NetBeans Platform

Tom Wheeler

06/15/07

St. Louis Java User Group :: June 2007

Who Is This Up Here?

- Senior Software Engineer @ OCI
- Started using NB Platform in 2005
- Became active on mailing lists
- Wrote two JNB articles in 2005
- Won NB Community Award in 2006

What is NetBeans?

- An open source project
  - Sponsored by Sun Microsystems
  - 100% pure Java
  - Way better than you probably remember it
- Both an IDE *and* a platform

OK, So What is the NB Platform?

- The platform is...
  - A bunch of plug-ins (modules)
  - 100% Pure Java
  - A framework for application development
  - What's left when IDE features removed
- IDE is therefore a platform-based app.
  - IDE = Platform + IDE modules

Deep Thoughts, Without Jack Handey

- If the platform is a foundation...
  - Which consists of a bunch of modules
  - And you can extend it by adding modules
  - And doing so creates a new application
  - And you can add modules to that new app
  - *Then isn't the new app also a platform?*

Demonstration

- If the IDE is a platform-based app...
- Shouldn't I be able to make my own IDE?

And Now for a Rhetorical Question...

- When is the last time...
  - You wrote a serious Web app in Java
  - Using only servlets and JSPs?

Why Use a Platform for Swing Apps?

- Frameworks are widespread for Web apps
- But seldom used for Swing...
  - No good reason for this, AFAIK
- 37% of any Swing app's code:
  - Is identical to 37% of any other Swing app
  - I totally made that up, but probably close

More Good Reasons to Use a Platform

- Support for modules/plugins
  - With dependency management
  - Deploy updates and new features easily
- Help your application grow over time
  - You're likely to create better APIs
  - Versioning support for smoother upgrades
- #1 Reason:
  - Spend your time on actual business logic

OK, So What Platforms Exist?

- NetBeans Platform
- Swing Application Framework (JSR-296)
- Eclipse RCP
- Spring RCP
- Countless others
  - But probably none you'd consider worthy

Swing App Framework – JSR 296

- https://appframework.dev.java.net/
- Supports some basic needs, like
  - Loading images / managing Actions
  - Basic data storage (like frame geometry)
- Does not support
  - Branding, modules, dependencies, windowing
- Migration plan for when you outgrow it?

NetBeans Platform

- http://platform.netbeans.org/
- Mature (platform available since 2001)
- Open source (CDDL, an MPL variant)
- 100% Pure Java
  - Easily reuse Swing code
  - Uses Ant extensively
- Many features
- Adequate documentation and examples

Eclipse RCP

- http://www.eclipse.org/rcp/
- Mature: RCP available since late 2003 (?)
  - Open source (EPL – a CPL Variant)
  - Plentiful documentation and examples
  - Many features
- Use of SWT
  - Platform limitations!
  - Reusing existing Swing code is tough

Spring RCP

- http://spring-rich-c.sourceforge.net/
- Sub-project of the Spring Framework
- Don't know much about it, but
  - Data binding and validation a major feature
  - Offers at least rudimentary management
  - Plugin support unknown
  - Little documentation
  - Current version is 0.2.1, released 9/06

Countless Other Platforms

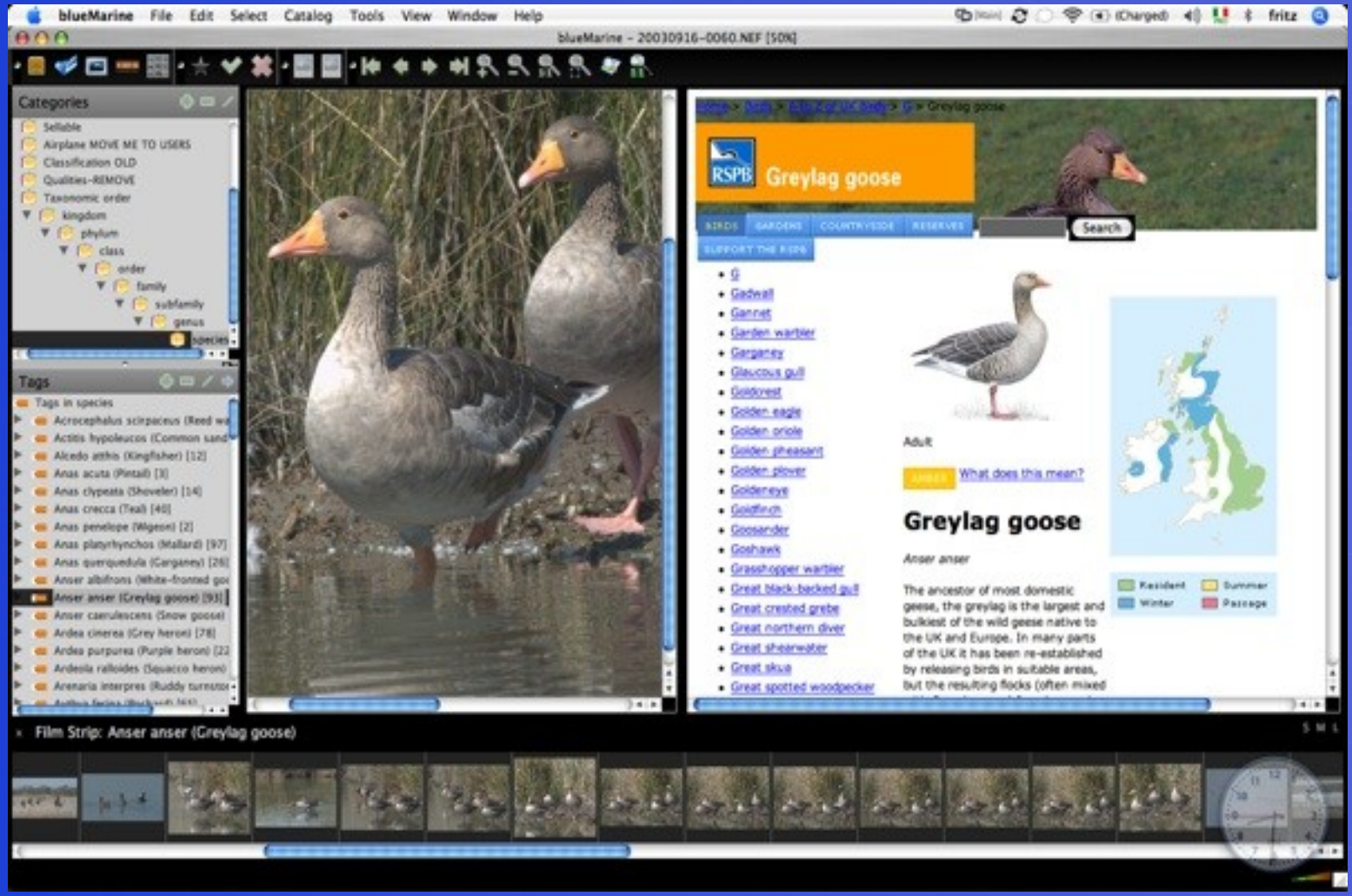
- There are lots of other minor players
- Some are relatively full-featured
  - But immature
- Others are relatively mature
  - But focus on a single feature
    - Example: Java Plugin Framework (JPF)
    - http://jpf.sourceforge.net/

OK, So Which Should I Choose?

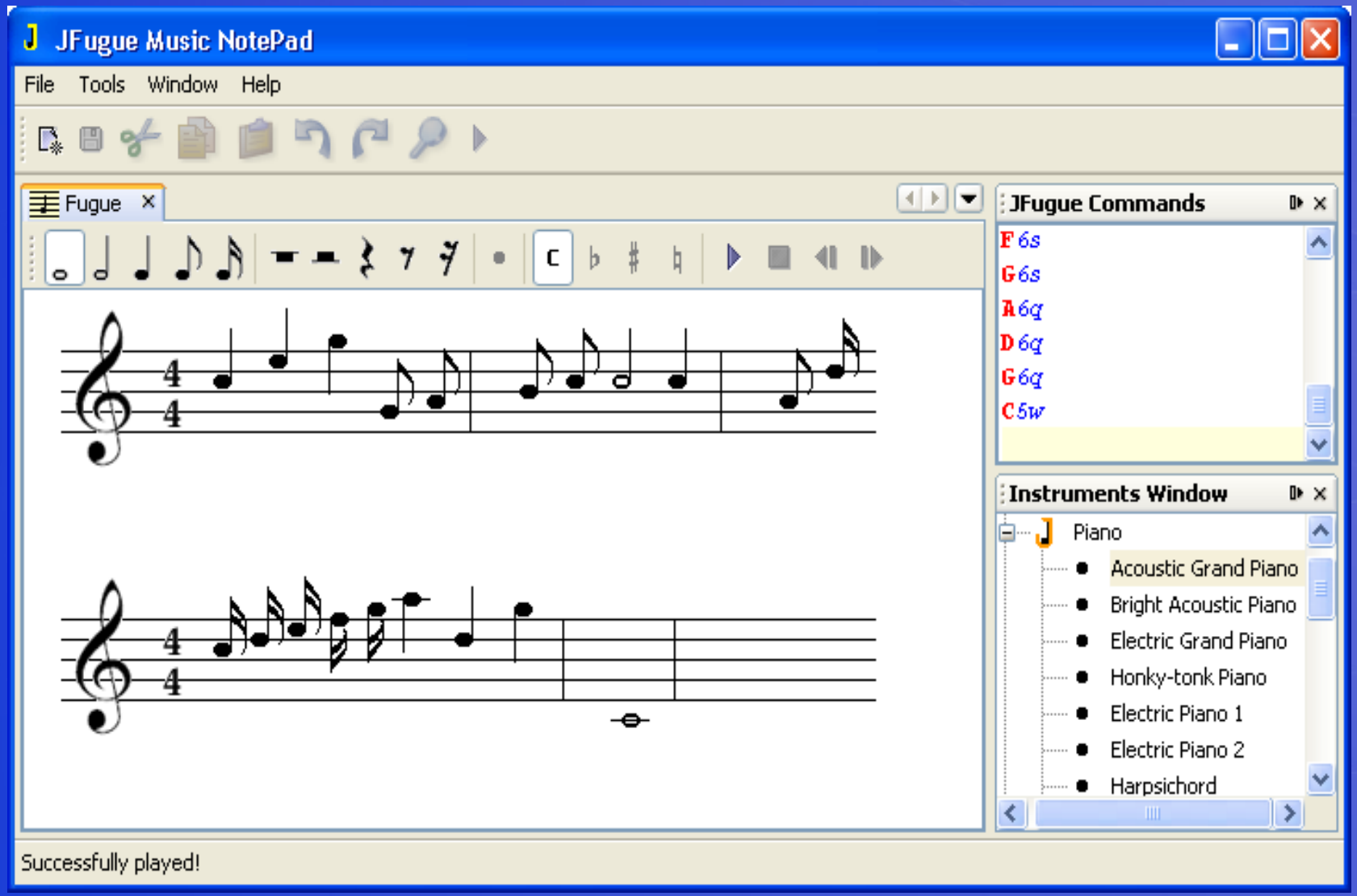
- Typically NetBeans vs. Eclipse
  - Features are roughly equivalent
  - Both are probably good choices
  - Depends on exact requirements
    - SWT was a dealbreaker for me, but YMMV
- Also note potential IDE "lock-in"
  - Eclipse RCP effectively requires Eclipse IDE
  - NB Platform heavily favors NetBeans IDE
    - Ant integration allows other IDEs somewhat

NB Platform Example: AIOTrade

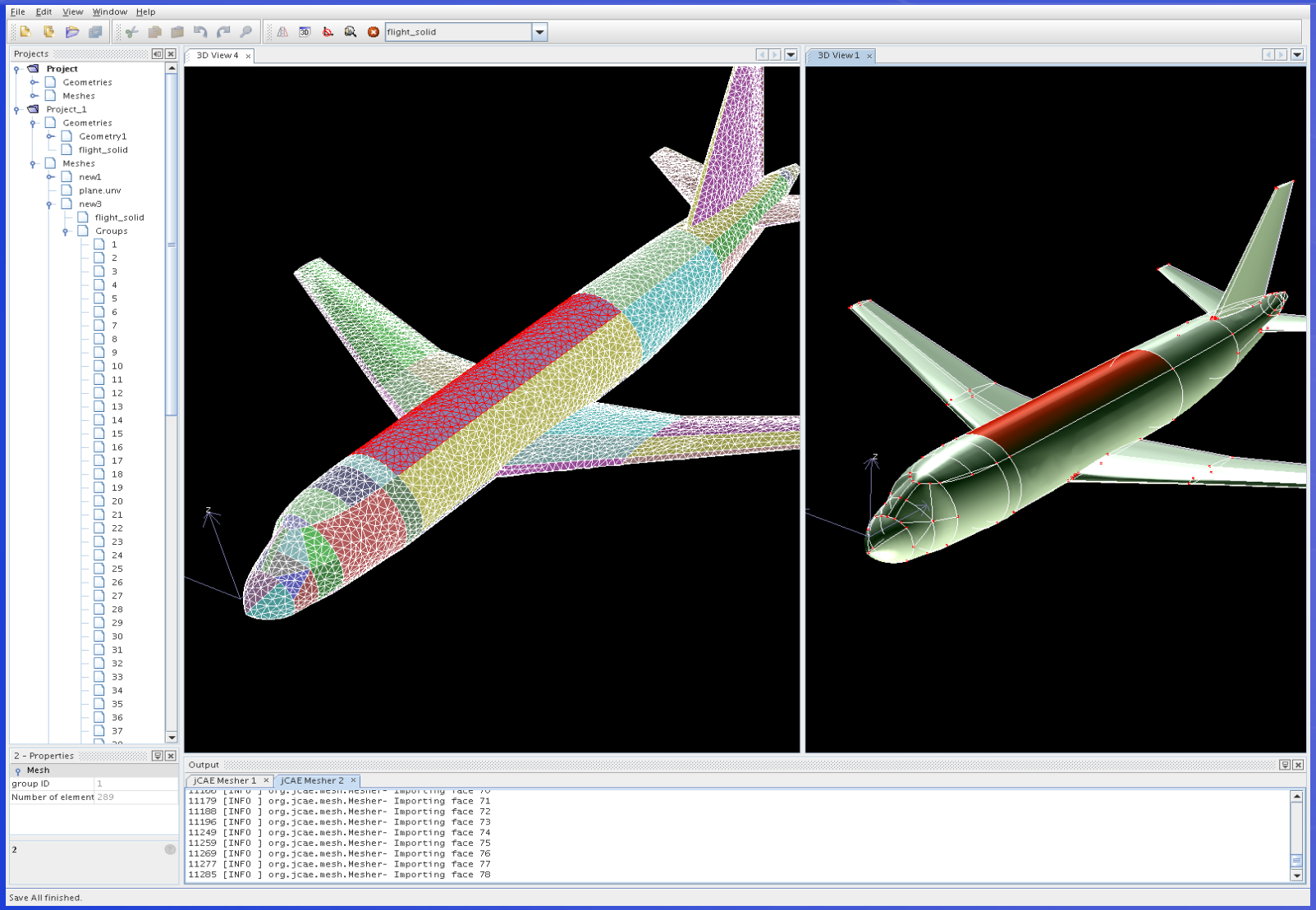
NB Platform Example: blueMarine



NB Platform Example: Music Notepad



NB Platform Example: JCAE



NetBeans Platform Features

- Modules with dependency management
- AutoUpdate Feature
- Declarative UI
- Windowing system
- Integrated JavaHelp
- Flexible Filesystem implementation
- Deploy via JNLP
- Build harness (Ant scripts)

NetBeans Platform Features

- Nodes, Explorer and PropertySheet APIs
- Wizard framework
  - Supports both static and dynamic paths
- Flexible filesystem implementation
- Many handy UI utilities
- Can reuse any module from the IDE

NB Fundamentals Overview

- Suite and modules
- The layer file and the System Filesystem
- Actions
- TopComponents and Modes
- Explorer, Nodes and Properties
- Cookies and Lookups

What is a Suite?

- A suite is
  - Configuration of a platform app.
- A suite contains
  - A list of modules
  - Branding (icons, splash screen, labels, etc.)

What is a Module?

- A module is
  - A single indivisible “piece” of an application
  - A provider of some feature or content
- A module contains
  - Exactly one manifest file and XML “layer” file
  - One or more resource bundles
  - Java code, JAR files and/or native libraries
  - Maybe some JavaHelp content

What Kinds of Modules Are There?

- There are two main types of modules
  - "Normal" (contain source code)
  - Library (contain one or more JAR files)
  - IDE wizards simplify creation of both types

What Should I Know About Visibility?

- In Java you have four types
  - private, default, protected and public
- Having "semi-public" visibility
  - For example, public only in same JAR
  - Would help in creating cleaner APIs
- NetBeans does this!
  - "Public" means "public" in that module only
  - Export the package so other modules can see

What is a Layer File? System FS?

- Complex apps need some type of registry
- NetBeans uses the “System Filesystem”
  - An XML-based filesystem
  - Menus, toolbars, etc. are configured here
- Each module has a “layer” file
  - This gets merged into System FS at runtime
  - Modules can add, modify and delete items
- You can use it for your storage too

And What About Actions?

- Same as in Swing, they "do things"
- Can generally use Swing AbstractAction
  - There are also NB-specific types
    - For both context-sensitive and stateless use
    - Makes handling `isEnabled()` easy

TopComponents a la Mode

- TopComponent is basically a JPanel
  - But also a window in NB windowing system
- Every TopComponent “lives” somewhere
  - This place is called a *mode*
- Modes are named after IDE components
  - Explorer
  - Editor
  - Output

Explorer, Nodes and Properties

- Nodes are central to NB programming
  - Presentation layer
  - Represent some type of data
    - For example: Customer, Order or Product
- Nodes are displayed in an explorer view
  - Typically a tree-based view
  - But there are other views (table, list, menu)
  - Can typically switch views w/o model change
  - Try that with Swing!

Tasty Cookies

- Cookies aren't what you think
  - Have nothing to do with HTTP or X-Windows
- Represent some capability of an object
- Can dynamically add and remove them
  - For example, `SaveCookie` interface
    - Has one method: `save()`
  - When active node has a SaveCookie
    - File -> Save is enabled
    - Otherwise it is not

Lookups: Even Better Than a Cookie

- Lookups are a more modern version
  - Don't require you to impl. marker interface
- There is also a “Global Lookup”
  - You can code to an interface
  - Find implementation at runtime
  - Ideal for plugging in algorithms
  - Similar idea now in Java 6 (ServiceLoader)

Putting It All Together: An Example

- Prepare to be mystified

Cue the Closing Credits

- NB Platform is
  - Free
  - Open source
  - A better way to build large Swing apps
  - Proven technology used by
    - Sun, Nokia, USDA and many others
- See the new NB Platform Book
  - http://www.netbeans.org/books/rcp.html