Multi-user Infinite Canvas Thingy

October 3, 2010

Client Specification

- The root of the client program is the Client class. This is the application/applet itself. Other than initializing the other components and coordinating communication between them. It is also responsible for setting the location of the canvas, either through the initial connection or through a jump command.

- There is also a ClientState class. A single ClientState object will be initialized for each instance of the program. The ClientState object will be used to store the shared state. Any field that needs to be accessed and modifed by multiple classes should be placed in here. This includes but is not limited to the currently selected tool and color, the current location of the canvas, and the persistent connection to the server. The use of a singleton for shared state will reduce the coupling between all the components.

- The Client will have three components embedded within it: the Toolbox, the Canvas Control Panel, and the Canvas.

- The Toolbox will contain a series of Tools. Each Tool (which extends mict.tools.Tool) shall be defined in Jython so that we can serialize the class and transmit it from the server to the client. This way, every client will support every tool the server has and we don't have to worry about a client possessing a tool that the server doesn't know how to process.

  – We will need to examine the various techniques for serializing classes to ensure that this is possible. Based on a preliminary analysis, this can be obtained with Java's ObjectOutputStream. These tools will implement the functional requirements specified in section 3.2.2 of the Software requirements specification.

  - Each tool will know how to serialize its commands so that we can send them to the server. They will also know their own name, associated icon, and tooltip. The tool needs to be entirely self-contained so the client doesn't need to know anything about how many tools there are, or what they do.
  - The interface specification for the Tool is attached.

- For each Tool, we will make a ToolButton. ToolButton is a subclass of JButton. It has an ActionListener that sets the associated tool as the active tool within ClientState when the button is pressed.

- The Canvas shall be a blank JPanel.

  - The Canvas shall have a single MouseListener. The MouseListener will wait for MousePressed, MouseDragged, and MouseReleased events and dispatch those events to the currently selected tool. Once the MouseReleasedEvent has been fired, the Canvas will request a serialized representation of the tool's action, and will dispatch it to the server.

- There will be a single Java class and a single Python module (JythonBridge.java and javabridge.py) to coordinate communications between Jython and Java components. Any Java component that wants to access something from Jython should go through JythonBridge. Because of the way Jython works, Python components can access the Java libraries directly if they need to extend them (for instance, extending Tools) but if they wish to get an instance of a Java object, they should go through javabridge

- When the program is first launched, the Toolbox will be empty and the canvas will have 2 JTextFields, a JPasswordField, and a JButton. The first JTextfield will be used to enter the server to connect to. The second will be the username, and the JPasswordField will be used to enter the password. The JButton will tell the client to connect to the server specified with the given log-in credentials. If login is successful, the fields will be removed. This implements the Canvas.Connect requirement mentioned in section 3.1 of the SRS document.

  - When the user presses the log-in button, the ClientConnection will be created. Upon successful connection, the Client will create the ClientState object, get the Tools and create the ToolButtons, and then add the Canvas and set the associated Graphics in the ClientState.
  - Depending on the user's permissions and location, the Canvas Control panel may also be populated to hold the interface for section 3.3 of the software requirements. For Admins, the panel will also contain the controls for the Canvas.User.Kick, Canvas.User.Ban, and Canvas.User.Pardon options from section 3.4 of the functional requirements.

- The Canvas Control panel will contain the buttons that allow users to acquire and release sections of the canvas, provided the conditions specified in section 3.3 are met.
  - The options in this panel will be implemented as Tools if possible, and will only be special-cased if the Tool interface does not permit the option to act as intended
  - These controls change as the user moves over the canvas. The pan and jump commands will include sections to modify this panel as needed.