OpenEyes - Drawing

Editors: G W Aylward
Version: 0.9:
Date issued: 30 September 2010

Target Audience

General Interest	
Heathcare managers	
Ophthalmologists	
Developers	✔

Amendment Record

Issue	Description	Authors	Date
0.9	Draft	G W Aylward	30 Sep 2010

Table of Contents

Introduction

The use of drawings is widespread within ophthalmology, and provides a useful method of entering clinical information on a patient. Ophthalmic EPRs have hitherto made use of 'off the shelf' drawing or painting packages which, although they are capable of producing any drawing, are not designed with opthalmology in mind, and tend to be slow and awkward to use as a result. OpenEyes provides a dedicated drawing package with a built in knowledge of ophthalmology. It enables fast and intuitive entry of ophthalmology diagrams, combined with the ability to extract clinical information in text form, as well as diagnostic codes for direct entry into a table.

Embedding in a web page

In order to be used in a webpage, the EyeDraw.jar file should be available in the main htdocs directory (see 'Serving OpenEyes'). The associated icon and other graphics files should be available in a subdirectory named graphics.

The applet is included in HTML using the Applet element which includes a number of parameters as children, as in the following example;

```
<Applet name="MyApplet" archive="EyeDraw.jar" code="EyeDraw" width=500
height=530>
<Param name="toolBars" value="primarycare, retina">
<Param name="startActions" value="AddRightFundus, Lock">
<Param name="backgroundColourString" value="#F688F6">
<Param name="isEditable" value="true">
<PARAM name="side" value="R">
<Param name="mayscript" VALUE="true">
Your browser does not support Java or Java is disabled, so nothing is displayed.
</Applet>
```

The following table lists the available parameters that can be passed to the applet, with more detailed explanation following

Parameter	Function	Default
dataString	String containing doodle data	Empty
isEditable	Flag to indicate edit or display mode	FALSE
toolBars	Comma separated list of toolbars to display (if in edit mode)	Empty

Parameter	Function	Default
startActions	Comma separated list of toolbars to display (if in edit mode)	Empty
backgroundColourString	HTML colour string	Empty
side	Right or left	R’
mayscript	Parameter allowing scripting of applet by javascript	FALSE

The dataString parameter

This string includes the data necessary to reproduce a drawing. Each doodle is specified by a collection of comma separated values, with each value set being separated by a semicolon. The following is an example for a drawing containing a right fundus and one spot of cryotherapy;

```
"0,0,0,0,1.00,1.00,0,0,RightFundus,0;-93,-4,0,0,0.52,0.52,0,0,Cryo,1;"
```

Each doodle expects the values in order as follows;

Value	Explanation
originX	x coordinate of the origin of the doodle in ‘doodle space’
originY	y coordinate of the origin of the doodle in ‘doodle space’
apexX	x coordinate of the apex point of the doodle in ‘doodle space’
apexY	y coordinate of the apex point of the doodle in ‘doodle space’
scaleX	Scaling factor along the x axis
scaleY	Scaling factor along the y axis
arc	Angle of arc (for doodles that use it) in degrees
rotation	Angle of rotation of doodle in degrees, positive clockwise
subclass	The unique name of the doodle subclass
order	The back to front order, zero being at the back

The isEditable parameter

This parameter is a boolean flag (true or false) indicating which mode EyeDraw should use. In the absence of this parameter, EyeDraw defaults to false (ie display mode). When in Edit mode, toolbars are displayed, and the drawing can be edited. In display mode, a smaller size is used for display, no toolbars are shown, and the drawing does not respond to mouse or keyboard actions.

The toolBars parameter

This parameter (ignored in display mode) is a comma separated list of doodle toolbars which are displayed in edit mode. It can be used to ensure that only relevant doodles are available for any particular drawing. For a list of currently available toolbars, please consult the OpenEyes website.

The startActions parameter

This parameter is a comma separated list of actions to take on launch of the applet. For example, startActions can be used to display a doodle in the background and lock it, so the user can then add doodles on top of it.

The backgroundColourString parameter

This is an HTML colour string which is hex encoded RGB (for example #F6F6F) which allows the applet to blend into the background of the page in which it is displayed.

The side parameter

This is a single letter ‘R” or ‘L’ indicating the side of the drawing. This determines how certain aspects of doodles are drawn (eg temporal or nasal)

The mayscript parameter

This is a standard applet parameter which, when true, allows javascript to call functions in the applet.

Saving drawing data

Saving doodle data is controlled by the host page, rather than the applet itself, since the user may choose to cancel rather that save their work. A combination of hIdden input elements and javascript is used to save the contents of the doodle when the host form is submitted. The following examples are taken from the OEExaminationEventClass The hidden elements are as folllows;

```
<input type="hidden" id="isDrawingData" name="isDrawingData" value="FALSE" />
```

This element is a boolean flag to indicate whether there is data from any applet on the page which needs saving.

For each applet (there is no limit to how many can be displayed on a page) there is an additional hidden element which is initially empty, and is used to store the applet data. The following is an example for the right anterior segment drawing;

```
<input type="hidden" id="RASDataString" name="RASData" value="" />
```

The following element displays a save button and which when pressed, triggers a javascript called save().

```
<input type="button" name="button" class="submit" value="Save" accesskey="S" title="Save (CTRL-S)" onclick="save()"/>
```

Adding new doodles to Eye draw

It is likely that users of OpenEyes may wish to add new doodles in the future, and this section is a step by step guide describing the steps required. A basic knowledge of the Java programming language is required, along with a suitable development environment (see Appendix 1 for further information).

1. Decide on a name

A one word unique name is required for the doodle in order to identify it within the software, and also as a key when storing it in the database. The maximum length of the name is 20 characters, and it must be made up of alphanumeric characters but no white space. As a Class it should start with a capital letter (see OpenEyes style guide). For the purposes of this example we are going to add a small blot haemorrhage to the first medical retinal toolbar (MR1). The doodle will be called 'BlotHaem'. Note that in Java names are case sensitive.

2. Create a subclass

In your Java editor, copy the Template subclass and paste it back into the text, then change the two occurrences of the word 'Template' to 'BlotHaem'.

3. Create an action

Repeat the steps in section 2 for the AddTemplateAction class. That is, copy and paste, then replace three occurrences of the string 'Template' with 'BlotHaem'.

4. Testing the drawing

Add the following line in the HTML of the page containing the applet.

```
<Param name="startActions" value="AddBlotHaem">
```

This will result in the doodle being drawn when the applet launches so that the drawing can be adjusted and refined without the need to create a tool bar button.

5. Add the action

In the section of the file entitled 'Add Doodle Actions', add the following line to the end of the section;

```
protected AbstractAction addBlotHaemAction = new
AddBlotHaemAction(this.createButtonIcon("BlotHaem.gif"), "Blot Haemorrhage");
```

In the section specifying the buttons within the MR1 toolbar, add the following line;

```
doodleToolbarAS.add(nonFocusableButton(addBlotHaemAction));
```

If you want a keyboard shortcut to call the doodle, add the following line;

```
inputMap.put(KeyStroke.getKeyStroke(KeyEvent.VK_H, InputEvent.ALT_MASK),
"BlotHaem"); actionMap.put("BlotHaem", addBlotHaemAction);
```

Adding a toolbar

Find the section of the source code entitled "// Load drawing toolbars" and add the following code where

```
    // PrimaryCare tool bar
    if (toolBars.indexOf("primarycare") > -1)
    {
        // Set up doodle toolbar for Primary Care
        JToolBar doodleToolbarAS = new JToolBar();
        doodleToolbarAS.setFloatable(false);
        doodleToolbarAS.setAlignmentX(LEFT_ALIGNMENT);

        // Create and add doodle buttons
        doodleToolbarAS.add(nonFocusableButton(addAntSegAction));

        // Add to layout
        toolbarPane.add(doodleToolbarAS);
}
```