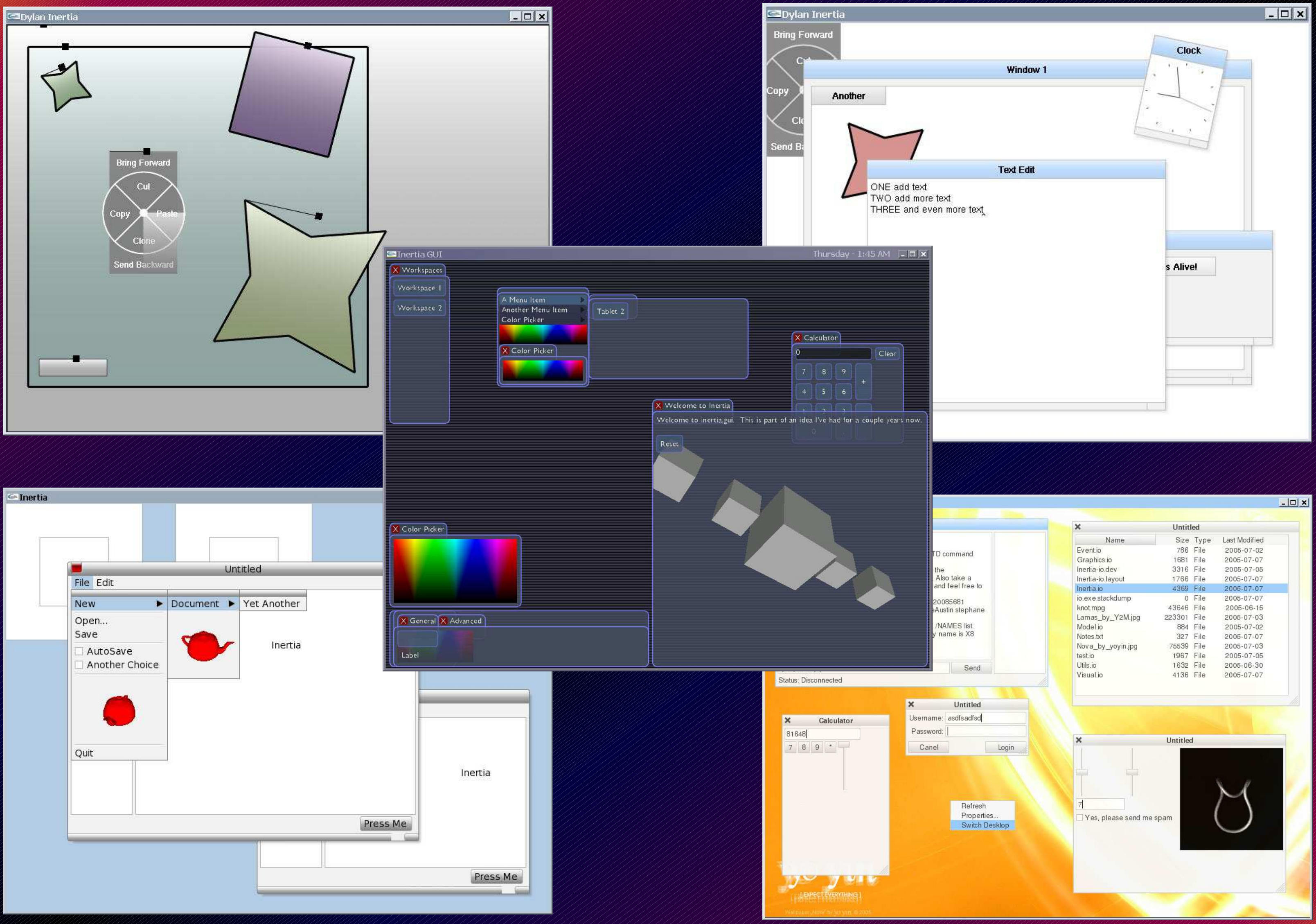
4/11/2006

Inertia

A simple, yet powerful
user interface environment

History

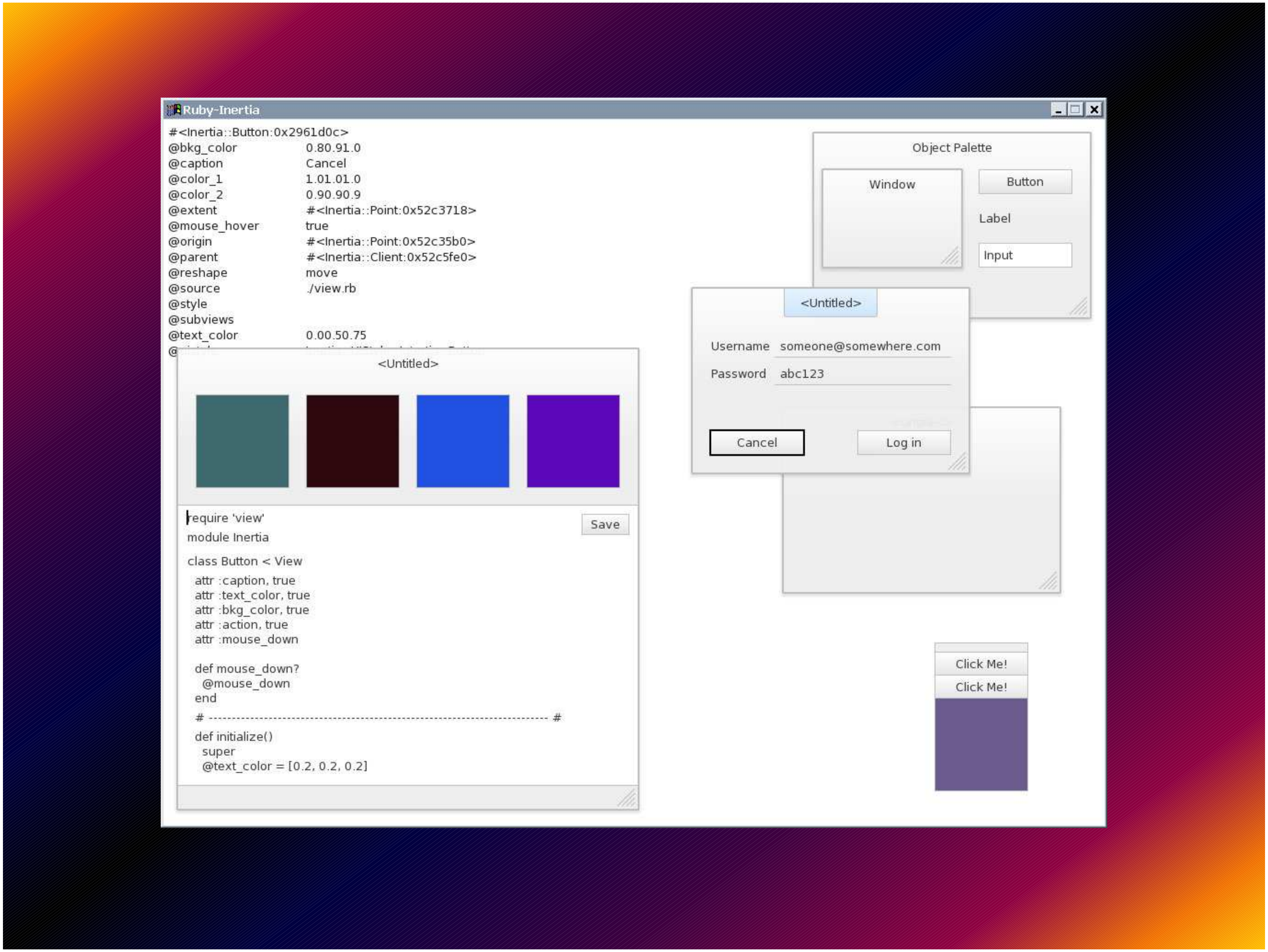
- Thoughts of a simpler Self or Squeak
- Prototypes have been written C++, Io, Dylan and now Ruby
- Tried a few ideas like ‘Magic Lens’, live editing and window grouping
- Looking for the most simple solution



ONE add text
TWO add more text
THREE and even more text

Welcome to inertia.gui. This is part of an idea I've had for a couple years now.

Name	Size	Type	Last Modified
Event.io	786	File	2005-07-02
Graphics.io	1681	File	2005-07-07
Inertia-io.dev	3316	File	2005-07-05
Inertia-io.layout	1766	File	2005-07-07
Inertia.io	4369	File	2005-07-07
io.exe.stackdump	0	File	2005-07-07
knot.mpg	43646	File	2005-06-15
Lamas_by_Y2M.jpg	223301	File	2005-07-03
Model.io	884	File	2005-07-02
Notes.txt	327	File	2005-07-07
Nova_by_yoyin.jpg	75539	File	2005-07-03
test.io	1967	File	2005-07-05
Utils.io	1632	File	2005-06-30
Visual.io	4136	File	2005-07-07



```
#<Inertia::Button:0x2961d0c>
@bkg_color        0.80.91.0
@caption          Cancel
@color_1          1.01.01.0
@color_2          0.90.90.9
@extent           #<Inertia::Point:0x52c3718>
@mouse_hover      true
@origin           #<Inertia::Point:0x52c35b0>
@parent           #<Inertia::Client:0x52c5fe0>
@reshape          move
@source           ./view.rb
@style
@subviews
@text_color       0.00.50.75
```

```
require 'view'
module Inertia
class Button < View
 attr :caption, true
 attr :text_color, true
 attr :bkg_color, true
 attr :action, true
 attr :mouse_down

 def mouse_down?
  @mouse_down
 end
 # ---------------------------------------------------------------------- #
 def initialize()
  super
  @text_color = [0.2, 0.2, 0.2]
```

Username someone@somewhere.com
Password abc123

Features

- No distinction between editing and runtime modes
- Live move, resize and reparenting of all visible objects
- Drag & drop between any objects
- Model view controller with UI styles

No Edit and Run

- To design, press [Alt] then
  - click = select View
  - drag = pick up, move or resize
  - release = drop / reparent
  - right button = meta menu
- [Shift] and [Ctrl] modifiers…

No Edit and Run

- [Shift] and [Ctrl] modifiers
  - [Alt] drag = move
  - [Alt] + [Shift] drag = copy
  - [Alt] + [Ctrl] drag = subclass
- To subclass Button, simply
  - drag a Button with [Alt] + [Ctrl]
  - A new source window will appear

Drag & Drop

- Drag and drop is simple:

  -
    ```
    def drag_accept( origin, object )
        object.instance_of? Color
    end
    ```

  -
    ```
    def drag_drop( object )
        @back_color = object
    end
    ```

UIStyles

- Defining a new style is simple:

  -
    ```
    class UIStyle::MyStyle
        class Button < UIStyle
            def draw_content( view )
                view.draw_rect( ... )
            end
        end
    end
    ```

Core Classes

- Point
- Event
  - MouseEvent, KeyEvent, ShapeEvent
- Font
- View
  - Screen, Window

Optimizations

- Font optimizations
  - Textured, uses glCallLists(), use draw_strings() to draw multiple lines
- Display lists for
  - Textured fonts, Window shadows
- Soon... textured Windows

Techniques

- Menu buttons, radiobuttons, etc.
  - Simply Buttons with ui style of ‘Menu’
- Event dispatching use of splat (*)
- Heavy use of setter methods
- Hierarchical event system
  - Screen handles grabbing, dragging

Implementation

- Uses SDL with OpenGL & FreeType
- So far development going smoothly
  - About 1200 lines of Ruby in 15 files
- What I like about Ruby

simple syntax	extensive library
-rprofile	setter methods

Places to Visit

- Ruby Inertia
  - http://www.mike-austin.com/inertia
- Self
  - http://research.sun.com/self/
- Squeak
  - http://www.squeak.org/