**Language**

* Types
  + typed resources and objects (rather than indexes)
  + automatic garbage collection
  + optionally type-checked script arguments (and possibly variables)
  + actual self reference (rather than self constant and id)
  + actual null value
  + user-defined types (for e.g. data structures, lighter objects)
* Values
  + unicode support, string escape sequences, NULL-safe strings
  + enumerated types (along with constants)
  + stronger built-in types (e.g. integers, booleans, characters)
  + built-in functions stored as well as scripts (and e.g. callable with execute\_script)
* Collections
  + arrays passed to and returned from scripts
  + more-definitely sized arrays (dynamically)
  + arbitrarily-dimensioned arrays
  + ds\_\* as built-in types
  + array and data structure literals
  + array slicing
* Scope
  + methods on values (would greatly simplify naming conventions)
  + namespaces (for e.g. extension-local variables, overriding built-ins)
  + multiple inheritance/mixins (related to instance groups under api)
  + room-local variables
  + room- and game-wide events
  + better security than "secure mode" (e.g. capabilities for inherent security)
  + don't treat uninitialized variables as 0
* Scripts
  + named user-defined events/methods
  + null as default for un-passed arguments
  + argument\_number
  + argument number-checked scripts
  + argument names (e.g. for intellisense/checking)
  + pass-by-reference for all types
  + multiple return values
* Control Flow
  + lambda expressions/closures
  + short-circuiting evaluation
  + multi-level break (rather than goto, could also be replaced by closures)
* Syntax
  + tighter syntax (eliminate some delphi/pascal bits)
  + initialization with declaration (i.e. var foo = bar)
  + ?: operator or ifs in expressions
  + function/script overloading
  + pre/post increment operator

**Editor**

Some of these suggestions could be implemented using extensions or new features from the language section.

* Workflow
  + non-modal editors (labeled with resource)
  + better window management (e.g. (un)dockable resource tree, tabs)
  + manipulation of multiple resources in tree (for e.g. moving all into group)
  + restrict resource names to GML identifiers
  + streamline extensions (e.g. no split between ged/x, use extension in place)
  + drag resources, etc. between projects
  + building from the command line (for e.g. other build systems)
  + configurable default resource names
  + duplication of groups
  + portable installation (for e.g. flash drives)
  + escape to close extensions window (like global game settings)
* Debugging
  + breakpoints/stepping within code
  + separate debugger thread (don’t block with e.g. show\_message)
  + view array/data structure contents
  + better errors (e.g. debugger, stack trace, some errors don't give location)
  + resource editing at runtime (e.g. recompile code, edit rooms while playing)
  + profiling
  + copy expression value to the clipboard
  + sorting debug expressions
* Code
  + parenthesis/bracket/brace highlighting/counting/folding
  + smart indentation/code formatter
  + highlight deprecated functions
  + better intellisense (e.g. arguments near call, escape to close)
  + comment out selection
  + search all code/whole project statistics
  + better usability (e.g. ctrl+f, tab/shift+tab, home stops at indentation)
  + right click should first move cursor
  + background color
* Objects
  + bypassing D&D entirely (e.g. write GML in the actions pane)
  + display inheritance graph and inherited events
  + pausable property
  + precise movement property
  + control over instance order (possibly within objects as well)
  + control over whether events are called (like visible property)
  + show GML equivalent of D&D in tooltips
  + undo in action editor
* Rooms
  + more intuitive instance manipulation (e.g. select, copy, move selection)
  + visible instance-specific modifications (e.g. scaling, rotation)
  + zooming
  + layers
  + better tile support (e.g. autotiles, flood fill, isometric sets)
  + easier access to instance properties (e.g. selected instance pane)
  + grid offset/color
* Sprites
  + onion skinning
  + selection/manipulation of multiple frames/selected part of frame
  + curved lines
  + default precise checking and preload texture to off
* Triggers
  + Sorting/moving in trigger list (like constants)
* Files
  + split gmk's into several text-based files (faster saving, resource sharing, version control)
  + remove redundant compression
  + conditional/incremental compilation
* Editors
  + sound editor
  + particle/effect editor

**Library**

Most of these suggestions should be implemented as extensions, possibly using the new extension features. The categories they belong in could sometimes be split out of the runner into new extensions that would come with the editor.

* Extensions
  + DLL access to in-game resources and structures
  + ability to modify the IDE (inc. user-defined resources/editors)
  + extension-local resources
  + full support for calling conventions
  + better memory management (possible for GM to free strings returned to it?)
* Drawing
  + animated tiles and backgrounds
  + isometric tilesets
  + alpha in color values
  + draw\_\*\_tiled\_region/x/y/number
  + more vector graphics support (e.g. antialiasing, gradients, vector formats)
  + more primitives (e.g. quads, arcs)
  + draw\_set\_width
  + draw\_roundrect radius
  + context support (e.g. brushes, color/alpha/font stacks)
  + text formatting (e.g. inline and overall bold, italic, underline)
  + shaders (e.g. better blend mode control, pallete swapping alternatives)
  + more layer support
  + better sprite sheet support (e.g. vertical strips)
  + surface support in 3D
  + correct normals for 3D solids
  + more 3d formats
* Collisions
  + vector collisions (inc. contact points, 2/3d distance/detection/responses)
  + collision\_polygon/sprite/path
  + collision start/end events
  + return all detected collisions from collision\_\*
  + move\_contact\_object
  + continuous collision detection (possibly as object property)
  + modifiable collision masks
  + actual physics library
* Instances
  + filtering instances through lambdas/scripts/strings
  + deactivating collections of instances
  + instance\_(de)activate\_object\_region
  + instance\_(de)activate\_circle
* Sound
  + sound instances (sound\_play returns handle, callback on sound end)
  + more file formats (e.g. ogg, modules)
  + seek on sound instances
  + get sound length
  + live modification (e.g. frequency manipulation)
  + game-wide mute
* Rooms
  + arbitrary number of views/backgrounds
  + inside view event
* Network
  + deprecate mplay, replace with sockets
* Reflection
  + serialization of more than ds\_\*
  + resource lookup by name
  + dynamic loading of .gm\* (a la sprite\_add\_sprite)
  + script\_add (a la object\_event\_add)
  + D&D action to execute user-defined events
* Math
  + diff\_angle
  + vector/matrix support
  + statistics (e.g. random distributions, mode, quartiles, percentiles)
* Files
  + higher-level types (databases, xml, better ini support)
  + read entire file or specific line of file
  + game save/load events
* Input
  + mouse wheel input in code
  + joystick\_check\_\*
* Timing
  + tweening
  + arbitrary number of alarms (e.g. callbacks rather than alarm events)
  + named alarms
  + support for framerate-independence (e.g. better timer resolution)
* Interface
  + native message boxes by default
  + better control of loading screen (e.g. position, animation, fullscreen)
  + window\_get\_focused
  + minimize/maximize window
  + more clipboard control (e.g. images, custom types)

**Runner**

* drm compatibility
* move libraries into extensions for modularity
* generate bytecode at build time
* remove debugging in release build
* remove unused resources