I'd be glad to expand on the **NexusScript** coding interface, focusing on features that make it robust, engaging for a multiplayer learning environment, and visually appealing within the CLI framework. This expansion adds features for debugging, team collaboration, and hardware interaction. 🛠️

5. Expanded NexusScript Coding Interface Features

The interface must be more than just a terminal; it needs to be a fully functional coding and collaboration environment that lives within the Virtual Computer (VC).

A. The In-Game Module Editor (edit Command)

The primary interface for writing code is the **Module Editor**.

Feature	Command Syntax	Description	Learning & Utility
Access/Save	edit $module_name	Opens or creates a script file. save and exit are sub-commands within the editor environment.	Teaches version control and file management.
Syntax Highlighting	(Automatic)	Within the editor, **NexusScript** keywords (func, if, set), variables ($), and comments (#) are color-coded.	Improves readability and helps spot basic syntax errors quickly.
Code Snippets	insert snippet [type]	Automatically pastes common structures (e.g., if blocks, for loops, a standard exploit template).	Reduces mobile typing fatigue and promotes correct structure usage.
Error Feedback	debug $module_name	Runs a dry-run check of the code. Catches basic syntax errors (e.g., mismatched brackets, undeclared variables) before execution.	Teaches static code analysis and debugging fundamentals.

B. Dynamic Debugging and Logging

Debugging in **Code Nexus** is a hands-on process, forcing players to trace execution flow and resource usage.

Feature	Command Syntax	Functionality	Learning Objective
Trace Mode (VIP Feature)	run $module_name --trace	Executes the script line-by-line, pausing after each line and displaying the current values of all active variables and object	Deepens understanding of execution flow and variable state.

Feature	Command Syntax	Functionality	Learning Objective
		properties.	
VC Resource Monitor	vc status	Displays real-time metrics on the Virtual Computer: CPU load, RAM usage by active modules, and network bandwidth saturation.	Teaches **Performance and Resource Management**. A poorly-coded loop can spike CPU, leading to module crash or detection.
Conditional Breakpoints	run $module --break [line_number]	Pauses script execution at a specific line. Essential for isolating complex logical errors in late-game modules.	Introduces the concept of **advanced debugging techniques** used in professional IDEs.
External Log Retrieval	target_log read [type]	Fetches simulated logs from the target system, allowing the player to see *why* an attack failed (e.g., "Connection refused on Port 22").	Correlates code failure with target security measures.

C. Multiplayer Collaboration Features

To support the multiplayer nature, the interface includes tools for sharing, selling, and collaborating on code modules.

Feature	Command Syntax	Utility	Player Interaction
Module Sharing	share $module_name @[player_ID]	Encrypts and sends a copy of a module to a teammate or collaborator.	Facilitates co-op missions and team coding.
Community Marketplace	market view module $name	Accesses the in-game marketplace where players can buy or sell their own custom-coded modules for Credits (\mathbb{C}).	Creates an internal, player-driven economy for valuable, complex scripts.
Module Trust Score	(Automatic)	A rating displayed on shared/sold modules based on community ratings, **NexusScript** efficiency score (low RAM/CPU usage), and security audit results (no malicious code).	Encourages players to write clean, efficient, and trustworthy code.

D. Advanced Hardware Interaction

The coding interface must allow for direct interaction with the player's unique, upgradable hardware.

Feature	Command Syntax	Description	Utility and Skill Check
Thread Control (High Level)	thread spawn $module_name	Allows a player to run multiple script processes in parallel (limited by CPU cores).	Teaches **Multi-threading and Concurrency**. Critical for running a simultaneous attack and defense.
Hardware Overclock	vc cpu overclock [level]	Temporarily boosts CPU/RAM performance (but increases the risk of a hardware burnout or detection by opponents).	A risk/reward mechanic that simulates pushing hardware limits.
Custom I/O	$nic.raw send $data	Allows high-level players to bypass high-level **Service_Objects** and send raw data packets.	Simulates **low-level networking** and bypassing application-layer security.