

Gameloft Java Emulator

Table of Contents

1.Features.....	2
2.What emulator can't do.....	2
3.Starting the Emulator.....	2
3.1Launch Pad.....	2
3.2Command line arguments.....	3
3.3Custom Devices.....	4
3.4Multi Jar.....	5
3.5Other options.....	7
4.Using the emulator.....	7
4.1Keypad.....	7
4.2Watches.....	8
4.2.1 Custom editors.....	10
4.2.2 Displaying 2D arrays.....	10
4.2.3 Exporting data.....	10
4.3Memory View.....	12
4.3.1 Memory.....	12
4.3.2 Classes.....	13
4.3.3 Instances.....	13
4.3.4 Images/Sounds.....	13
4.3.5 Scene Graph.....	13
4.4X-RAY View.....	14
4.4.1 Using X-RAY with 2D.....	14
4.4.2 Using X-RAY with 3D.....	15
4.5Profiler.....	16
4.6Monitors.....	17
4.7Methods.....	17
4.8Resources View.....	19
4.9Display Infos.....	20
4.10Display Options.....	20
4.11Capture Screen.....	21
4.12Video Capture.....	21
4.13Suspend/Resume.....	21
4.14Pause/Step.....	21
4.15Network availability.....	21
4.16Special Commands.....	22
4.17Using WMA to Send/Receive SMS.....	22
4.18Using JRat Profiler.....	24
4.19BeanShell Console.....	26
4.20Using obfuscated jars.....	27
5.DOJA.....	27
5.1Implementation.....	27
5.2Raw Files.....	28

5.3Scratch Pad.....	28
6.Using remote debugger.....	28
6.1Preparing the code.....	28
6.2Enabling the debugging agent in emulator.....	28
6.3NetBeans.....	28
7.FAQ.....	30
8.Version History.....	31
9.Bug Report and Suggestions.....	31

1. Features

- Stand alone application (no dependency on third party APIs)
- Support almost any phone
- Support MIDP1, MIDP2, Nokia, Spint APIs (all included in the jar)
- Integrated Zoom
- Mouse cursor position/rectangle dimension
- View/Edit all variables of all classes, instances and arrays
- View all loaded and drawn/never-drawn images at any time and actual regions displayed
- X-RAY View to display all drawing operations for each frame
- Basic profiler to view # of drawImage/frame, # of drawRegion/frame, etc
- Suspend/Resume
- Record/Play macros of key sequences
- Record movie sequences in animated GIF files
- View exact line of Exception stack trace (when not obfuscated and not preverified)

2. What emulator can't do

Memory usage is totally irrelevant since the emulator is also in java, so don't refer to Runtime memory functions: freeMemory() and totalMemory(). Hopefully, it's possible to get an approximation of memory used by the MIDlet in the MemoryView.

The emulator is constantly evolving and some packages or behaviors may not be implemented. Also, many sound formats are not supported, only MIDI et WAV.

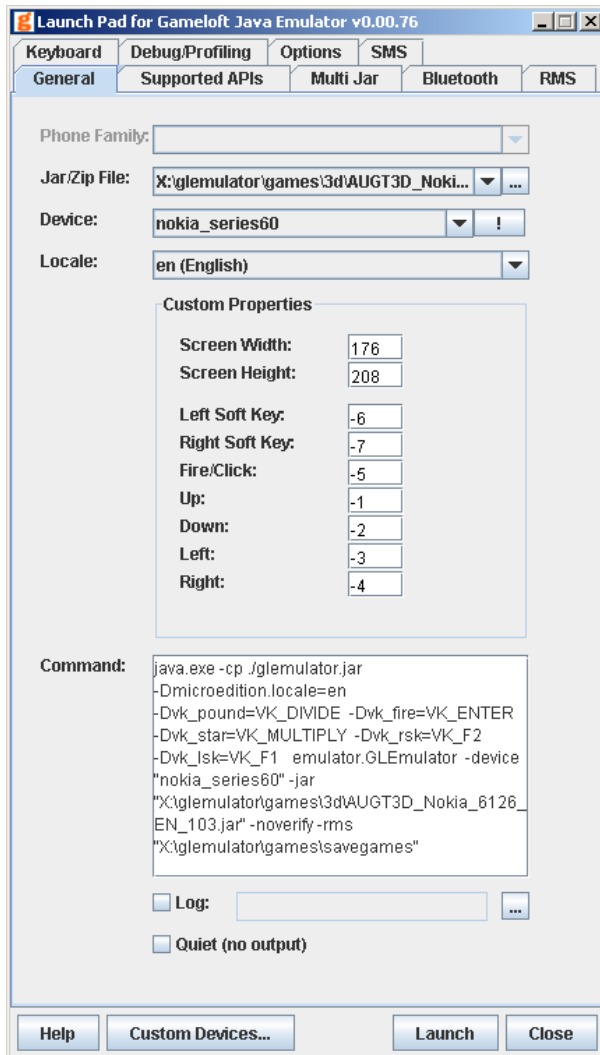
3. Starting the Emulator

3.1 Launch Pad

Use the launch pad to start a MIDlet without using the command line. The Launch Pad let you choose many options:

- Jar file or Zip file (Zip file should be a release containing one jar and one jad)
- Device model and properties (screen size and keys)
- Supported APIs (will only prevents warnings when some API-specific classes/methods are used)

- Multijar let you run multiple instances of a game to test localization
- Bluetooth parameters to test interactivity
- RMS maximum size
- Keypad let you customize keyboard keys for softkeys, fire, etc
- Monitoring options (“released images” and “method invocations”)



3.2 Command line arguments

Here are the arguments for Java:

```
-cp ./glemulator.jar emulator.GLEmulator
```

Argument	Description
-cp ./glemulator.jar	Java class path. Emulator jar must be specified. Essential Note that the midletjarfile is not anymore in the classpath.

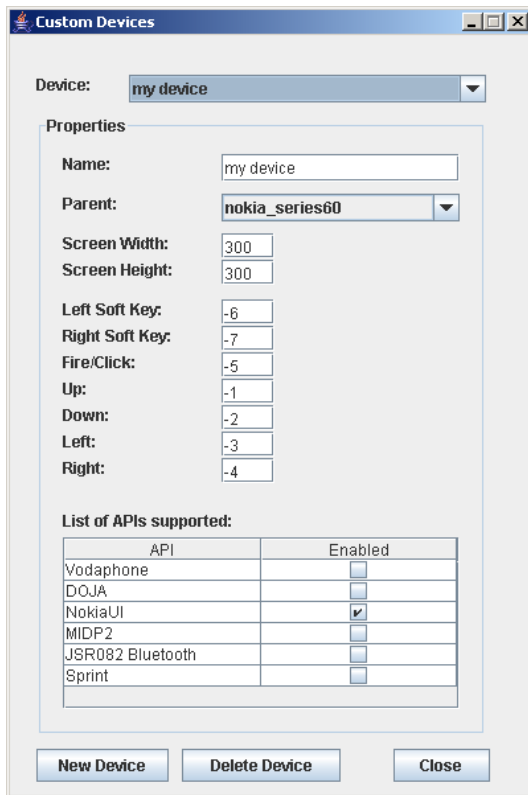
emulator.GLEmulator	Entry class to start the emulator.
---------------------	------------------------------------

Here is the list of available arguments:

Argument	Description
-device <device>	Name of the device, such as <i>motorola_triplets</i> . Essential.
-jar <midletjar>	Path to the jar file containing the MIDlet. Essential.
-devicefile <deviceFile>	Path an external xml file containing device informations. Optional.
-ppfolder <ppfolder>	Path to a folder containing pre-processed java files with javapp.exe to display original line when an exception occurs. Optional.
-rms	Path where the RecordStore file will reside Optional
-cp	Class path where the classes can be found. Classes will be loaded from this location instead of the jar file. Optional, for use with a debugger like Eclipse. <i>-midletclassname</i> needs to be specified.
-midletclassname	Name of the MIDlet class to load. Optional, need <i>-cp</i> argument to be specified. For use with a debugger like Eclipse.

3.3 Custom Devices

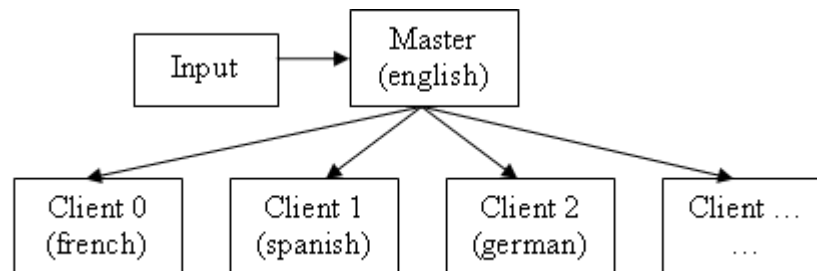
If a device does not exist in the Launch Pad list, you can add a custom device. In the Launch Pad, click on the “Custom Devices...” button at the bottom, it will display this window:



Click on “New Device” to create a new device and make sure to change the name. Select the “Parent” device from which the properties will be inherited. When the “Close” button is clicked, the file “customdevices.xml” will be created in the current folder. Only the properties that override the parent properties are stored. The Launch Pad device list and properties will then be updated.

3.4 Multi Jar

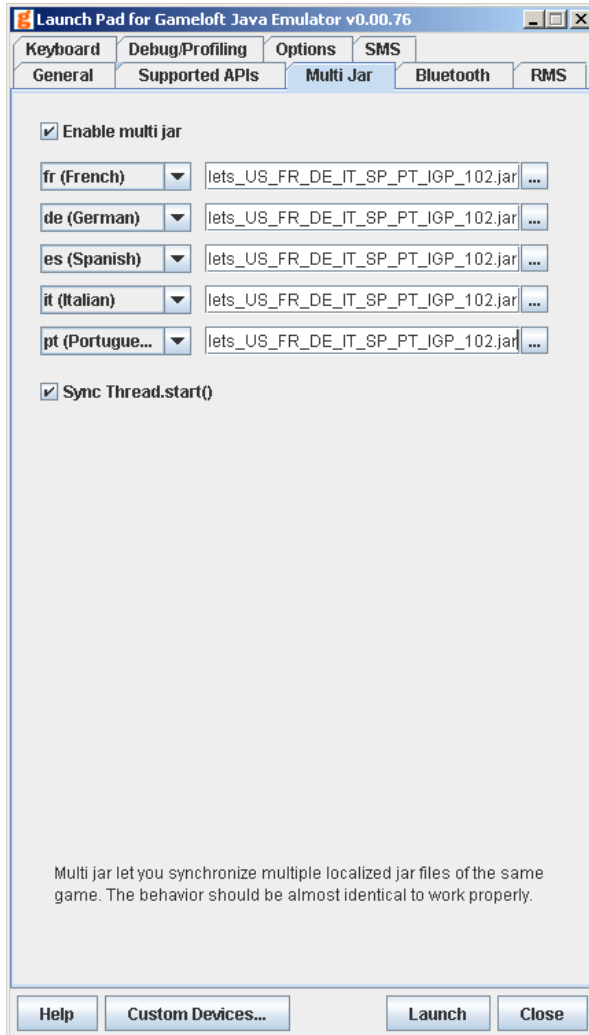
Multi Jar lets you run multiple instances of a game to test localization. The emulator will spawn client emulator instances that will be in sync with the master (each client JVM is connected to master via sockets).



All the inputs and events will come from the master and sent to clients. To keep everything in sync, even calls to *System.currentTimeMillis()* and *Random.setSeed(...)* are

“controlled” by the master.

From the “Multi Jar” tab of the launch pad, select “Enable multi jar”. Then select a jar for each client and its language. All the jars can be identical to the master jar.



The system property “microedition.locale” will be set properly for each client. If the midlet does not check for this property, it’s possible to temporarily “break” the sync with the master to select the language in each client emulator. From the main window of the master, simply uncheck the menu “MIDlet->Multijar Sync”, select the language in each client, and then check back the “Multijar Sync” option to be sure no more input is sent by accident.

The option “Sync Thread.start()” (enabled by default) is there to fix an asynchronous problem when the game is calling *Thread.start()* on a new thread before the call to *MIDlet.startApp()*, i.e.: in the canvas constructor. When enabled, the new running thread will simply wait for *MIDlet.startApp()* to terminate before running normally. If this option is not enabled, synchronization problems may occur.

3.5 Other options

Launch Pad Option	Description
Monitor all images (included released images)	Emulator will keep references of all images created, even if the MIDlet released them (displayed in Memory View). It can tell which images were released.
Monitor method calls (invocation count)	Some bytecode is inserted to enable method profiling to record method invocation count and timing.
Omit calls to <code>hideNotify()/showNotify()</code> handlers for interrupts	Select this option to simulate the behaviour of some phones which do not call <i>hideNotify()/showNotify()</i> when an interrupt occurred.
Omit calls to paint after <code>showNotify</code>	Select this option to not call <i>Canvas.paint(...)</i> when calling <i>Display.setCurrent(...)</i> . Some MIDlets do not handle well when <i>paint</i> method is called before their initialization finish.
Queue repaint calls	In glemulator, calling <i>Canvas.repaint()</i> will immediately call <i>Canvas.paint(...)</i> method. Select this option and repaint calls will be queued and processed by a separate thread. This behaviour can prevent MIDlet to cause stack overflow when calling <i>repaint</i> inside <i>paint</i> . Enabling this option will cause synchronization problems in multijar, due to asynchronous behaviour.

4. Using the emulator

4.1 Keypad

The following table shows the keyboard key mapping:

Keyboard Key	Phone Key
F1 (*)	Left Soft Key
F2 (*)	Right Soft Key
Return (*)	DPad Click
Arrow Up	DPad Up
Arrow Down	DPad Down
Arrow Left	DPad Left
Arrow Right	DPad Right
Num Pad / (*)	Phone Key #

Num Pad * (*)	Phone Key *
Num Pad 7	Phone Key 1
Num Pad 8	Phone Key 2
Num Pad 9	Phone Key 3
Num Pad 4	Phone Key 4
Num Pad 5	Phone Key 5
Num Pad 6	Phone Key 6
Num Pad 1	Phone Key 7
Num Pad 2	Phone Key 8
Num Pad 3	Phone Key 9

(*) This key can be modified in the Launch Pad

4.2 Watches

This window displays all the variables for a given context. By default, the Canvas is selected. The drop down menu contains all the available class of the MIDlet.

By clicking with right button, the selected variable instance will be displayed in a new window. It's then possible to explore all the instances in memory, including objects and arrays.

To modify a value, double click on the current value, input text for the new value and press Return.

Variable	Value
iG3D	javax.microedition.m3g.Graphics3D@1aef798
iCamera	javax.microedition.m3g.Camera@13dd8
iLight	javax.microedition.m3g.Light@e7eec9
iAngle	0.0
iTransform	javax.microedition.m3g.Transform@645fd
iBackground	javax.microedition.m3g.Background@260d8d
iVb	javax.microedition.m3g.VertexBuffer@43ad4a
iIb	javax.microedition.m3g.TriangleStripArray@a...
iAppearance	javax.microedition.m3g.Appearance@8bf3ec
iMaterial	javax.microedition.m3g.Material@3b3757
iImage	javax.microedition.lcdui.Image@1f2e95f
iSprite	javax.microedition.m3g.Sprite3D@b78915
px	0x0
py	0x0
myStrings	length=100 {Test 0 Test 1 Test 2 Test 3 Test ...
myBytes	length=110 {0x0 0x1 0x2 0x3 0x4 0x5 0x6 0...
myShorts	length=100 {0x0 0x1 0x2 0x3 0x4 0x5 0x6 0...
myObjectArray2d	length=3 {[Ljava.lang.String;@1c9ce70 [B@...
myBool	false
myBools	length=20 {false false false false false f...
testsete	length=20 {[B@1b00766 [B@1b5eba4 [B@81...
myColor	0x0
roots	length=1 {javax.microedition.m3g.World@19...
trPot	javax.microedition.m3g.Transform@fae78f
group	null
pot	javax.microedition.m3g.World@1979eb

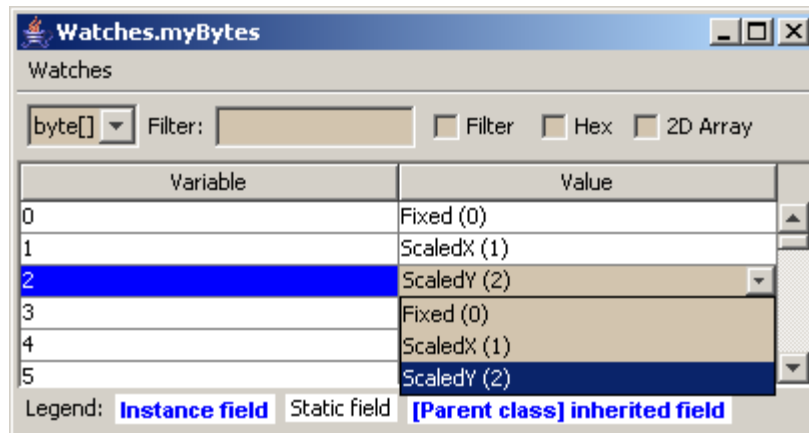
Legend: **Instance field** Static field **[Parent class] inherited field**

It's also possible to display only specific variables with a filter. Simply type text in the text box and the fields which the name or value containing it will be highlighted in yellow. Press Return repetitively to cycle through all highlighted fields.

Selecting the Filter check box will display only the matching fields.

Selecting the Hex check box will display primitive numeric values in hexadecimal.

4.2.1 Custom editors



Custom editors can be used to modify watch values. Supported editors are:

- Drop-down list such as “true” and “false” for booleans, or any other list items
- Slider to fine-tune integer values
- Color picker

Currently, custom editors should be specified in code with special command:

```
EMU://watcheditor:<data type><editor description>
```

```
System.getProperty("EMU://watcheditor:,0|Fixed;1|ScaledX;2|ScaledY");  
System.getProperty("EMU://watcheditor:boolean,true;false");  
System.getProperty("EMU://watcheditor:byte,slider[-128;127]");  
System.getProperty("EMU://watcheditor:MyCanvas.iAngle,slider[0;360]");  
System.getProperty("EMU://watcheditor:MyCanvas.myColor,color");
```

Data type can be any field such as *MyCanvas.iAngle*, or data type such as *byte*.

Editor description is the type of editor to be used, such as a drop-down list, each item being separated with a “;”, an optionally a display text after value: value|text.

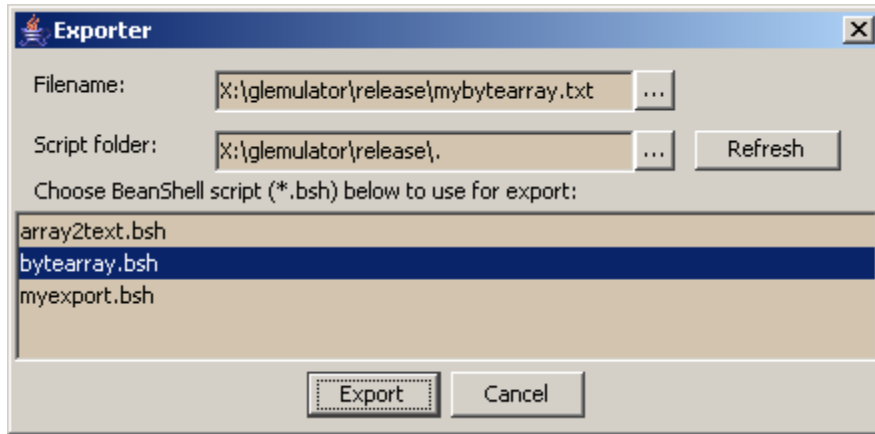
This process will eventually be automatised by using an external script file.

4.2.2 Displaying 2D arrays

Selecting *2D Array* on an array of objects will display the array as a grid. For example, this applies to *Object[][]*, *String[][]*, *byte[][]*, *short[][]*, *int[][]*, etc.

4.2.3 Exporting data

Data from watches can now be exported using a custom exporter, supporting many formats, such as file header, source code, xml, etc. Click *Watches->Export*:



Prototype of method that will be invoked:

```
String export(Object object, String filename);
```

param *object* - is the selected object instance in watches.

param *filename* - is the filename selected in dialog.

return – message to display after export.

Here are the source of example exporters:

bytearray.bsh (export a byte array to a text file, each element followed with “;”)

```
import java.io.*;

String export(Object object, String filename)
{
    if(object instanceof byte[])
    {
        out = new PrintStream(new FileOutputStream(filename));

        byte[] array = (byte[])object;
        for(int i = 0; i < array.length; i++)
            out.print("'" + array[i] + " ");
        out.println();
        out.close();
        return "Successful";
    }

    return "object not byte[]";
}
```

array2text.bsh (export any array to a text file, each element followed with “.”)

```
import java.io.*;
import java.lang.reflect.*;

String export(Object object, String filename)
{
    if(object.getClass().isArray())
    {
        out = new PrintStream(new FileOutputStream(filename));

        int length = Array.getLength(object);
        for(int i = 0; i < length; i++)
            out.print("'" + Array.get(object, i) + ".");
        out.println();

        out.close();
    }
}
```

```

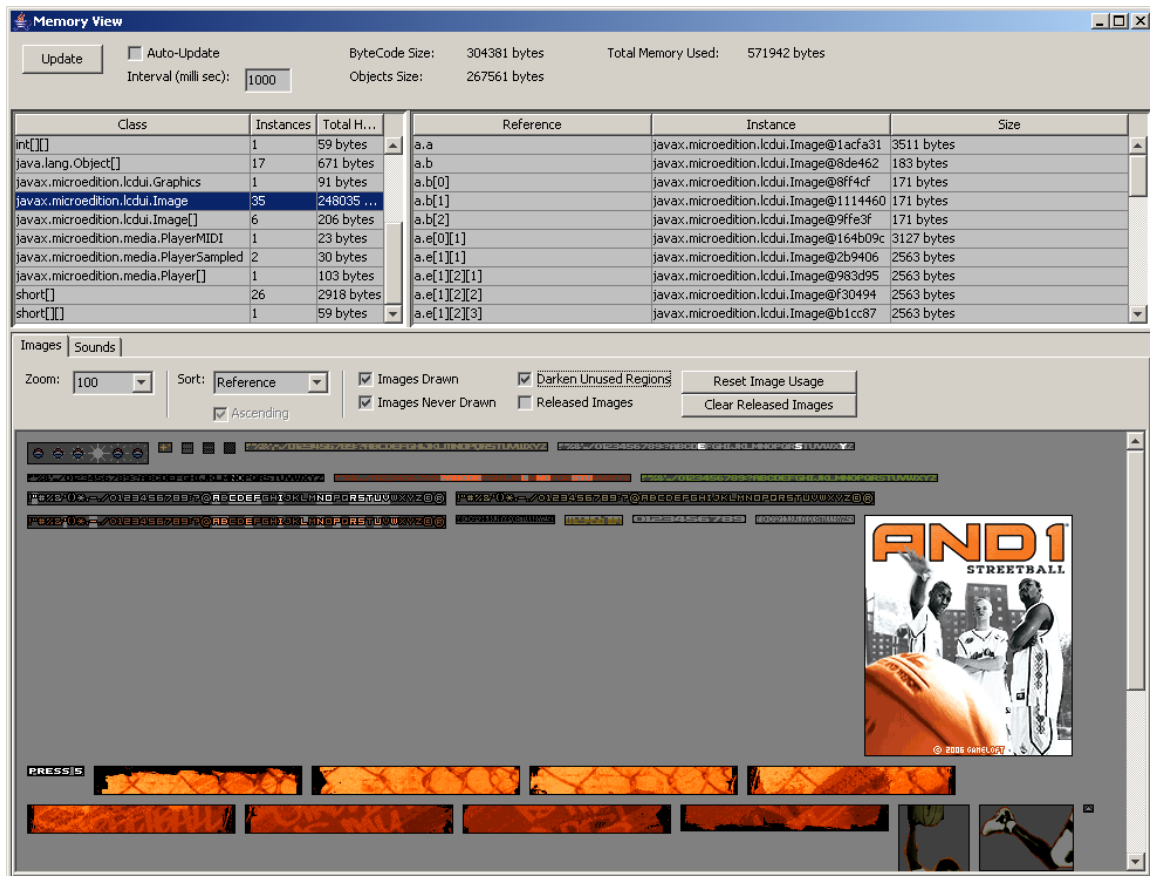
        return "Successful";
    }

    return "object not an array";
}

```

4.3 Memory View

The memory view displays useful information about objects loaded in memory. Pressing the Update button will make a Snapshot of memory. It can freeze the game for a short time since it scans recursively all the objects. It's possible to update automatically the memory view by selecting "Auto-Update" and the interval.



The window has 4 sections: Memory (top), Classes (left), Instances (right) and Images/Sounds/Aurora Sprites/Scene Graph (bottom).

4.3.1 Memory

ByteCode Size is the sum of all the .class files in the jar.
 Object Size is the sum of all the instances in memory.
 Total Memory is the sum of both.

4.3.2 Classes

This table contains all the types of classes referenced by the MIDlet, the number of instances currently in memory and the memory used for each type.

4.3.3 Instances

This table contains all the instances of the selected class type (on the left part). The *Reference* column is the “shortest” path to the instance, but there could be multiple references to the same instance. The *Instance* column is the string representation of the instance (.toString()). The *Size* column is the memory used by the instance, including a 15 bytes overhead. For a string, each character counts for 2 bytes, and for images, 2 bytes per pixels are assumed.

4.3.4 Images/Sounds

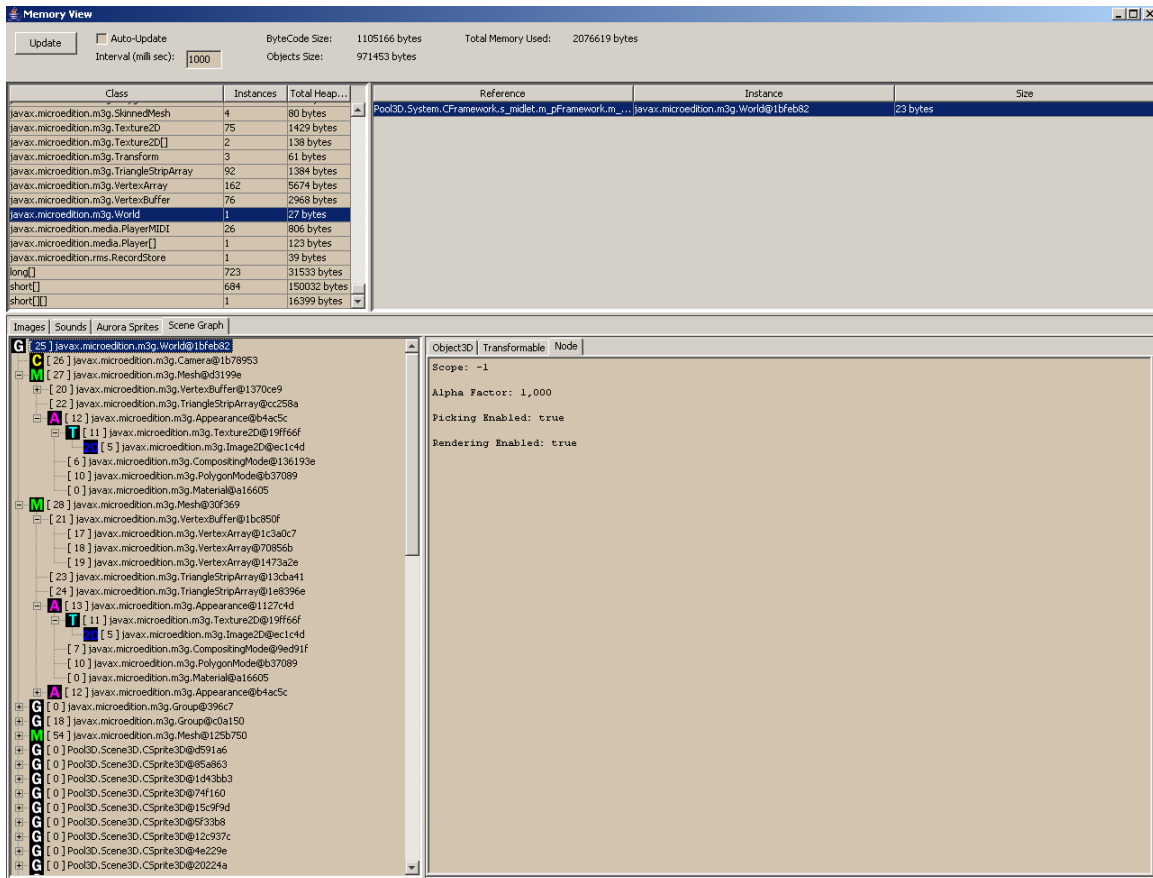
The bottom part displays all loaded images in memory. Select “Images Drawn” or “Images Never Drawn” to only display image that were used or not used. Select “Darken Unused Regions” to show parts of images that were excluded by clipping area. The image usages (draw count and unused regions) can be reset by clicking “Reset Image Usage”. Also, the images can be sorted by Reference, Creation Time, Size and Draw Count.

It’s also possible to display the images that have been released by the MIDlet and that would be garbage-collected. To enable this option, select “Monitor all images” from the Launch Pad (monitoring tab). Note that images can be loaded/released many times by the game and each instance will be displayed. The button “Clear Released Images” can be used before a major game transition, for example, to view all the released images from a specific point.

The border of images is black for normal images, red for released images and green for selected instance. The symbol “M” will be displayed above any mutable image.

4.3.5 Scene Graph

When running a 3D game using M3G API, it’s possible to view the scene graph from any Object3D instance as root. Simply select the class/instance on the top, and select the “Scene Graph” tab. Then, you can select any object from the tree, and its properties will be displayed on the low right part, each tab is for class and super classes. Currently, it’s only possible to view the properties. You can also right click on the instance to open a Watch window on this instance, but since this API is mainly implemented natively, class members are not available.



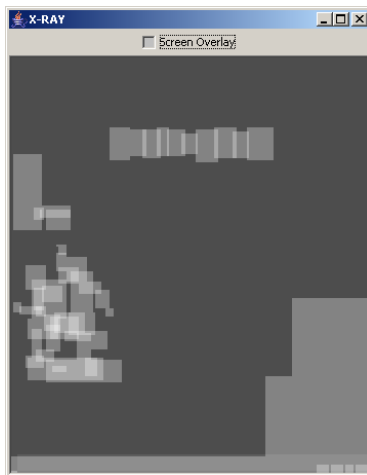
4.4 X-RAY View

Press X to display X-RAY View.

The X-RAY View displays information about what is being drawn each frame. Each drawing operation adds a semitransparent layer, which are accumulated. Black means nothing is drawn. The lighter colors means there was many drawing operations. Red color is used for drawRGB, since it is very slow on phones.

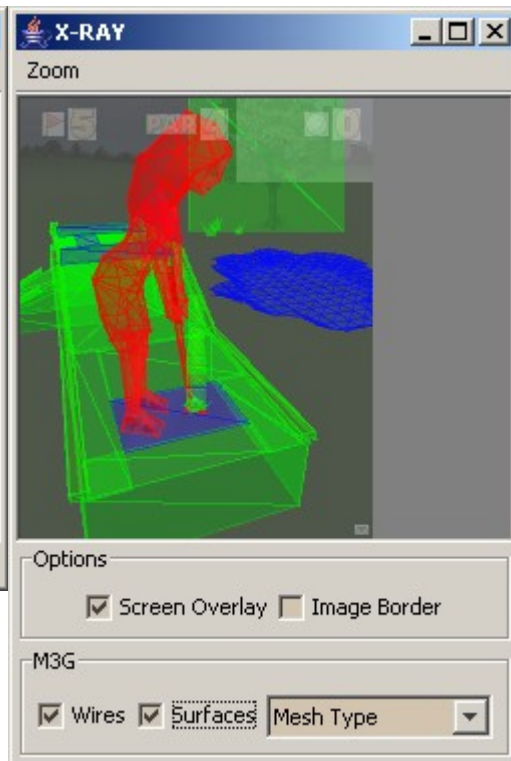
4.4.1 Using X-RAY with 2D

Here is the same screen with normal and X-RAY views:



4.4.2 Using X-RAY with 3D

To enable X-Ray with 3D select *Wires* to view polygon borders, or *Surfaces* to fill polygons.

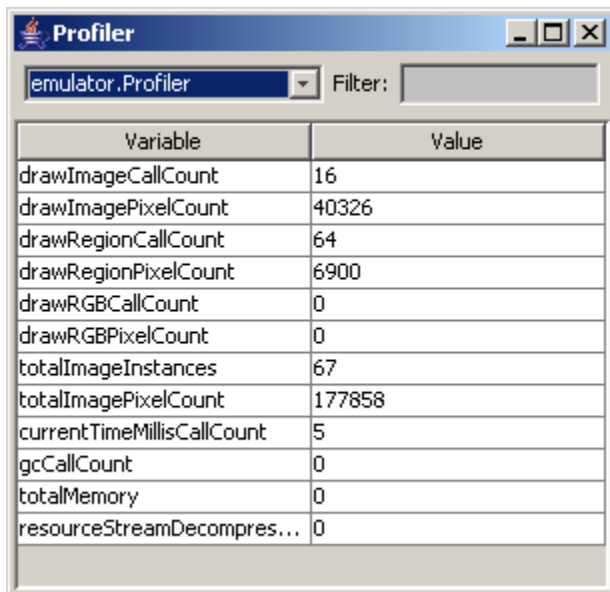


The combo box allows selecting type of view:

View	Type	Color
Mesh Type	Mesh	GREEN
	Vertex Buffer	BLUE
	Skinned Mesh	RED
	Morphing Mesh	YELLOW
	Selected Mesh instance in memory view	WHITE
Compositing Mode	ALPHA	RED
	REPLACE	GREEN
	Others (MODULATE, MODULATE_X2, ALPHA_ADD)	BLUE
Polygon Shading	SMOOTH	RED
	FLAT	GREEN
Material Type	Material	RED
	No material	GREEN

4.5 Profiler

Press “P” to display the profiler.



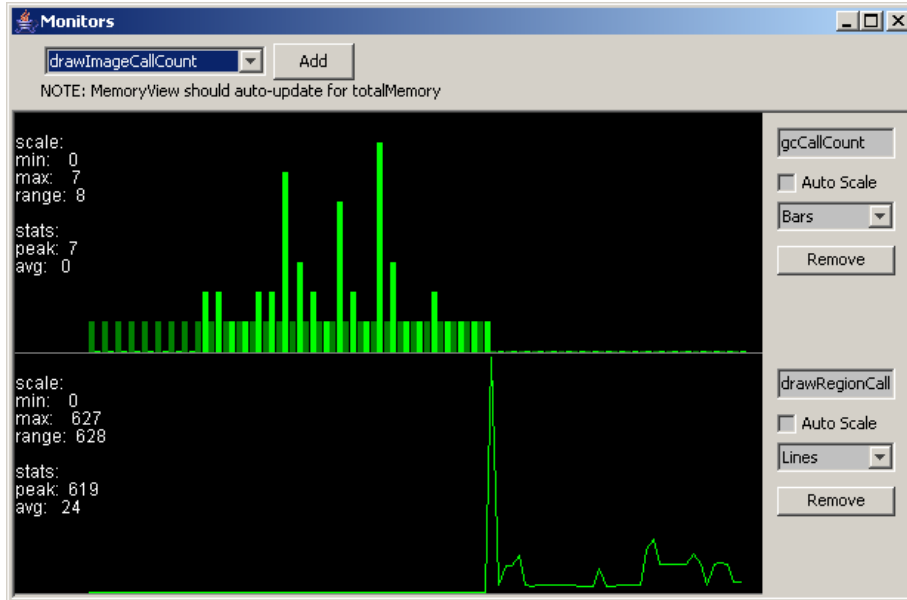
The screenshot shows a window titled "Profiler" with a dropdown menu set to "emulator.Profiler" and a "Filter:" input field. Below is a table with two columns: "Variable" and "Value".

Variable	Value
drawImageCallCount	16
drawImagePixelCount	40326
drawRegionCallCount	64
drawRegionPixelCount	6900
drawRGBCallCount	0
drawRGBPixelCount	0
totalImageInstances	67
totalImagePixelCount	177858
currentTimeMillisCallCount	5
gcCallCount	0
totalMemory	0
resourceStreamDecompres...	0

This window displays some useful information about the current frame. The number of image instances and the total surface in pixels is also displayed (this information is useless when monitoring released images).

4.6 Monitors

Monitors window let you display any of the Profiler values to see how they change with time. It's useful to view where and how often the gc is called, how many bytes were decompressed from resource streams, display performance, etc.



4.7 Methods

This window displays information about all methods inside the MIDlet classes. Here is a description for each column:

- **Class:** name of the class to which the method belongs;
- **Name:** name of the method, where *<init>* is a class constructor and *<clinit>* is the static class initialization;
- **Description:** method description such as return type and parameters;
- **Code Size:** actual byte code size of the method;
- **# References:** number of references in the code where this method is called (some public/protected methods are not referenced by the code but they will be invoked by the system, such as `startApp()`);
- **Calls:** run-time number of invocations of this method;
- **Total Time (ms):** total number of milliseconds spent in this method;
- **Average Time (ms):** total time divided by number of calls;
- **% Time:** total time divided by maximum total time.

Methods

Methods

Method Details

☐ Trace Methods ☐ Trace Threads

Class	Name	Description	Code Size	# References	Calls	Total Time (ms)	Average Time (ms)	% Time
MyGameCanvas	UpdateStateMachine	void UpdateStateMachine...	403	16	3004	4522,83	1,51	100,00
MyGameCanvas	paint	void paint(javax.microedi...	161	0	1000	4418,81	4,42	97,70
MyGameCanvas	StateMainMenu	void StateMainMenu(int)	1243	1	2969	4192,09	1,41	92,69
ASprite	PaintModule	void PaintModule(javax....	262	2	7506	612,24	0,08	13,54
MyGameCanvas	StateCanvasIof	void StateCanvasIof(int)	771	1	28	272,15	9,72	6,02
ASprite	PaintModule	void PaintModule(javax.m...	491	6	7430	439,56	0,06	9,72
ASprite	PaintFrame	void PaintFrame(javax.mi...	46	67	96	191,08	1,99	4,22
MyGameCanvas	Clear	void Clear(javax.microedi...	18	10	998	1116,73	1,12	24,69
AObject	PaintTextAnim	void PaintTextAnim(javax...	80	10	989	555,07	0,56	12,27
ASprite	DrawString	void DrawString(javax.mi...	578	83	989	526,96	0,53	11,65
AObject	Paint	void Paint(javax.microedi...	17	34	10	42,88	4,29	0,95
AObject	PaintSprite	void PaintSprite(javax.mi...	112	1	10	42,71	4,27	0,94
ASprite	PaintAFrame	void PaintAFrame(javax....	166	7	10	39,48	3,95	0,87
MyGameCanvas	InitGame	void InitGame()	474	1	1	31,93	31,93	0,71
MyGameCanvas	CreateSprite	ASprite CreateSprite(int)	19	40	9	29,80	3,31	0,66
ASprite	BuildCacheImages	void BuildCacheImages(in...	339	30	7	27,77	3,97	0,61
MyGameCanvas	EnsureLoaded	void EnsureLoaded(int)	460	25	7	26,44	3,78	0,58
MyGameCanvas	resLoadData	byte[] resLoadData(int)	470	11	19	25,56	1,35	0,57
MyGameCanvas	LoadRecord	byte[] LoadRecord(java.l...	94	3	2	18,30	9,15	0,40
MyGameCanvas	EvalExpression	int EvalExpression(java.l...	1145	5	15	13,43	0,90	0,30
MyGameCanvas	LoadOptions	boolean LoadOptions()	100	1	1	13,29	13,29	0,29
MyGameCanvas	ParseMessage	java.lang.String ParseMe...	1162	21	7	10,31	1,47	0,23
MyGameCanvas	resGetString	java.lang.String resGetSt...	309	217	38	9,90	0,26	0,22
MyGameCanvas	SwitchState	void SwitchState(int)	12	6	2	8,75	4,37	0,19
MyGameCanvas	StateAslSound	void StateAslSound(int)	85	1	4	8,50	2,12	0,19
MyGameCanvas	PushState	void PushState(int)	37	36	2	8,33	4,16	0,18
AObject	BuildCacheImages	void BuildCacheImages(in...	339	4	non	36,22	0,04	0,01

☐ Show Line Numbers ☐ Show Frames

```
name      : Paint
signature : null
access    :
desc      : (Ljava/microedition/lcdui/Graphics;)V
maxStack  : 2
maxLocals : 2

ByteCode:
L0
  ALOAD 0
  GETFIELD AObject.m_flags : I
  LDC 8388608
  IAND
  IFEQ L1
  RETURN
L1
  ALOAD 0
  ALOAD 1
  INVOKESPECIAL AObject.PaintSprite (Ljava/microedition/lcdui/Graphics;)V
  RETURN
L3
  LOCALVARIABLE this LAObject; L0 L3 0
  LOCALVARIABLE g Ljava/microedition/lcdui/Graphics; L0 L3 1
  MAXSTACK = 2
  MAXLOCALS = 2
```

To activate calls and timing information, select the option “Monitor method calls” from the Launch Pad, or with the following command line parameter:

-monitormethodcalls

The “Reset Calls” button can be used to reset the calls and timing to 0, to view the number of calls in a single frame. All the information can be sorted by clicking on a column header. For example, sort to view the methods referenced only once that could be eliminated.

When the “Trace Methods” check box is selected, each method call is printed in the console, useful for debugging. The “Trace Thread” check box will display the thread that called the method.

When a method is selected, some of its details will be displayed in the bottom part:

- **Name:** name of method;
- **Signature:** signature of method;
- **Access:** access of method (final, static, public, protected, private);
- **Desc:** description of method (internal name);

- **Max stack:** maximum stack size of method;
- **Max locals:** maximum number of local variables;
- **Exceptions:** list of exceptions declared to be throwed;
- **Byte code:** byte code instructions, line numbers, local variables, etc.

If “Show Line Numbers” is selected, the LINENUMBER instructions will be displayed.
If “Show Frames” is selected, the local variable frames (scoping) will be displayed.

Consider the following example of byte code:

```
L0
  LINENUMBER 388 L0
  ALOAD 0
  GETFIELD AObject.m_flags : I
  LDC 8388608
  IAND
  IFEQ L1
  RETURN
L1
  LINENUMBER 395 L1
  ALOAD 0
  ALOAD 1
  INVOKESPECIAL AObject.PaintSprite (Ljava/microedition/lcd/graphics;)V
L2
  LINENUMBER 396 L2
  RETURN
L3
  LOCALVARIABLE this LAObject; L0 L3 0
  LOCALVARIABLE g Ljava/microedition/lcd/graphics; L0 L3 1
  MAXSTACK = 2
  MAXLOCALS = 2
```

L0, L1, L2 and L3 correspond to Labels. LINENUMBER is a debugging declaration which tells the corresponding line of source code. LOCALVARIABLE is a debugging declaration which gives details about each local variable:

LOCALVARIABLE <name> <class type> <start> <end> <index>

- **name:** the name of local variable
- **class type:** the type descriptor of this local variable
- **start:** the first instruction corresponding to the scope of this local variable (inclusive)
- **end:** the last instruction corresponding to the scope of this local variable (exclusive).
- **index:** the local variable's index

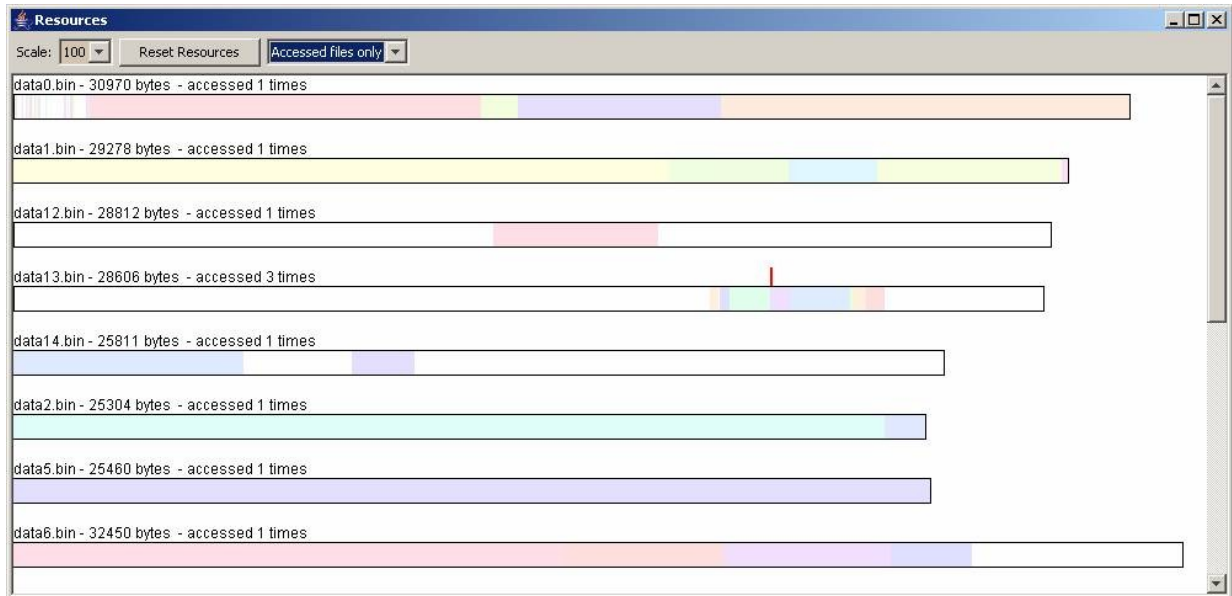
4.8 Resources View

Resources view lets you monitor jar resources activity. You can view:

- All resources
- Accessed files only (jar files accessed/opened at least once)
- Unclosed files only (jar files which have an open input stream, close() method not

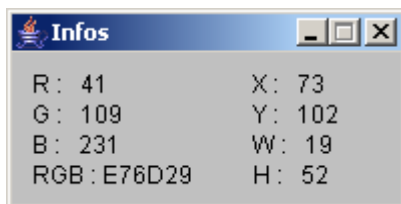
called on it)

Colors are simply to differentiate the chunks inside the jar resources. Each chunk corresponds to a buffer read from the stream and its length. The red marker is the current stream position. When a chunk has been read many times, its color becomes progressively more opaque/vivid. Pressing “Reset Ressources” will clear all the chunks and reset access times.



4.9 Display Infos

Press “I” to display some information about the mouse.



Press the mouse and drag it to make a selection box. The Infos window will display the coordinates of the mouse and the size of the box. The pixel color at mouse position is also displayed.

4.10 Display Options

Select “Options...” from the menu to display these options.

This window lets you limit the frame rate of the MIDlet, change the color depth and modify the gamma value to match the phone screen. Some work needs to be done to calibrate the gamma correction.

4.11 Capture Screen

Press “Ctrl+C” to capture a screen shot of the display, without the window frame. The image will be available in the Windows clipboard. You can then paste it directly in an email.

Press “C” to capture a screen shot directly in a file (name will auto-increment). The folder can be specified in the “Set Screen Capture Folder” menu.

4.12 Video Capture

Currently, animated GIF and AVI video formats are supported. The format can be specified in the menu “Video Format”.

For AVI video format, a dialog will be shown to select compression algorithm and parameters. To prevent this dialog to show and keep the same compression settings, you can uncheck the option “Show AVI Compression Dialog” in the “Video Format” menu. For the best quality, select “Full Frames (Uncompressed)” but the size of the video will be very big (around 50Mb for 30 sec at 15 fps with 176x220 resolution). With compression, you can choose “Microsoft Video 1” with compression quality between 75 and 100. The video frame rate is determined by the frame rate in the emulator options (menu View->Options).

Press “R” to start recording a video and “S” to stop the recording. The output folder can be specified in the “Set Video Capture Folder” menu.

4.13 Suspend/Resume

Select the “Suspend” and “Resume” menus to see how the application reacts when an interruption occurs in different contexts. On suspend, `MIDlet.pauseApp()` and `Canvas.hideNotify()` will be invoked. On resume, `MIDlet.startApp()` and `Canvas.showNotify()` will be invoked.

4.14 Pause/Step

Press the “Pause” key (“Pause/Step” menu) to pause the MIDlet and press it again to step frame per frame. Press “Ctrl+P” to resume normally.

4.15 Network availability

To test the behaviours of the MIDlet when the network is not available at a specific time, you can switch on/off the network availability with key Ctrl+N, or in the MIDlet menu.

The status on the main window will then turn red when unavailable, and green if available.

Note that turning off network availability will only affect the method *Connector.open(...)* function, meaning that once the connexion is made, it will not be interrupted.

4.16 Special Commands

It's possible to request special commands from the emulator like this:

```
System.getProperty("emu://pause");
```

The following table describes the commands and their description:

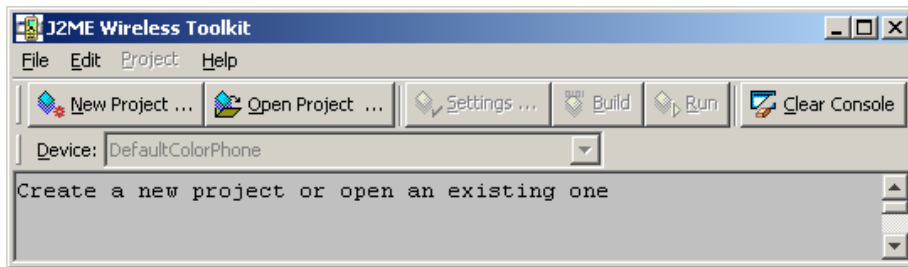
Command	Description
<code>System.getProperty("emu://pause");</code>	Request the emulator to enter pause state immediately. Note: with some pre-processors, <code>//</code> can be understood as a comment, so you may need to use this trick: <code>System.getProperty("emu:/'+'/pause")</code>

4.17 Using WMA to Send/Receive SMS

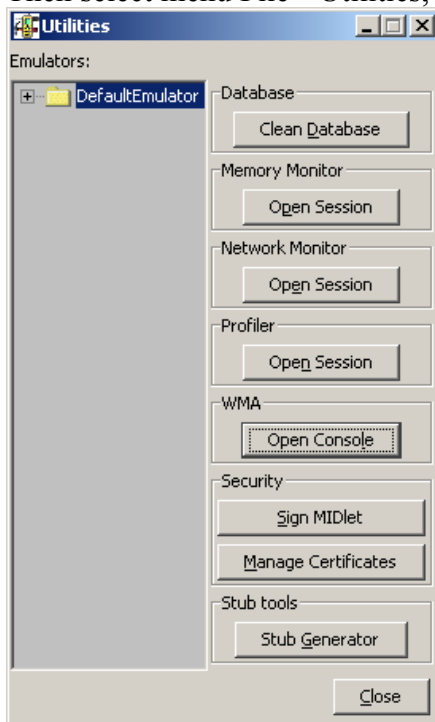
It's possible to send and receive SMS with the WMA (Wireless Messaging API). The emulator must be configured to use the OTA server from the WTK 2.2 (will not work with previous versions). In LaunchPad->SMS, check the "Enable SMS" option and browse for the root folder of WTK 2.2 (i.e.: D:\dev\j2me\WTK22). The emulator needs this path for 2 reasons: some jars (*wtklib\kvem.jar* and *wtklib\kenv.zip*) from the WTK are needed, and the system property *kvem.home* must be set to this folder. The command line will be set correctly.

Once the MIDlet is launched, SMS can be sent and received to/from other MIDlets launched, including ones from the WTK 2.2 emulator. One interesting debugging tool from the WTK 2.2 can also be used: the WMA Console.

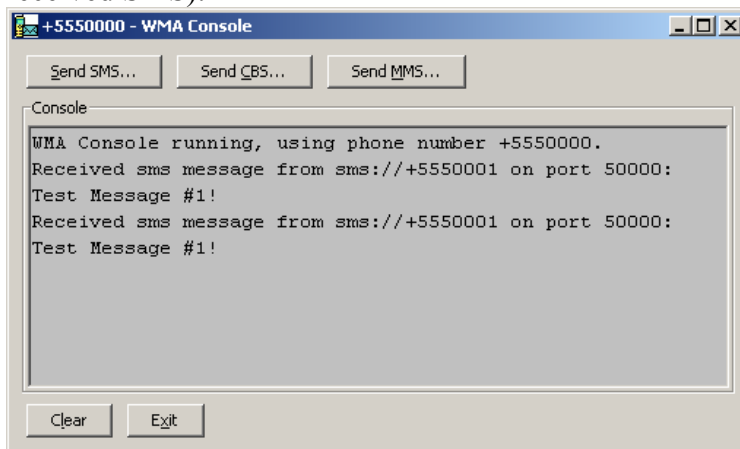
Start **ktoolbar** (*bin\ktoolbar.bat*).



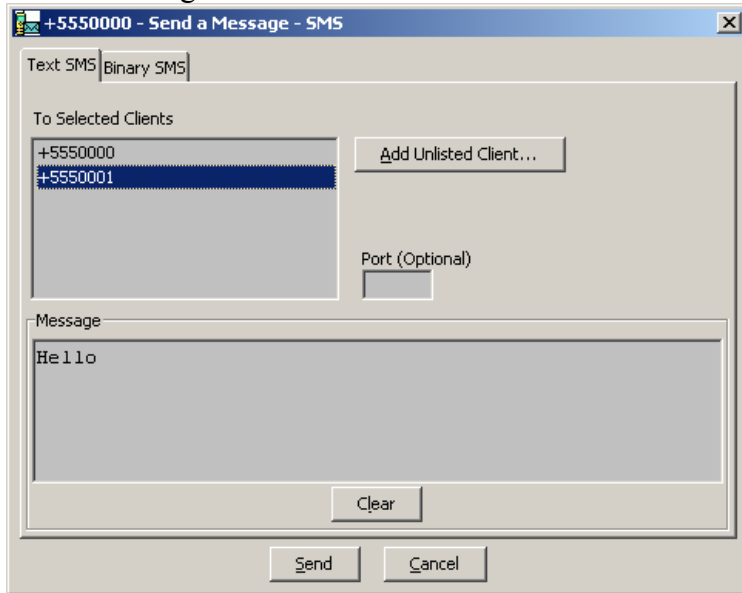
Then select menu File->Utilities, this window will appear:



Click “Open Console” below WMA, the console will appear (here it shows some received SMS):



Then click “Send SMS...” to send a SMS. This window will appear. A list with connected devices will be shown. Select the phone number (in this case +5550001), write a message and click “Send”.

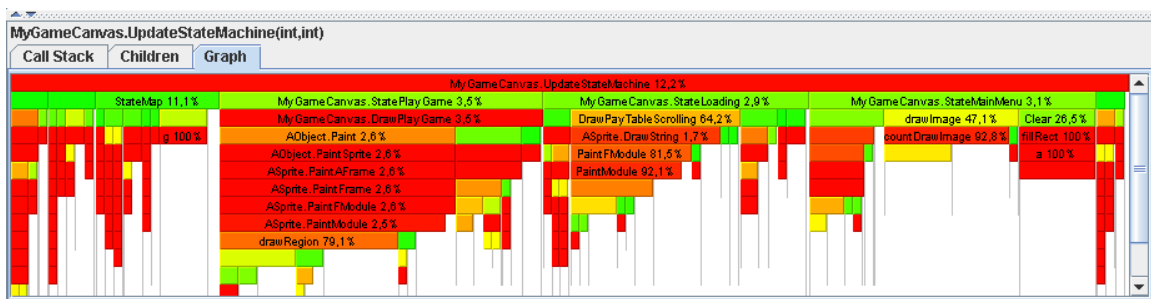
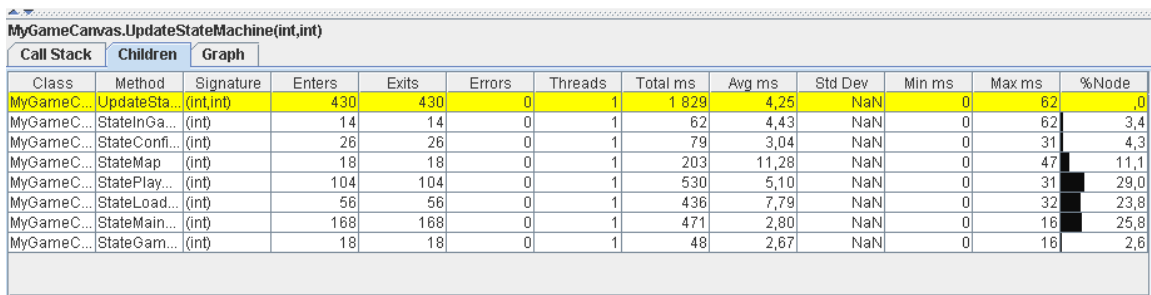
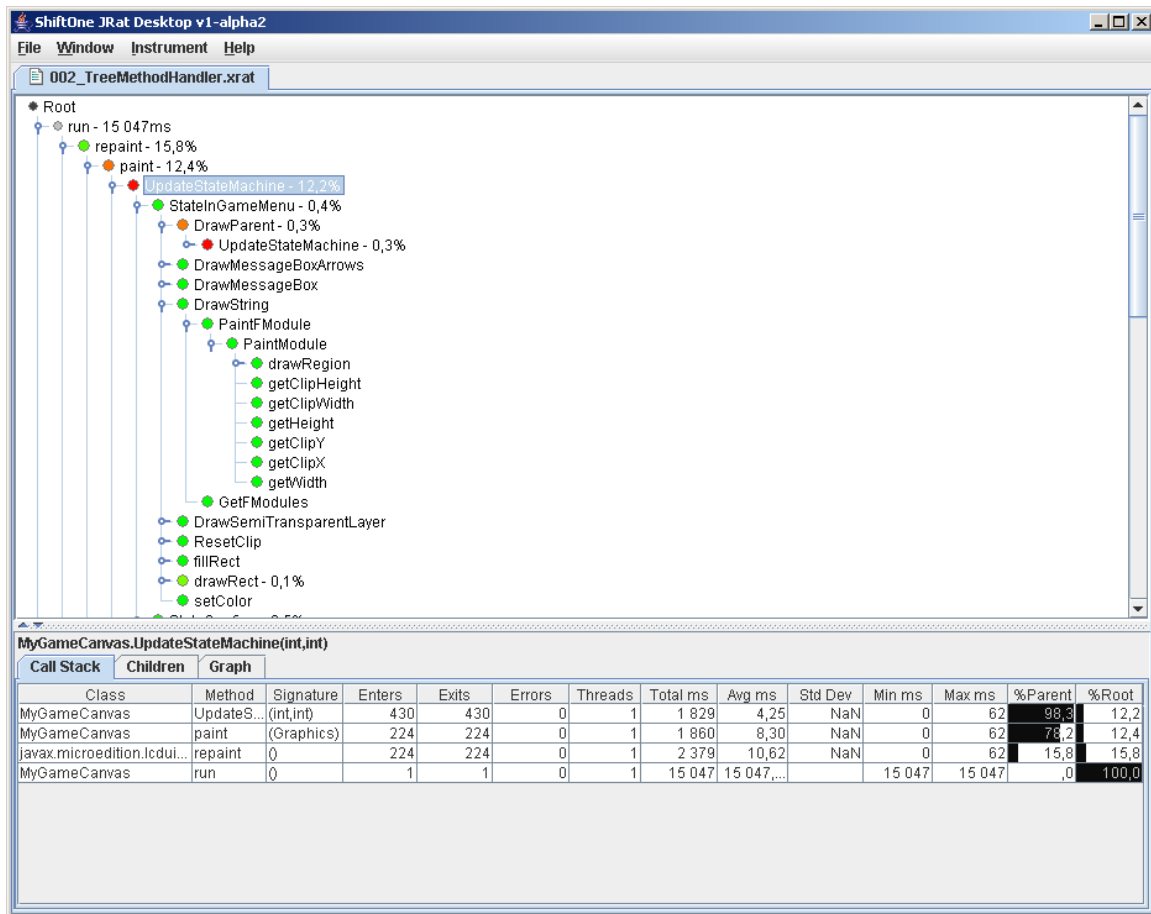


4.18 Using JRat Profiler

You can profile a midlet using the provided JRat Profiler. To enable this profiler, check the “Enable JRat Profiler” from the “Debug/Profiling” tab in the launchpad. **This profiler needs JRE 1.5+**, because it uses a JVMTI agent. It will not work with JRE 1.4.x.

The profiler records each method call and its timing, and dumps the results to a file when the emulator exits. Make sure to exit the game properly (main thread finishes normally) to have the maximum timing information. Only when the emulator exited, you can click on the “Open JRat latest report” button in the launchpad.

Here are some screen shots of the JRat viewer:



From these views, you can see where the most time is spent. You can right-click on a node for a context-menu. For more information, see the JRat web site:
<http://jrat.sourceforge.net/index.html>.

Note that some of the emulator implementations can be seen from the methods calls. These should be fixed eventually.

To enable JRat profiler from the command line, you need to add the following parameters to the JVM:

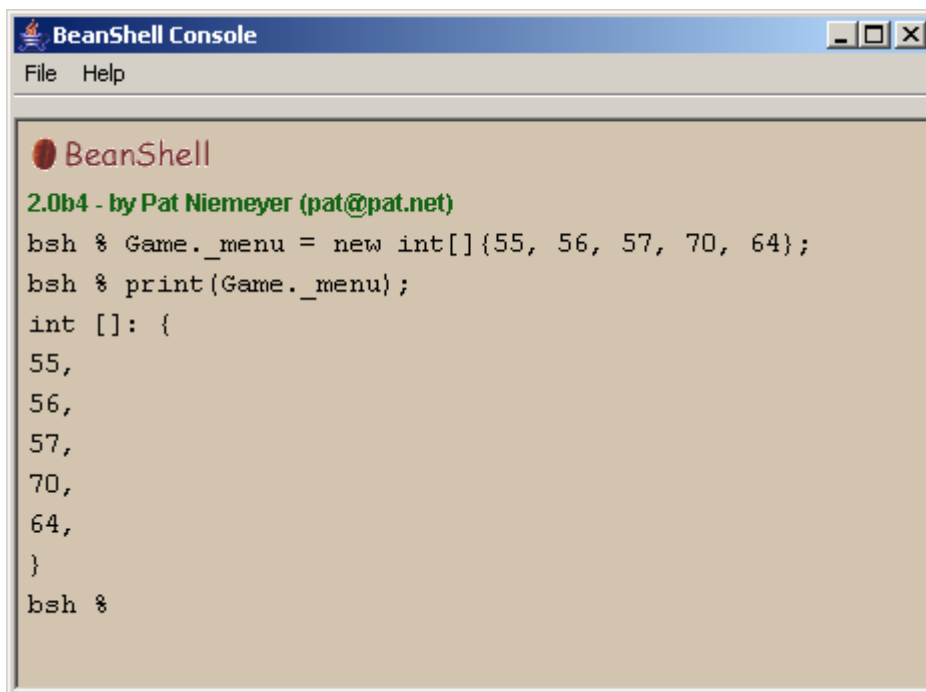
```
-cp glemulator.jar;shiftone-jrat.jar  
-javaagent:shiftone-jrat.jar=!emulator.*,video.*,com.gameloft.*,org.objectweb.asm.*  
-Djrat.factory=org.shiftone.jrat.provider.tree.TreeMethodHandlerFactory
```

4.19 BeanShell Console

The BeanShell is a very powerful tool that lets you type java statements to interact with the midlet. It can be used for debugging purposes, but should be used with care. Also, note that the code is running in a separate thread. Actually, you can:

- Invoke any method of the midlet
- Create and modify any variable/arrays
- Load external scripts for some repetitive tasks (File->Load script...)
- Execute any java statement

For example, you can type this in the console to instantiate a new array of integers and assign it to *Game._menu* static field:



For more information on BeanShell, follow this link:

<http://www.beanshell.org/manual/bshmanual.html>

4.20 Using obfuscated jars

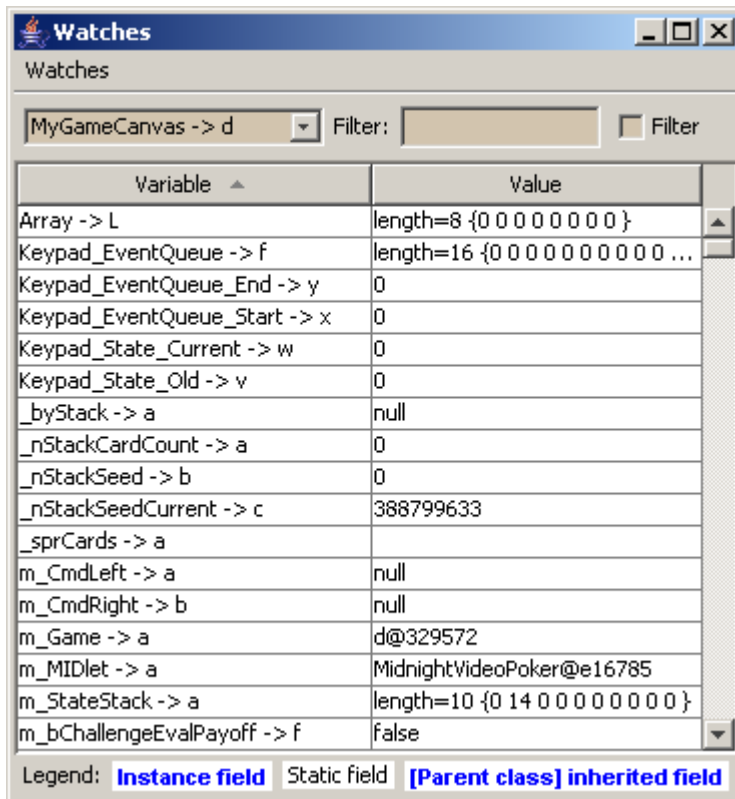
If the MIDlet jar is obfuscated with [Proguard](#), it's possible to ask Proguard to create a mapping file, using the following parameter:

-printmapping <[file name](#)>

When the emulator starts, it looks for a mapping file like <**jar name**>.map in the same folder as the jar. It's also possible to specify a custom file name in the launch pad (Debug/Profiling tab), or with the following emulator command line parameter:

-proguardmappingfile < file name>

This mapping file can be used by the emulator to “translate” the obfuscated symbols such as classes, fields and method names. The *Method*, *Memory View* and *Watches* windows will then display the translation (original name obfuscated name), making symbols easier to understand. Here is an example of *Watches*:



5. DOJA

5.1 Implementation

The emulator supports DOJA, but some packages may not be fully implemented.

Since the emulator is designed to run MIDlets, a DOJA application is controlled by a

“special” MIDlet internally. Also, the DOJA api uses the MIDP api internally for almost everything, such as `javax.microedition.lcdui.Image`, etc, which is visible in the memory view.

5.2 Raw Files

To simplify the download of `.raw` files, it’s possible to tell the emulator to get them in the jar’s folder. This option can be enabled with the option “Redirect ‘http://’ urls to files in jars’s folder”, in the “Debug/Profiling” tab of the launch pad. This has the effect of redirecting all http requests to local files, all in the same folder as the jar.

5.3 Scratch Pad

The scratch pad is located in the same folder than the `.jar` and the `.jam` files and have the same name of the `.jar`, without the extension, followed by the `id` and `.scr`. Consider the 3 following files as an example, all in the same folder:

```
Asphalt2_M341i_101_EN_Credits.jam
Asphalt2_M341i_101_EN_Credits.jar
Asphalt2_M341i_101_EN_Credits0.scr
```

Instead of downloading the resource files from a web server, an alternative is to concatenate all the `.raw` files into the scratch pad `.scr` file.

6. Using remote debugger

Debugging using a remote debugger allows to set break points in code, step by step and many more features.

6.1 Preparing the code

Java source code should be compiled using `javac -g` to enable debugging information. *Preverify* and *obfuscator* tools should not be used.

6.2 Enabling the debugging agent in emulator

The debugging agent in the emulator should be enabled, by adding some JVM parameters to the command line (or in the Options tab in launchpad):

```
-Djava.compiler=NONE -Xnoagent -Xdebug -Xrunjdp:transport=dt_shmem,address=2502,server=y,suspend=n
```

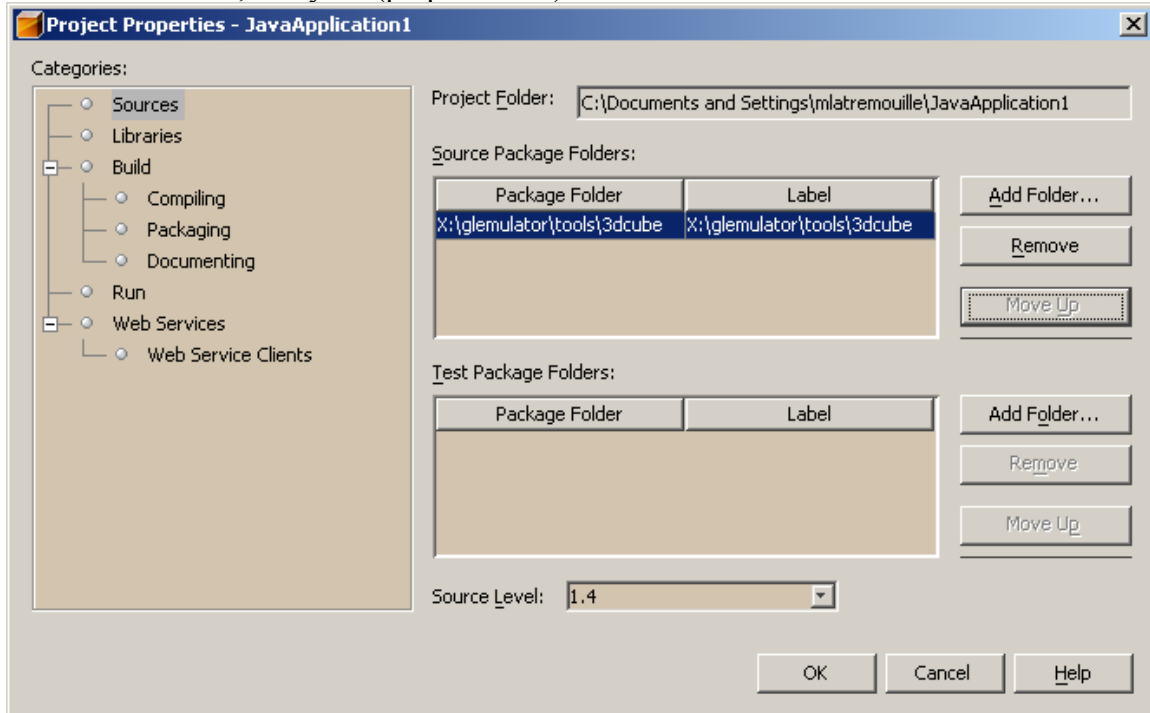
`transport=dt_shmem` means Shared Memory. If debugger does not support this type of transport, set `transport=dt_socket` instead, to use sockets.

`suspend=n` means that the application will not wait for the debugger from the beginning. This allows the debugging session to start anytime when the debugger will be attached. To start debugging from the beginning, use `suspend=y`.

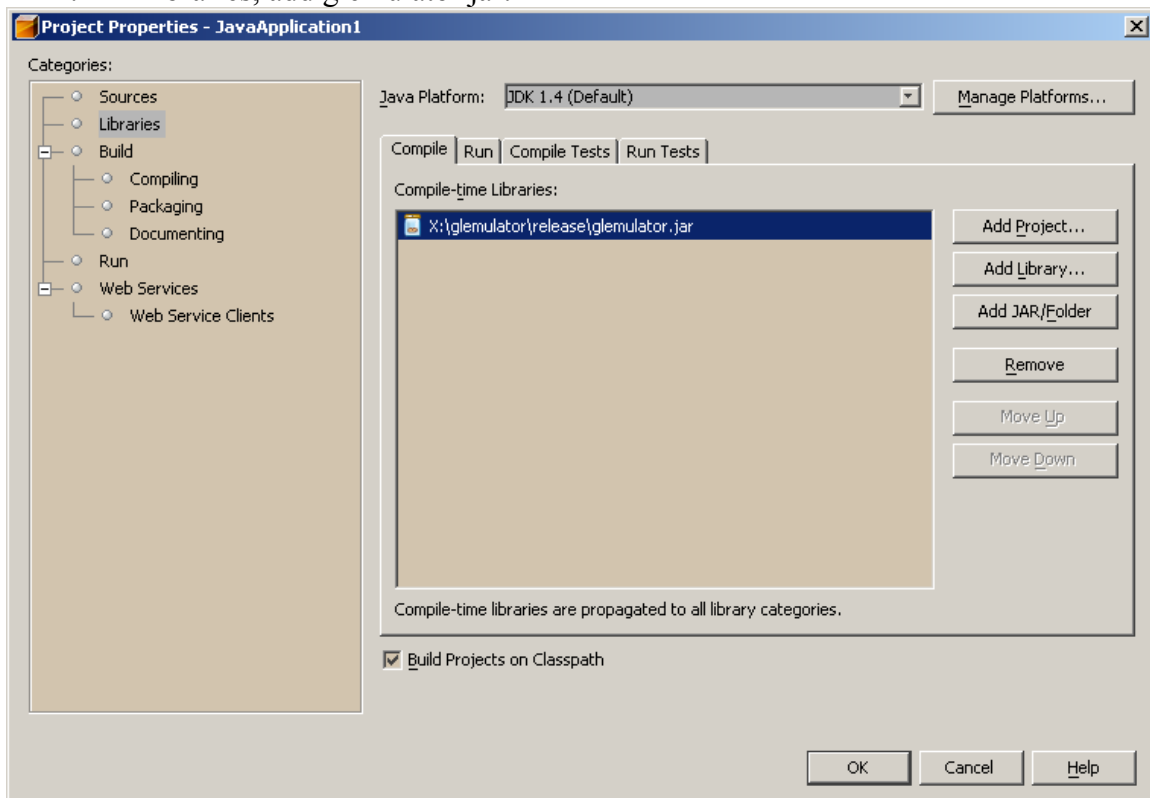
6.3 NetBeans

1. Create a new general project.

2. Click on menu File->Application1 properties.
3. In *Sources*, add your (preprocessed) source folder:

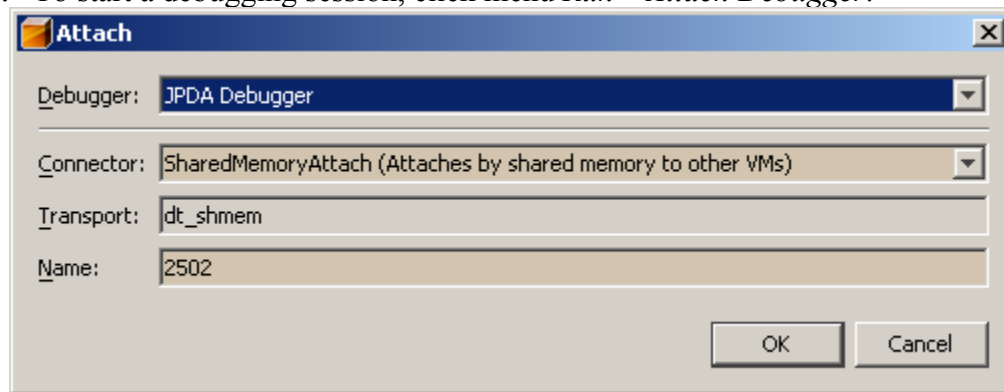


4. In Libraries, add glemulator jar:



5. Close properties dialog with OK.
6. Start the emulator using launchpad or command line.

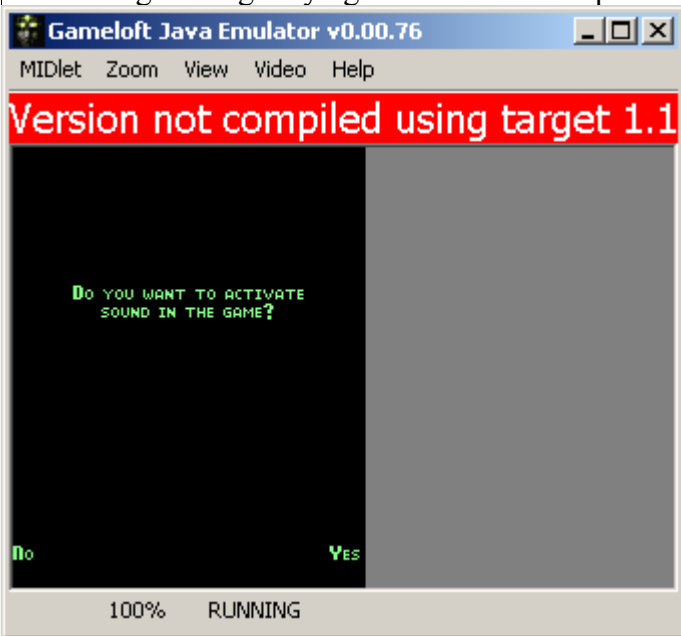
7. To start a debugging session, click menu *Run->Attach Debugger*:



8. In *Name* field, set the same *address* parameter than for the debugging agent, 2502 in this case.
9. Open source files and start debugging.

7. FAQ

Q	I have this error while starting the emulator: <code>Error occurred during initialization of VM java.lang.Error: java.lang.ClassNotFoundException: emulator.CustomClassLoader</code>
A	The parameter “-Djava.custom.class.loader=emulator.CustomClassLoader” should be removed since it’s not used anymore.

Q	I have a big message saying “Version not compiled using target 1.1” like this: 
A	The MIDlet source code should be compiled with “javac -target 1.1” to run properly on phones. The actual version is displayed in the log like this: <code>***** WARNING: class compiled with target=1.2. Should be 1.1</code>

Q	I have a security error when I try to start the emulator:
---	---

	GLEmulator security check exception.
A	The computer must be inside gameloft.org network and must be set to get DNS information automatically from a gameloft.org server. Goto Control Panel->Network Connexions->Local Area Network->TCP-IP->Obtain DNS automatically.

Q	I have a Japanese or Chinese game and I can't see the font correctly when using system font (when using method Graphics.drawString).
A	The current system font (default is Arial) does not contain the character set needed for these languages. In the Launchpad->Options, check "Specify System Font" and select an appropriate font, such as "MS Gothic".

Q	I have problems with text containing characters for other languages (French, Spanish, German, etc).
A	This can be related to file encoding mismatch with default system encoding. You can try setting the default encoding of the JVM, in text field Launchpad->Options->Custom JVM Parameters: -Dfile.encoding=Cp1252 or -Dfile.encoding=UTF8 You can see a list of java supported encodings here: http://java.sun.com/j2se/1.3/docs/guide/intl/encoding.doc.html

8. Version History

Date	Name	Description
2006-03-01	Mathieu Latrémouille	Document created
2006-06-13	Mathieu Latrémouille	Updated document for new features
2006-06-14	Mathieu Latrémouille	Updated mail for suggestions

9. Bug Report and Suggestions

You can send suggestions or report bug at:
[World-GL Emulator Suggestions](#)