GBASpriteEditor - Plug-in development document

Version 25/11/2003

Please donate if you plan to use this editor for commercial development
gbaspriteeditor.gbadev.org

About this document

GBASpriteEditor currently supports plug-in development for additional tools, import routines and export routines. This document goes some way towards describing how to develop additional plug-ins. However, a knowledge and understanding of the Java programming language is assumed.

In order to develop plug-ins for the GBASpriteEditor you will need to set your classpath variable to find the GBASpriteEditor.jar file.

Plug-in conventions

Each set of plug-ins must be contained within a Jar archive. A text file called plugins.txt is required to specify each plug-in class. This file must be located in the root directory of the Jar archive. The default plug-in set has a plugins.txt file with the following contents :

```
org.gbadev.gbaspriteeditor.defaultpluginset.GBADefaultSpriteImport
org.gbadev.gbaspriteeditor.defaultpluginset.GBASoSpriteAnimationExport
org.gbadev.gbaspriteeditor.defaultpluginset.GBASoImageExport
org.gbadev.gbaspriteeditor.defaultpluginset.GBADefaultPaletteExport
org.gbadev.gbaspriteeditor.defaultpluginset.GBAPencilTool
org.gbadev.gbaspriteeditor.defaultpluginset.GBARectangleSelectionTool
org.gbadev.gbaspriteeditor.defaultpluginset.GBAMoveSelectionTool
org.gbadev.gbaspriteeditor.defaultpluginset.GBAColorPickerTool
org.gbadev.gbaspriteeditor.defaultpluginset.GBAFloodFillTool
```

Note that the tools are ordered in the order they should appear on the tool bar.

In order for a user to install a new plug-in set they must copy the plug-in set Jar archive into the plug-ins folder. It is also possible to create subdirectories in the plug-ins folder to place Jar archives in. This opens the possibility of storing any readme files or other documentation along side the plug-in.

Try and keep any files which the plug-ins are dependent upon (e.g. tool images) inside the Jar file.

Tool plug-ins

All tools must extend the abstract class GBATool. None of the plug-ins use a constructor. Instead an init method is called :

```
public GBATool init(GBASpriteEditorFunctions guiFrame)
```

When overriding this method always call `super.init(guiFrame);`

Each GBATool has a reference to the variable frame which is an instance of the abstract class GBASpriteEditorFunctions which is responsible for much of the sprite editor's general functionality.

```
protected GBASpriteEditorFunctions frame;
```

To find out more about the GBASpriteEditorFunctions see the API which is available along with the source code.

Each tool has it's own options panel to the right of the tool selection bar. If you wish to add options for your tool create a JPanel containing the relevant components within the init method and return it with this get method :

```
public JPanel getToolOptionsPanel()
```

To give your tool an icon, implement this method :

```
public ImageIcon getIcon()
```

and return the relevant ImageIcon e.g. :

```
return new ImageIcon(GBAColorPickerTool.class.getResource("colorpicker.png"));
```

GBATool implements both the MouseListener interface and MouseMotionListener interface by default and when selected a tool will become a listener for the edit pane. In order to find out where upon the current layer a mouse event occurred, use something similar to the following code :

```
int x = frame.getEditPane().findXOnImage(e.getX());
int y = frame.getEditPane().findYOnImage(e.getY());
```

To find out if the point is within the bounds of the layer :

```
boolean withinBounds = (!(frame.getEditPane().isWithinImage(e.getX(), e.getY())));
```

To find the palette index of the color the mouse event occurred at :

```
int colorIndex = frame.getLayersPane().getEditLayer().getPoint(x, y);
```

To find the palette index of the selected color :

```
int selectedIndex = frame.getPalettePane().getEditColorIndex();
```

To draw a point, set the point within the current edit layer then call requestDrawPoint :

```
frame.getLayersPane().getEditLayer().setPoint(palIndex, x, y);
frame.getEditPane().requestDrawPoint(new GBADrawPoint(pal.getColor(palIndex),
                                                      x,
                                                      y));
```

This way the editor only redraws the point edited. To redraw the entire image call :

```
frame.getEditPane().updateAllLayers();
frame.getEditPane().updatePane();
```

However, this can be very slow so only do this if it is really needed.

To find out more on tool plug-ins see the API reference and look at the source code for the

default plug-in set (org/gbadev/gbaspriteeditor/defaultpluginset).

Import / export plug-ins

There are three types of import / export available, sprite import (GBASpriteImportPlugin), palette export (GBAPaletteExportPlugin) and sprite export (GBASpriteExportPlugin). Below is a description of the GBASpriteImportPlugin. Essentially all three work in pretty much the same fashion so see the API to find out more about the export plug-ins.

To create a plug-in to import sprites extend the abstract class GBASpriteImportPlugin which can be found in the package org.gbadev.gbaspriteeditor.importplugin.

Implement the init method instead of using a constructor. This method will be called when the plug-in has been found.

```
public GBASpriteImportPlugin init()
```

To set the name of the plug-in which appears in the combo box when selecting the plug-in to use, override the toString method and return the name of your plug-in.

Each plug-in has it's own preference panel. Override the method below and return an instance of a JPanel containing your plug-ins preferences.

```
public JPanel getImportPreferences()
```

In order to update the panel when the plug-in becomes selected, override the updateImportPreferences method :

```
public void updateImportPreferences(GBASet set,
                                    String spriteName)
```

When the user has selected the file to import and clicked on the Done button, the method below is called :

```
public GBASprite importSprite(File inputFile,
                              String spriteName,
                              GBASet set,
                              GBAPalette pal) throws FileNotFoundException,
                                                     IOException
```

Implement this method to create your own import routine. Throwing the exceptions FileNotFoundException and IOException will display a pop-up warning the user.

From there on in it is pretty much up to you how you implement your plug-in which should expand the editor to fit individual user needs. Note that the implementations of the default plug-ins are generally no more than a hundred lines or more code depending on the complexity of the preference panes.

Hopefully alongside the API and the source code for the default plug-in set there should be enough detail in this document to get started. If you have any useful information or pointers you wish to add to this document contact me at the email address on the contacts page on the GBASpriteEditor website.