Mobile General Engine features

Engine Version

The Android and iOS engine versions are in step with desktop engine version and build number. A substantial subset of the the desktop feature set is available, together with a library of mobile specific functionality.

What Does Not Work

The following features have no effect:

- Clipboard related syntax and functionality (planned for a future release).
- Printing syntax and functionality (planned for a future release).
- Setting the mouseLoc (no support on mobile devices).
- Socket syntax and functionality (planned for a future release).
- dbPostgreSQL, dbODBC and custom externals (planned for a future release).
- Industrial strength encryption and public key cryptography (planned for a future release).
- dbMysql SSL support (planned for a future release).
- Paint tools (planned for a future release).
- revBrowser (use native browser control instead).
- revFont (use ‘custom font inclusion’ mechanism instead).
- Drag-drop related syntax and functionality (no support on mobile devices).

- Backdrop related syntax and functionality (no support on mobile devices).
- Cursor related syntax and functionality (no support on mobile devices).
- revSpeak (no support on mobile devices).

What Does Work

The following things do work as expected:

- Rendering of controls with non-system themes (default is Motif theme).
- Date and time handling.
- Gradients, graphic effects and blending.
- Any non-platform, non-system dependent syntax (maths functions, string processing functions, behaviors etc.).
- revZip, revXML, dbSqlite and dbMysql.

Debugging on iOS

At present the options available for debugging applications running on iOS target devices is limited. Scripts work in a similar fashion between Desktop and Mobile so this helps.
There is a simple means of logging from an emulated target device using the LiveCode command form:
put *string*
This writes the string out to the standard error stream. These messages is visible in the Console.app when running in the simulator, and in the Console tab of the Xcode Organizer for a given target device while it is connected to the host computer.

Windowing and Stacks

The mobile engine uses a simple model for window management: only one stack can be displayed at a time.
The stack that is displayed is the most recent one that has been targeted with the **go** command. The currently active stack is the target for all mouse and keyboard input, as well as being in receipt of a**resizeStack** message should the orientation or layout of the screen change.
The **modal** command can also be used, and causes the calling handler to block until the modal'ed stack is closed as with the normal engine.
Note: Performing a further go stack from a modal'ed stack causes the new stack to layer above the modal stack and is best to be avoided.
At this time menus and other related popups are not fully implemented, as these are implemented in the engine as a specialized form of go stack they cause the current stack to be overlaid completely.
Note: The 'go in window' form of the 'go stack' command does not work correctly in the Android and iOS engines and must not be used. Since there is only one stack/window displayed at once on this platform, a generic 'go stack' should be used instead.

System Dialogs – Answer and Ask

The mobile engines support a restricted version of the answer and ask commands – both using the system-provided AlertDialogClass (Android) and UIAlertView class (iOS).
The answer command can be used in this form:
answer *message* **[with** *button* **and …] [titled** *title* **]**

This will use the iPhone standard alert popup with the given buttons and title. The last button specified will be marked as the default button.
The ask command can be used in this form:
ask [question | password] *prompt* **[with** *initialAnswer* **| with hint** *hint* **] [titled** *title* **]**
If neither question nor password is specified, question is assumed. The value entered by the user will be returned in it. If the user cancelled the dialog, the result will contain cancel.
The hint can be used to specify background text that will disappear as soon as the user enters data. The hint will never be returned.
On iOS, the text field present in the ask dialog will use the current keyboard type as set by the iphoneSetKeyboardType command.
Note: You cannot nest calls to ask/answer on iOS. If you attempt to open an ask or answer dialog while one is showing, the command will return immediately as if the dialog had been cancelled.

Non-File URL Access

The Android and iOS engines have support for fetching urls, posting to urls and downloading urls in the background. The mobile engines do not support libUrl, and as such there are some differences between url handling compared to the desktop.
The mobile engines support the following non-file URL access methods:

- GET for http, https and ftp URLs
- POST for http and https URLs
- PUT for ftp URLs

Note: When using URLs for these protocols be aware that the Android and iOS system functions used to provide them are much stricter with regards the format of URLs. They must be of the

appropriate form as specified by the RFC standards. In particular, in FTP urls, be careful to ensure you urlEncode any username and password fields appropriately (libUrl does allow characters such as '@' in the username portion and still work – mobile is not so forgiving).
To fetch the google home page you can do:
put url ("http://www.google.com") **into** tGooglePage
To post data to a website, you can use:
post tData **to url** tMyUrl
To upload a file to an FTP server you can use:
put tData **into url** "ftp://ftp.myftpserver.com"
To download a url in the background, you can use:
load url tMyUrl **with message** "myUrlDownloadFinished"
The callback message received after a load url is of the form:
myUrlDownloadFinished *url, status, data*
Here, *data* is the actual content of the url that was fetched (assuming an error didn't occur).
Progress updates on ongoing url requests are communicated via the urlProgress message. This message is periodically sent to the object whose script initiated the operation. It can have the form:
urlProgress *url*, "contacted"
urlProgress *url*, "requested"
urlProgress *url*, "loading", *bytesReceived*, [*bytesTotal*]
urlProgress *url*, "uploading", *bytesReceived*, [*bytesTotal*]
urlProgress *url*, "downloaded"
urlProgress *url*, "uploaded"
urlProgress *url*, "error", *errorMessage*
pBytesTotal is empty if the web server does not send the total data size.
You can also download a url direct to a file – this is particularly useful when downloading large files since the normal 'url' chunk downloads into memory. To do this use:
libUrlDownloadToFile *url, filename*

Unlike the libUrl command of the same name, this command blocks until the download is complete, and notifies progress through the **urlProgress** message as described above.
When using GET and POST with http(s) URLs you can use the httpHeaders global property to configure the headers to send. This works the same as the desktop engine, any specified headers overriding those of the same key that would normally be sent, and any new keys being appended.

Out-of-Bounds Group Scrolling

Two properties *unboundedHScroll* and *unboundedVScroll* enable you to configure whether scroll values for a group can be set to values outside of the actual content bounds. This makes it much easier to support the bouncing features in scrollers.

Snapshots

The iOS engine supports both the object and screen snapshot variants of the import and export snapshot commands.
To fetch a snapshot of an object use:
import snapshot from [rectangle *rect* **of]** *object*
export snapshot from [rectangle *rect* **of]** *object*
To fetch a snapshot of the screen use:
import snapshot from rectangle *rect*
export snapshot from rectangle *rect*
In the screen snapshot case, co-ords are given relative to the top-left of the screen and include the status bar.
Note: There does not seem to be a way to render the status bar without using private features of the iOS API. Therefore, if your snapshot rectangle includes part of the screen where the status bar is, it will be clipped out.