OpenROAD 2006 Features

- OpenROAD2006 has been out for a while now
  - and OpenROAD4.1 will soon be retired

- What does 2006 have to offer?
  - Big changes to the IDE! but also:
    - Some very powerful functional changes, with considerable practical significance to you and me
      … these are the focus of this presentation
  - and: Some important changes that we have been pursuing since 2006 came out, that are coming to fruition
    … these will be briefly mentioned
A List of Changes

- The IDE
- Portlets
- Visual enhancements
- Field and Class properties
- Events
- Constructors
A List of Changes

- **OpenROAD Server changes**
- **Major overhaul to the documentation**
- **OpenSource OpenROAD**
  - Community product changes
The New IDE

- The Interface is
  - Friendly
    ... usability is the test, and new developers like it
  - In line with current industry standards
  - Part of a systematic upgrade of the interfaces
    - and preparation for opensource contributions

- Take the time to look and try
  - key elements like cross-reference, class browser, that are now under the hand
  - really nice features in the debugger (plus break on change)
  - eClient deployment
The New IDE
Applications and Components
Debugger
Web Deployment (eClient)
Portlets

- You can reparent a CompositeField to a new host frame at runtime
  - without affecting the routing of its local events
- Existing tools can be plugged into a framework
  - with minimal adjustment (they still work standalone as well)
  - tools can be switched in at runtime, in a few lines of code
- This is a powerful enhancement
  - The new IDE uses this extensively
  - Client organizations developing on 2006 are exploiting this heavily
  - Don’t miss it!
Portlets – an example
Visual Enhancements

Transparency:

- Until 4.1, the FP_CLEAR setting only had meaning for:
  - FreeTrim
  - ShapeFields
    ... and then, only if these were not "real fields"

- Now, FP_CLEAR makes all field types transparent - except:
  - Imagefields and Imagetrim and PaletteFields
    – use BDB_TRANSPARENT
  - Scrollbars
→ Now you can ...

- have a continuous background pattern across a form, even if there are compositefields on it
  ... Before, each CompositeField restarted the background pattern
- give TabFolders a background colour or pattern
  ... Even the tabbar and tabs
- have curved arrows on displays
- have under-fields that are visible and clickable
- simulate rounded corners on fields to match other visual standards
Text handling – Underlining
  – Cell enhancements

- **Underlining**
  – Any field, anywhere
  – Font must be named native font
  – Brings OpenROAD into line with web interface textfields
  – Can't yet differentiate TreeView nodes, alas

- **TableField Cell enhancements**
  – Individual cells can be differentiated by
    • Emboldening
    • Underlining
    • Typeface, size, and name
  – Significant increase in ability to "read" table data
Field Properties

- **EntryField: ExactWidth**
  - Solves most remaining alignment issues
    - Use with snap-to-grid …

- **ButtonField: TextDisplayBehavior**
  - Variety of ellipses (…) for overlong text

- **Tabbar: BarPosition**

- **SliderField:**
  - DisplayValues, PageSize, StepSize, TickIntervals
  - Style

  \[
  \text{Style} = \text{SFS\_TRACKBAR} \\
  \text{DisplayValues} = \text{FALSE} \\
  \text{TickIntervals} = 10
  \]
Class Properties

- `StringObject.LocateString(backwards=TRUE)`
  ...at last

- `SessionObject.WinHelp()` supports HTML Help
Events

- **MouseEnter and MouseExit**
  - Flyover effects, as per web interfaces
    - Even for TableField rows (ChildMouseEnter, Exit)
  - Now you can give TableField rows “their own” dropdownlists at runtime
    - See taggedvalue editor
UserClass Constructors

- You can now add executing code to the Initialize block of a userclass
  - The code executes for each object as it is created
    - In declarations, .create() calls, array filling, frame initialization (if a field is mapped to that userclass)
    - Not in global variables! (a terrifying thought)
  - The executing code knows
    - Which object it belongs to (CurObject)
    - Which procedure, frame or method created it (CurMethod.Parent)
Constructors

- The core purpose (as the name suggests) is to construct more-complex objects
  - A UserClass definition can specify that a cow is created with legs and a head
    - But it can't say how many legs ...
  - The constructor code can create 4 legs, 1 head, 1 tail, etc for each cow.
    (And colour them in)
→ Making a Right **Cow**

Codeless, cubist cow.

No frame code at all.
Not one line. Just some fields.

Codeless, cubist, constructed cow.

…but we do have Cow, Head, Tail, and Leg userclasses.
- and a constructor (with the only code):

```
INITIALIZE =
  DECLARE
    ct = integer NOT NULL;
  ENDDECLARE
BEGIN
  FOR ct = 1 TO 4 DO CurObject.Legs.InsertRow(); ENDFOR;
  CurObject.Appearance.FileHandle = 'cow.bmp';
END;
```

“Every cow a perfect cow”
Constructors

A good way to guarantee integrity of the application's business objects

- 2 chessplayers
- 4 wheels
- 6 eggs

- But also …

- A good way to set default values for
  - object attributes
    (remember, object values don't exist before startup)
  - Subtype-specific inherited attributes
    - Whose values ('pen', 'ruler', 'paper') can't differ before startup
  - Array-attribute manager-objects …
Constructors

- **Array-attribute manager-objects**
  - The simplest approach to collection methods is to host them in the individual object
    - `Cow.GetHerdOfCows()`
  - otherwise you have 2*classes (cow and cows)
  - But: for array attributes (invoice.items), which 'individual item' will you use to manage the items?
    - Invoice.Items[1]  ..But then there must always be a row 1
    - it = Item.Create()  ..But that means unrelated variables
  - I use Invoice.Items.ClientData: very convenient …
    - but how to know .ClientData is set, without checking every time?
→ Set .ClientData = Item.Create()  *in the Constructor*
Constructors

- Help to set up frames too. For example:
  - Having a register of all fields in the frame is exceptionally useful
    - AllFields, FormFields, ToolbarFields, MenuFields, …
    - especially if your “register” exploits the list, with methods like
      - FieldsByClass, FieldsByProperty
        » get all tabpages …
      - FieldsByKeyword, FieldsByKeywordValue
        » get all fields that have “resize=none;” in the clienttext …
  - Now you have field registers, without any frame code
    - Give the register class a constructor that calls GetAllFields
    - Just drop a field with the “register” datatype onto the form
  - So you can add this to existing frames without risk!
→ Constructors

- **Provide a neat way to get debug info**
  - without adding code to every block that you then have to remove to go live!
  - Now you can just declare a userclass
    - The userclass has 2 versions: with and without an Initialize
    - You import one over the other to change behaviour
  - The Initialize code just passes the calls to the real handler
  - And without the Initialize, no code executes. No clutter. No code change going to Live!
Instance Variables

- The userclass Initialize block can now declare variables as well as procedures
  - Available to all methods & procedures in the class
  - Persistent across the lifetime of the object
  - Equivalent to other languages’ "private attributes"
    - OR “private” attributes = other product “protected”
  - In the past, the lack of persistent variables discouraged developers from porting code from frames to methods
    - Now code can uplift with much less change and risk
OpenROAD Server

- HTTP transport
- Java interface
- .NET interface
HTTP Transport

- Before 2006 we only supported OpenROAD Server access via DCOM
  - Very efficient, but
    - Limited to intranet (cannot pass firewalls)
    - Draconian security (especially after XP SP2)
    - Mostly incomprehensible (do this "because it works")

- We now support HTTP routing (http, https), which offers several advantages
  - No firewall limitation
  - Not arbitrary in its restrictions
  - Exploits web-server security (written for real humans)

- Specify in the connection call by setting
  - routing='http'
  - location=<http url>
HTTP Routing

- Gatekeeper
  - Does security as well as transformation
    - Lookup lists, referenced by the gatekeeper code
      - Which clients are allowed to call
      - Which SCPs the client is allowed to call
    - Username/password pairs, in the 'flags' parameter
      - One for the gatekeeper to process
      - One for the webserver to process (or IIS client, or Integrated Windows Authentication, ... as appropriate)
  - URL target
    » eg. https://myserver/myapp/srs.ashx
Java and .NET interfaces

- Effectively 'wrap' the COM interface
  - Replaces the Java-COM bridge approach previously required
- Similar to the standard COM interface, but
  - More strongly datatyped
  - Richer
- Straightforward to use
- Minor .NET limitations:
  - No type-specific get/set methods (unlike Java interface)
  - ASOSession.RSO is not available
    - Some VB/COM code conversions to .NET could be affected
  - Exception handling needs attention
→ OpenROAD Server

- Supports deployment of server-side 4GL business logic
- Makes OpenROAD 4GL business logic available to:
  - OpenROAD desktop, browser and mobile clients
  - J2EE (Java applets, servlets, & beans)
  - .NET (C#, ASP.NET, VB.NET)
  - Browsers (HTML, ASP, JSP)
  - Other clients (Visual Basic, C++, etc.)
- Supports publishing of OpenROAD 4GL object classes as Web services
- Visual server farm administration utility
OpenROAD Mobile Client (mClient)
OpenROAD Mobile Client

- Complete port of the OpenROAD Runtime to handheld devices and mobile phones
  - Supports Windows Mobile 5+ and Windows CE 4.2+

- Features
  - Runtime installer and application delivery (< 7 Mb)
  - Exploits native Windows Mobile features
  - Server connectivity via HTTP
  - Binary application compatibility with other runtimes
The 4GL OpenROAD product and development environment has been OpenSource for some time

- Active community involvement in the Dark Side
  - Collaborative development projects
    - Unicode UTF8 enablement
    - XML source management
    - Transforge
    - OpenROAD Look and Feel
    - And more …
  - New 4GL tools, published to the community
    - DAOGEN, PROXGEN, THUG, …
  - Development sprints
    - Side-by-side coding of new features to OpenROAD
  - Futures discussions
    - openroad-community, openroad-developers
Consider DAOGEN ...

- “Data Access Object Generator tool”
  - Easy-to-use developer tool built in OpenROAD 4GL
    - Wizard-driven
  - Generates OpenROAD classes
    - that provide simple object/relational mapping
  - Generates OpenROAD Server application
    - that publishes the DAO methods as commonly available services
  - Used to generate 80% of the code for the OpenROAD version of the Frequent Flyer demo
    - Then productized in short order
  - Now published as an open source, community project

- Very smart product! If you haven’t seen it, you must investigate. And the others are equally good...
OpenROAD is thriving ... 

- More people involved in product development than at any time in the last dozen years
- Substantial projects underway at all levels of the product
- OpenROAD is open for business