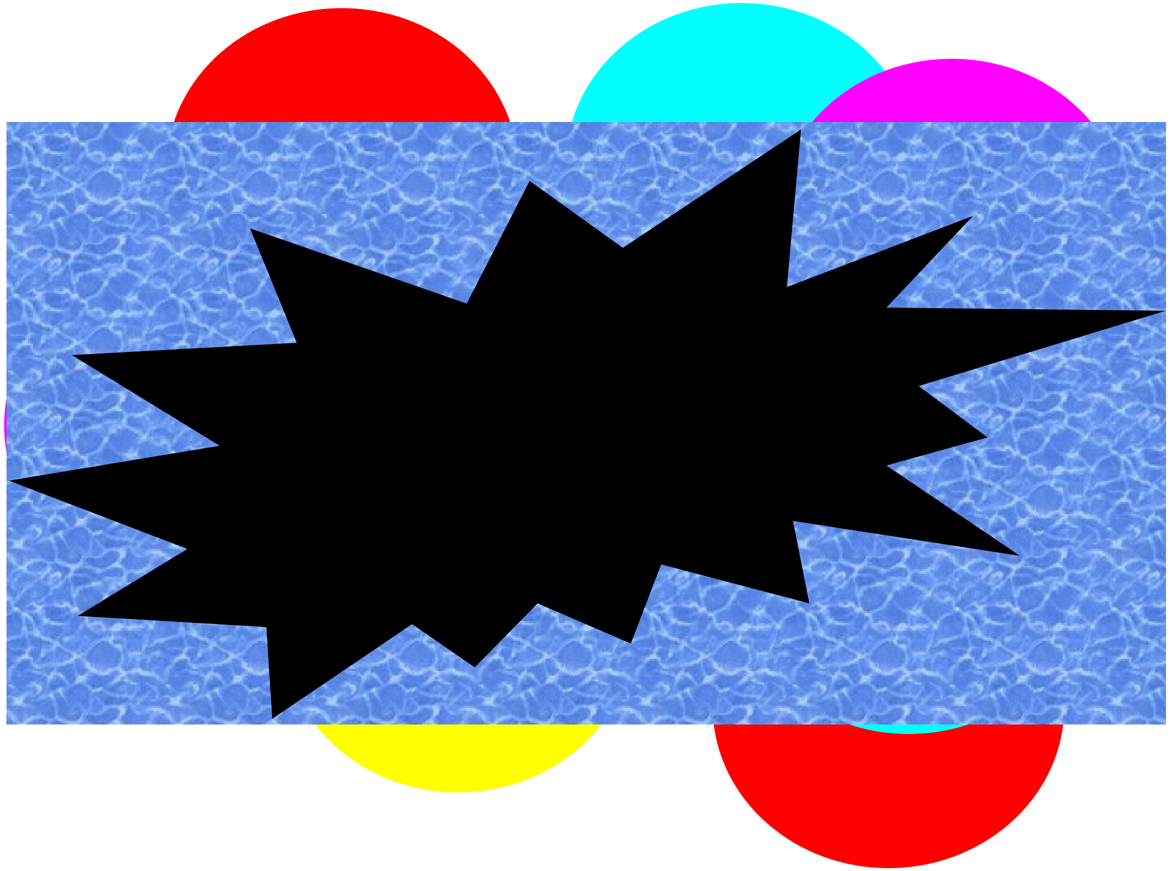
  It is a clever trick to implement complex clipping on graphics systems that have minimal capabilities

The XOR Example (2)

How does the obsolete implementation trick work:

a) Enable XOR drawing mode
b) Draw the background
c) Enable BLACK drawing mode
d) Draw the clipping polygon
e) Enable XOR drawing mode
f) Redraw the background as in b)

The XOR Example (3)



The Need to Refactor

A different kind of bug

- When interfaces do not suffice
- Often a result of missing separation of interface and implementation
  - implementation trick as interface
  - the implementation trick becomes obsolete
- Different approach to fixing depending on whether the project is in an early or a stabilization phase

Implementation vs. Interface

The root cause of many problems

- Bitmap as array of pixels
  - the XOR example
  - color space, dithering
  - previews, extracts, etc.
- Clipping polygon vs clipping rectangles
- Unicode codepoint vs. uint16/uint32
- a modern example

Implementation vs. Interface

A modern example

- #10 0x1fa901a6 in std::for_each<__gnu_cxx::__normal_iterator<rtl::Reference<canvas::Sprite> const*, std::vector<rtl::Reference<canvas::Sprite>, std::allocator<rtl::Reference<canvas::Sprite> > > >, boost::_bi::bind_t<void, void (*)(OutputDevice&, basegfx::B2DPoint const&, rtl::Reference<canvas::Sprite> const&), boost::_bi::list3<boost::reference_wrapper<VirtualDevice>, boost::reference_wrapper<basegfx::B2DPoint const>, boost::arg<1> (*)()> > > (__first={_M_current = 0x1f779120}, __last={_M_current = 0x1f779124}, __f={f_ = 0x1fa5d28c <vclcanvas::(anonymous namespace)::spriteRedrawStub2(OutputDevice&, basegfx::B2DPoint const&, rtl::Reference<canvas::Sprite> const&)>, l_ = {<storage3<boost::reference_wrapper<VirtualDevice>,boost::reference_wrapper<const basegfx::B2DPoint>,boost::arg<1> (*)()>> = {<storage2<boost::reference_wrapper<VirtualDevice>,boost::reference_wrapper<const basegfx::B2DPoint> >> = {<storage1<boost::reference_wrapper<VirtualDevice> >> = {a1_ = {t_ = 0x1f769a30}}, a2_ = {t_ = 0xbfffe578}}, <No data fields>}, <No data fields>}}) at /usr/include/c++/4.0.0/bits/stl_algo.h:158rn example

Help to improve

- As a user
  - use it
  - find problems
  - isolate problems
- As an expert
  - provide expertise and suggestions
- As a developer
  - find the root cause in the code
  - provide a patch to fix the root cause

Isolating problems

- make a problem reproducable
- reduce the test case to be obvious and minimal
  - find the point where the problem starts/goes away
- provide a screenshot for visual problems
- submit a crash report for stability problems
- test it with other versions
- test it on other platforms

TODOs

- More Integration
- Smoother Graphics
- Better Performance
- Printer Pull Model
- Better PDF-export
- Apple Script

Questions & Answers

OpenOffice.org
Conference 2008 Beijing

OpenOffice
Conference 2008

Thanks！

凝聚全球力量 绽放开源梦想

www.OOobeijing2008.com