Developing Plugins for AgentSheets

Thought Amplifier.

AgentSheets®

Developing Plugins

AgentSheets 2.1

Alexander Repenning and Ronald Sudomo

1 Philosophy 2
2 Writing Plugins 2
2.1 Defining Actions 2
2.2 Defining Conditions 3
3 Compiling Plugins 4
4 Using Plugins 4
5 Library Dependencies 4
6 Dialog Plugin: A Complete Example 5
6.1 Listing: Dialog Plugin 7

1 Philosophy

The main objective of this plugin architecture is ease of use. The most elaborate architecture will be of little value if it comes with a steep learning curve, which, in the end, only few developers will be able to master. Plugins should be simple to build and simple to distribute. Rather than providing developers with an intricate API including complex class hierarchies and interfaces, the AgentSheets plugin architecture will extract the necessary information from your code. In other words, the architecture adapts to the developers not the other way around. Using introspection mechanisms AgentSheets will analyze plugins and create Visual AgenTalk interfaces of relevant methods found.

2 Writing Plugins

To build a plugin you need to write Java code and you need to have a Java compiler. The source of your plugin file, like any other Java file, defines a Java class. The *AgentSheets Plugin Manager* (APM) will be able to recognize any compiled Java file (a .class file) as a plugin if:

- the plugin file is in the “Plugins” package
- the class definition is public

Your class definition may include any number of method definitions. To create new Visual AgenTalk commands your methods need to have a certain signature. AgentSheets will analyze plugins and look for these signatures. If AgentSheets finds these signatures it will:

- ***create new Visual AgenTalk commands*** including a command GUI and add them to the command palettes. These commands can be used in any project. Like any other command you can use them by dragging them from the palettes into your agent behavior editor window.
- ***extend the Visual AgenTalk compiler***. Your methods will turn into new actions and conditions that each agent can use.

2.1 Defining Actions

An action has the following signature:

```
public static void commandName_Action (parameters)
```

commandName is the name of the action as it will show up in the action palette

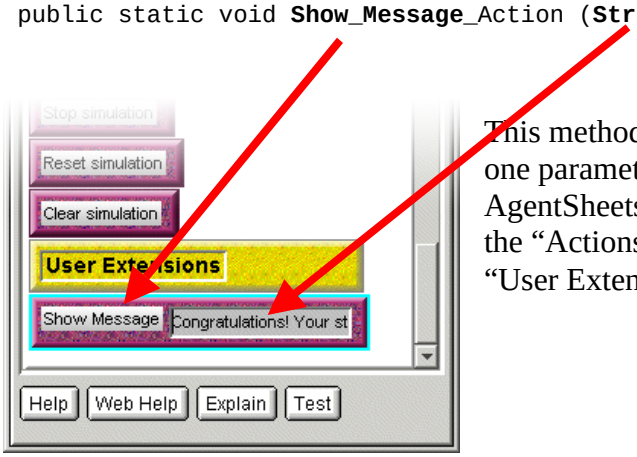
parameters: the method can have zero or more parameters of type `int`, `float` or `String`.

each `int` or `float` parameter will be mapped to a Visual AgentTalk FormulaType. A FormulaType appears as editable text box to contain Visual AgenTalk formulae. When AgentSheets executes your new action containing FormulaTypes it will compute that value of each formula first and then call your method with the result of the evaluated formulae. For instance, an action including a FormulaType set to “radius * 2 * pi” would first compute the value of this formula which is the result of multiplying the value of the agent property “radius” with the constant “pi” and the number 2 and then use this value as single float (or int) to be passed to your Java method.

each `String` parameter will be mapped to an AgentSheets StringType.

Example:

```
public static void Show_Message_Action (String message)
```



This method defines a "Show Message" action. It has one parameter of type `String` that is mapped to an AgentSheets StringType. The new action is added to the "Actions:" palette where it is inserted after the "User Extensions" tag.

2.2 Defining Conditions

Conditions, unlike actions need to return a boolean value indicating if the condition is true. A condition has the following signature:

`public static boolean` *`commandName`*`_`**`Condition`**`(`*`parameters`*`)`

commandName is the name of the condition as it will show up in the condition palette

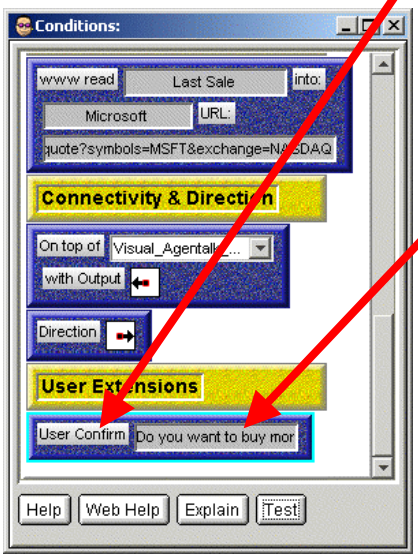
parameters: the method can have zero or more parameters of type `int`, `float` or `String`.

each `int` or `float` parameter will be mapped to a Visual AgentTalk FormulaType. A FormulaType appears as editable text box to contain Visual AgenTalk formulae. When AgentSheets executes your new condition containing FormulaTypes it will compute that value of each formula first and then call your method with the result of the evaluated formulae. For instance, a condition including a FormulaType set to "radius * 2 * pi" would first compute the value of this formula which is the result of multiplying the value of the agent property "radius" with the constant "pi" and the number 2 and then use this value as single float (or int) to be passed to your Java method.

each `String` parameter will be mapped to an AgentSheets StringType.

Example:

```
public static boolean User_Confirm_Condition (String question)
```



This method defines a “User Confirm” custom condition. This condition gets added to the Conditions palette. It has one parameter of type String that will be mapped to an AgentSheets StringType.

3 Compiling Plugins

Once you have written the code, you can compile it like any other Java program. You can even write your plugin in C/C++, but you have to provide a Java wrapper for it. The Java version supported is the same as the one used in the AgentSheets application (currently Java 1.3.1).

4 Using Plugins

To use plugins put them into the “Plugins” folder of the AgentSheets application. Put the binary files (.class), not the source files, into that folder. When AgentSheets gets launched it will search for files in the “Plugins” folder, will introspect files and will add new commands to the actions and conditions palette. Now you can open or create projects and use your new commands.

When you turn your project into a stand-alone applet using the Ristretto button, Ristretto will automatically copy the necessary plugin files into your applet.

When you share a project that makes use of a plugin you have to make sure you also include the required plugins. Without these plugins present loading a project will result in a Java error.

5 Library Dependencies

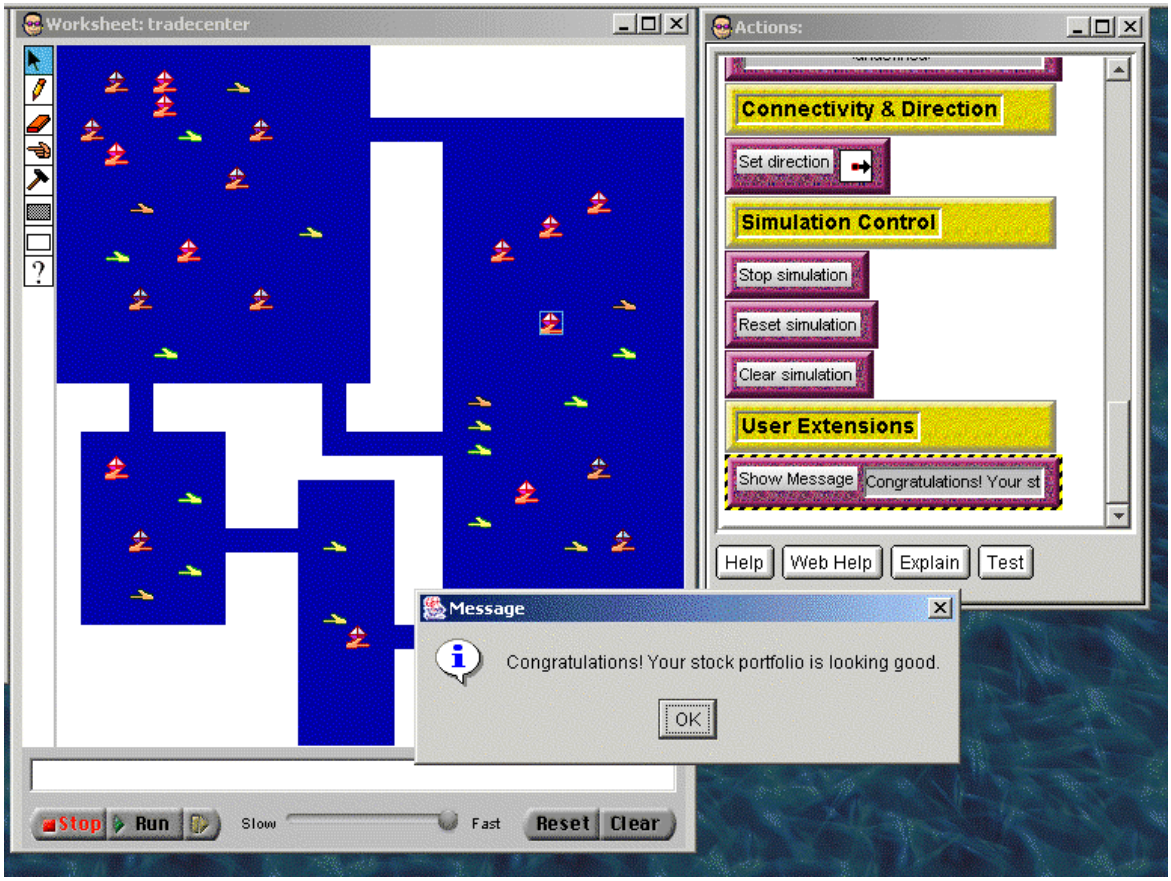
AgentSheets simulations can run in any browser (Windows, MacOS, and Unix) as long as the browser is at least Java 1.1 compliant. The AgentSheets simulation player software, called µAgentSheets, is based on the AWT framework that is part of Java 1.1. Problems can occur if your plugin requires additional libraries. If these libraries are not found on the client trying to run your plugin then Java will crash. Ristretto will copy the binary version of your plugin files but it cannot track library dependencies.

A simple example of this problem is when you are building a plugin based on Swing; the people playing your simulation will need to have Swing installed as well. It is hard to predict what libraries users will have. Depending on their browser and platforms they may or may not have Swing installed. For instance, Explorer 5 on MacOS 9 is Java enabled but does not include Swing. Windows XP default install does not include Java at all. Unfortunately, Java error messages that are created in the case of a missing library are typically not very informative and will lead most users to simply give up trying to run your applets. Very little can be done in these cases. However, if you try to make applets accessible to a wide range of users who may be using all kinds of browsers and platforms, the dependency on libraries should be minimized. For instance, you may want to spend the extra energy to write your plugin using AWT functions instead of the more sophisticated Swing or Java2D functions – if possible.

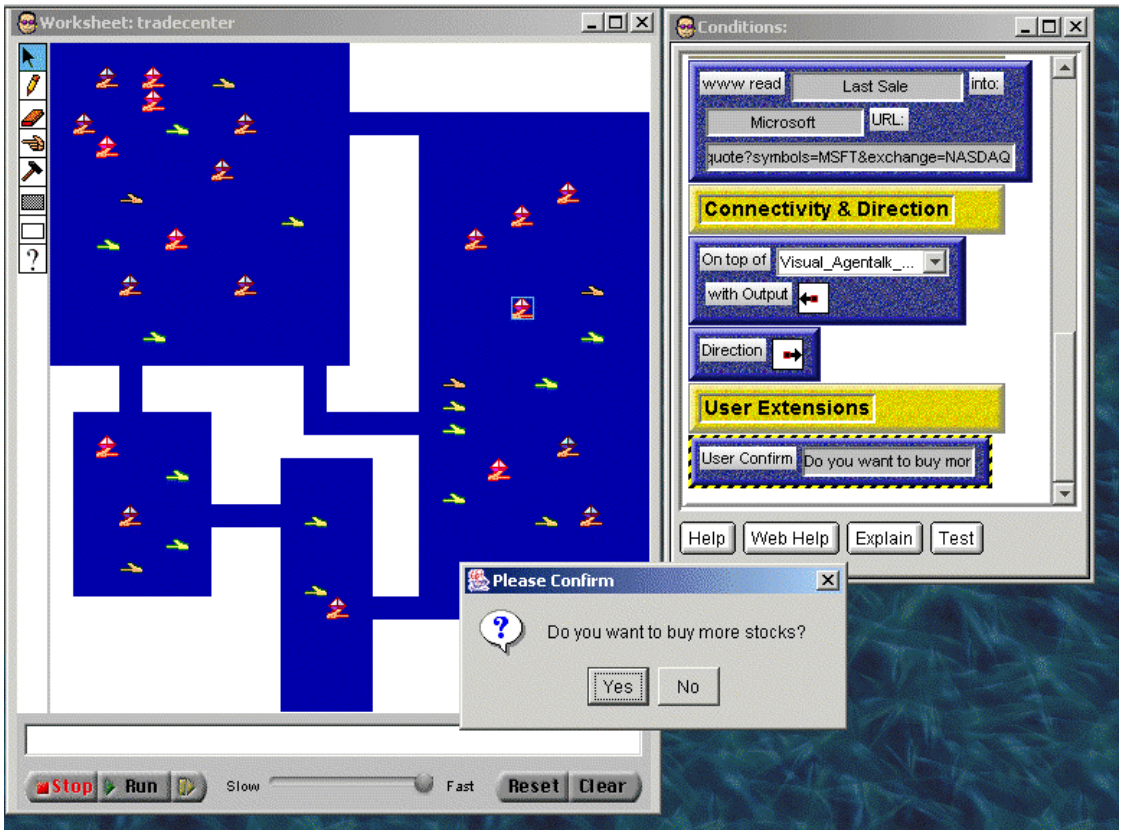
6 Dialog Plugin: A Complete Example

The two code examples of an action and a condition definition are part of a complete example (code is listed in the next section) of extending AgentSheets with handy dialogs.

The Show Message action will pause the simulation, bring up a modal dialog box containing a message string, and wait for the user to OK. Selecting it in the actions palette and then hitting the Test button can test this new action:



The User Confirm condition allows users to control a simulation through yes/no questions. Again the new condition can be tested immediately using the Test button:



Below is the complete listing of the plugin.

6.1 Listing: Dialog Plugin

```
/*
 * An AgentSheets plug-in example.
 *
 * Copyright (c) 2001-2002 AgentSheets, Inc. All Rights Reserved.
 *
 */

package Plugins;

import javax.swing.*;


/**
 * This simple AgentSheets plug-in adds a custom action to show a message
 * dialog and a custom condition to request user confirmation.
 *
 */
public class DialogPlugin {

    /**
     * This custom action shows the specified message to the user.
     *
     * @param message  the message to show
     */
    public static void Show_Message_Action(String message) {
        String title = "Message";
        JOptionPane.showMessageDialog (null, message, title,
                                       JOptionPane.INFORMATION_MESSAGE);
    }

    /**
     * This custom condition shows a confirmation dialog to the user.
     *
     * @param question  the question that needs confirmation
     * @return true if the user clicked "Yes"; false if the user clicked "No"
     *         or close the dialog
     */
    public static boolean User_Confirm_Condition (String question) {
        int option;
        String title = "Please Confirm";

        option = JOptionPane.showConfirmDialog(null, question, title,
                                               JOptionPane.YES_NO_OPTION);
        return (option == JOptionPane.YES_OPTION);
    }

} // end class DialogPlugin
```

Sim Whatever.

AgentSheets®

Corporate Headquarters
AgentSheets Inc.
6560 Gunpark Drive
Boulder, CO 80301
USA
http://www.agentsheets.com
Tel: (303) 530-1773
Fax: (303) 530-3018

European Demo Center
Fraunhofer-Institute for Computer Graphics
Rundeturmstraße 6
D- 64283 Darmstadt
Germany
http://www.igd.fhg.de
Tel: + 49 6151 155 0
Fax + 49 6151 155 199