HSVM (Fiber VM Execution Method)

By: NotALeafyWannaBe

What Is A HSVM?

A **HSVM (Hooked State Virtual Machine) [Note: This is explaining a HSVM for Fiber Execution Methods This does not apply to all HSVM's.]** is when you remake Fiber's opcodes and convert code to Fiber's format then run it on your vm (which will also use your remake of Fiber's opcodes but the opcodes have to be the same/do the same thing as Fiber or else Fiber will crash.) and pass/insert it to fiber's output.

Example: "Form.ShowDialog();" gets converted to "openformfiber" function, then the code gets executed on YOUR virtual machine, then it passes/inserts the executed script to fiber's VM, **BUT** because the script has already been executed on your state and on your VM, Fiber's VM just passes/inserts the executed script to Fiber's output which does the "opening of a form", etc.

What are the Downsides to a HSVM?

The Downsides to a **HSVM** include the following.

1. Some functions, for example "addbutton" function let's say this function just can't be reversed and your VM cant run it (this will happen for some functions and is just the nature of a HSVM for fiber execution), this means you have to just convert the code from the user/input to fiber's format and just get fiber's VM to run that code for you.

2. This is what we try to avoid the most for the reason being that we want to run as many functions and as much code as we can on our own state and our own VM, but this will always happen with a **HSVM**, and there isn't anything we can do about it.

How is the code sent to fiber?

Fiber uses a Named Pipe to do some things cross platform and/or between the bootstrapper (auto updater) and such things like that. Now in theory we should just be able to send regular code into the pipe right? **Well, no.** This is because of 2 main things.

1. The code has to be in fiber's format.

2. The pipe does not handle script execution.

So then how is this possible?
Well because of where the pipe is what what it does we are still able to send code to it to tell it that we want the script to go to the VM and this is because of Fiber's poorly coded pipe in which i from what the fiber developers have said, they need the pipe, so from there we can use a method for script execution which covers the first "1" problem, and from there that is how we achieve script execution on fiber!

Developers Note: What a HSVM will require to make and get running at its best performance?

A HSVM for fiber script execution will need **lots** of time and work to get working properly, and can be very buggy if just one thing isn't coded right.

Why can't i find a open sourced CCVM for fiber script execution yet?

Since HSVM's are really powerful and also hard to make, usually most people don't want to release their hard work for others to use while those people who use it didn't do the hard work themselves. I myself have made this method already and i can tell you first handed it was **not** easy.

My Github: [NotALeafyWannaBe's Github](#).

On my github i have released a **open sourced Wrapper for fiber script execution** and some other things as well!

How many execution methods for fiber are there?

From when this slide was made there are 3 methods of execution for fiber. Those methods are listed below.

1. **HSVM** (this method ***one of the best methods for fiber script execution***).

2. Wrapper (this method can memory leak a lot if you dont code **almost everything** correctly).

Why don't HSVM's memory leak?

HSVM's can and might memory leak if there **not** coded correctly, but when **HSVM's** are coded correctly then 99.9% - 100% of the time there is no memory leaking, some functions in fiber like for example "opening a new form" will **USE** memory **but it wont leak**, even then, that's on fiber's side and not yours because like i said 100% there will always be some functions that you have to run on fiber's on VM by just wrapping the function and having it ran on Fiber's VM.

One of the biggest reasons that a **HSVM** does not memory leak when coded correctly is because your running almost everything (besides some functions) on your state and your VM so you have control of what's being executed on your side and **HOW** to handle what's being executed on your state and your VM this gives you the opportunity to make sure that scripts that are being converted and ran on your state and your VM do not memory leak because since its running on your state and your VM you can call things like "GC.Collect();", etc.

BUT like i had said earlier not everything can be ran on your VM and your state because then you would be making your own entire game engine/application that will not be able to be used for fiber because it will be a application/game engine and not a executor, so this means some things won't be able to be controlled on your side this does also mean that you **can run into lots** some problems with custom functions, etc.