Multi-user Infinite Canvas Thingy

Donald Huckle, Benjamin Kaplan, Rob Weiser, Mark Wyzyrkowski

October 4, 2010

1 Tool

The Tools are the heavy lifters of the program. Each Tool is self-contained as is responsible for drawing itself on the canvas. They are also responsible for drawing themselves on the final canvas

1.1 Serializing the Tools

Each of the Tools will be stored on the server and transmitted to the Client upon log-in. To facilitate this, each of the Tools should be implemented in Jython. Preliminary research shows that the Jython classes can be serialized using Java's ObjectOutputStream but this has not yet been tested.

1.2 Mouse Events

The Tool contains MousePressed, MouseMoved, and MouseReleased methods that are called when the mouse is pressed, moved (while pressed), and then released while this Tool is active. Each of these methods returns a String, which will be transmitted to the server in order to apply the change to the final canvas. These methods are passed the canvas's Graphics object and the Point on the canvas where the event was fired from.

1.3 drawFromString

Here, the Tool will be passed a String generated by the tool and a Graphics object to draw the figure on. The changes to the canvas generated by this

method should be identical to the changes generated by the tool on the client side.

1.4 Tool Properties

Each tool will need to know it's name, the icon to represent it, the tooltip, and a unique identifier that represents the tool. The icon and tooltip will be used on the client-side to represent the tool to the user.

2 Client Program Specification

This section describes the layout of the client program. It will discuss all the components of that program and how they interact with each other. There are 5 main parts to this: the Client, the ClientState, the Toolbox, the Canvas Control Panel, and the Canvas.

2.1 Client

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. The Client will hold a ClientState object that stores the shared settings. In addition, the GUI will consist of 3 components : the Toolbox, the Canvas Control panel, and the Canvas

2.2 ClientState

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.

2.3 Toolbox

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.

2.3.1 ToolButtons

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.

2.4 Canvas

The Canvas is the section of the Client program that will display a section of the overall canvas. The user will draw on this JPanel in order to modify the canvas, and other user's modifications will be updated here.

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. As these methods are called, the Canvas shall transmit the serialized commands to the server.

2.5 Java-Python Bridge

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

2.6 Canvas Control

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.
- Users will only be presented with options appropriate to their permissions. If they do not have permission to acquire sections for instance, they will not see that button presented to them. If they have permission but are unable to acquire it (for instance, if someone else owns it), the button will be inactive.
- Admin users will see the buttons to trigger the Canvas.User.Kick, Canvas.User.Ban, and Canvas.User.Pardon options from section 3.4 of the requirements here as well.

2.7 Launching MICT Client

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.

3 Client-Server Specification

The client server interaction is based on a TCP connection between the server and the client. In the communication there are three major players. The Server's Waiter thread, The Client's client connection thread and the Client's GUI which will use event handlers such as mouse-clicked and mouse-dragged to handle relevant actions. When the client is first run the GUI will be created.

3.1 Connecting to the Server

The Client's GUI will have an event handler that creates the client connection that connects to the server when the login button is pushed. The connection to the server will be persistent; Once established the waiter thread on the server's side and the Client Connection thread will communicate.

3.2 Client-Side Generated Communication

When an event handler is called to do some operation it will often result in the event handler sending a message to the client connection thread. This thread will then pass the information on to the server. Operations that will cause communication to be generated by the client include: mouse clicked, mouse dragged, mouse released. These events have event handlers associated with them. The event handlers will send the positions of the mouse as well as the tool currently in use. Additionally data from client state may also be transmitted. It is important to not that the tool object will be used to specify what information will be sent to the server.

3.3 Server-Side Generated Communication

When the Server has some action that it needs the client to take, such as correcting the client's canvas info so that it no longer includes an illegal draw operation(e.g. drawing on a protected part of the canvas); The server will initiate communication with the client. When this happens the information is passed from the server to the client connection thread. The client connection thread will send the information to the client class. Because the Swing architecture is not thread safe the client class will have the GUI invoke the operation later by using the invokeLater(Runnable doRun) method.