

# Trans3 with MuMu Debugger

## Overview

This modified version of Trans3 (v3.2.1) implements a basic visual debugger for RPGCode. This enables you to visualize the execution of an RPGCode program by jumping between breakpoints, or walking through it one step at a time.

This is built differently from the official release, and contains some experimental changes, which may lead to unexpected behavior. Please, **back up your project before using this build!**

**This is a work in progress!**

## Features

- Examine the active Machine Units, the internal representation of an RPGCode program
- Assign watches with variable explorer
- Watch up to 9 variables at a time
- Dump variables, canvases, and machine units to disk

## Interface / Controls

- **F1 – F4** are reserved for unfinished features.
- **F5**: *Run* until the next breakpoint
- **F6**: *Step Out* of the currently running function
- **F7**: *Step Over*, or skip past, a function call
- **F8 / Tab**: *Step Into* a function call
- **F9 / Enter**: *Run To* the highlighted line (Note: triggers only if the exact line becomes active)
- **~~F10~~ / Space**: *Toggle Breakpoint* for the highlighted line
- **F11**: Open the *Action Menu* to access less-essential operations, including:
  - Breaking into VS Debugger (for debugging Trans3)
  - Dumping all canvas contents to PNG
  - Dumping variable contents to CSV
  - Dumping machine units to CSV
  - Exported files are saved to your project's `_MUMU` sub-directory.
  - Clearing Watches / Breakpoints
  - Disabling MuMu (see `mumu.disable`)
- **F12**: *Exit MuMu* closes the MuMu display and ignores any remaining breakpoints for the current program.
- **Up / Down / Page Up / Page Down / Home / End**: Navigate code view
- **1 – 9 / 0 / ~**: Assign a watch to a slot, using the variable explorer.
  - The variable explorer searches for variables as you type. You can press tab to adopt the top suggestion, or enter any other value – the input string is re-evaluated at each step.
  - There is only rudimentary support for expressions. You can use variables within keys, but

operators & function calls are not yet allowed. So you can use something like:

```
myarray["one"][obj->foo[12]]->member->q, but not items[x+1].
```

- Type -> after an object name to search its member variables.
- You can use either single- or double-quotes when specifying literal keys.

### Variable Explorer Controls:

- **Tab** selects the first suggestion.
- **Enter** assigns the watch.
- **Delete** clears the selected slot.
- **Escape** cancels, discarding any changes.

## MuMu Directives

MuMu does not display by default. You control its operation using literal directives. These are just normal quoted strings in your program, e.g.:

```
"mumu.watch:foo";  
debugger("Not yet...");  
"mumu.pause";  
debugger("Here we are!");
```

I chose literals (as opposed to adding new functions) so as to maintain drop-in compatibility with other Trans3 versions – they should harmlessly skip over them.

There are currently five directives:

- `mumu.pause` will trigger a breakpoint on the next executable machine unit.
- `mumu.auto` will turn on automatic display, launching MuMu for every program.
- `mumu.watch:name` will programmatically define a watch in the next available slot, wrapping if none are available.
  - Substitute `name` for the variable name.
  - Use single quotes to specify literal array keys.
- `mumu.disable` will hide the debugger for the remainder of the current program, and then “detach” from future programs (until Trans3 is restarted.)
- `mumu.style` allows (some) customization of the interface.

## Known Issues

- Terminating certain statements with a newline instead of a semicolon causes line number to be off by one.
- Reference parameters do not display.
- Parameters of inherited methods do not display.
- Inline functions / `RPGCode()`
- Variable browser: Identically-named variables of different scope should display their own value.