**Lua Feasibility Research**

**Lua Academic Paper**

* *“Extension languages should have a clear and simple syntax, because their main users are not professional programmers.”* – Research into Roblox scripts should show how well this design philosophy works*.*
* *“Extension languages are not for writing large pieces of software, with hundreds of thousands lines. Therefore, mechanisms for supporting programming-in-the large, like static type checking, information hiding, and exception handling, are not essential.”* – Game mods can be quite large, so perhaps some of these features would be convenient when writing large mods (e.g. DBM).
* By design, Lua is a **general purpose** extension language, and is therefore not in any way designed for the development of game modifications (mods).
* Lua has several meta mechanisms which allow it to be so extensible:
  + **Dynamic associative arrays:** Implement a multitude of data types, including ordinary arrays, records and sets. Lever the data description power of the language by means of constructors.
  + **Reflexive facilities:** Allow for the creation of highly polymorphic parts. For instance, whilst persistence and multiple name spaces are not present in Lua, they can be implemented via reflexive facilities.
  + **Fallbacks:** Used to extend the meaning of syntactical constructions, e.g. can be used to implement different kinds of inheritance.
* *“Lua has no notion of a* ***main*** *program”* – This makes sense for game mods so the proposed new language should probably maintain this concept.
* *“Lua is provided as a library of C functions to be linked to host applications.”* – If Lua is already interpreted as a library of C functions, then it should be fairly simple to translate the proposed new language to C too.
* Lua has a **global** scope, which I may want to carry forward to the proposed new language.
* *“The unit of execution of Lua is called a* ***chunk****. A chunk may contain statements and function definitions. When a chunk is executed, first all its functions and statements are compiled, and the functions added to the global environment; then the statements are executed in sequential order.”* – Due to this execution model, Lua in a sense is Event*-*Driven by the host application. I think this model of execution likewise makes sense for mods.
* *“Lua is a dynamically typed language.”* – As a scripting language, this seems like the correct choice. For larger mods however, perhaps it does not make sense to use a dynamically typed language, so this is something that should at least be considered.
* *“Functions can return multiple values, and multiple assignments can be used to collect these values.”* –