Hardware that Hums, Software that Sizzles

Customizing your *GURPS* Computer

by Michael J. Daumen

> *Aaron flipped the case over with twitching fingers, trying hard to ignore how desperate he'd become. The last of his money and favors were sunk into this box; it, and the hope for one more score, were all he had now. Carefully, he opened the static-free cover, blowing gently to remove any dust. The chip gleamed beneath the overhead lights. Wired in his deck, it was the key which could set him free -- or seal his fate.*

Computers in ***GURPS*** are defined by their Complexity rating, a catch-all number that reflects speed, capacity, size, and TL. The simplicity of Complexity is deceiving; one variable represents a wide variety in both devices and programs. But it is hard to introduce more character-computer interaction into the ***GURPS*** system without adding more statistics. With this in mind, here are some ideas for giving computer experts more to do in a campaign than hack-and-backslash.

How it all Works

The concept that makes it possible to squeeze more into a computer and out of a program is fractional Complexity. Expanding Complexity one decimal place -- 3.1, 4.9, and so on -- gives players the motivation to tinker with computers, and GMs the flexibility to introduce new hardware and software without increasing performance by factors of ten each time.

In hardware terms, an extra tenth of Complexity provides room to run one more program of Complexity-1, as per normal ***GURPS*** rules. For example, a computer of Complexity 4.1 can run two Complexity 4 and one Complexity 3 programs, or one Complexity 4 and eleven Complexity 3 programs. Characters get fractional increases by installing custom hardware into the computers they use.

In contrast to computer systems, optimized and custom programs have smaller Complexities, leaving more room in memory for other programs. Running a Complexity 4.9 program instead of a Complexity 5 program saves enough space for one additional Complexity 4 program in memory. Programmers get savings in Complexity by writing new software and modifying existing programs.

Despite the modifications made by these rules, computers and programs always operate at their original or "base" Complexity. When determining Complexity-derived effects such as skill levels (anything but the number of programs a computer can store), always round off to the nearest integer. To avoid confusion, the greatest improvement possible to either computers or programs is 0.4 -- preventing arguments about whether a Complexity of 4.5 should be treated as 4 or 5!

A Complexity increase/decrease of 0.1 is a Simple invention; a 0.2 increase is Average; 0.3 is Complex; and 0.4 Amazing (using the terms as they appear in the Gadgeteering rules in CII123). The difficulty of making these changes are represented by skill roll modifiers of 0, -2, -6 and -10 respectively.

Hardware Upgrades

Computer users planning an upgrade must first select the amount of Complexity to replace and the amount of fractional Complexity they are seeking. The former can be a number up to the computer's Complexity minus 1. For example, up to 6 points of Complexity can be modified for a Complexity 7 system. The benefit to be obtained from an upgrade can vary from 0.1 to 0.4.

Of course, before a new part is installed, it must be available. It's possible to find the desired components for sale in the right market, or to requisition them from a supplier. Rummaging through unused or forgotten hardware is also possible -- in a cyberpunk environment this skill is a fine art! Characters with Scrounging can make a roll, subtracting the Complexity and TL of the desired component. Success means a part is available; whether the owner objects to its use is another matter.

Better hardware is designed using the ***GURPS*** rules for inventions (B186). Make the concept roll against the designer's Electronics (Computers)-3 or Engineering (Electrical Work)-6. The extra difficulty represents the fact that off-the-shelf components would have the latest refinements; if a modification were obvious someone would have already done it! Instead of the -15 penalty to the concept roll, subtract the operating TL, base Complexity, and the gadgeteering penalty for the amount of the improvement. A critical success does not count this modification towards the four maximum (see below); a critical failure means that the hardware in question is so integral to the computer that its Complexity cannot be modified in subsequent attempts.

Example: Aaron's fixer knows a student who thinks he has come up with a breakthrough in microchip architecture. He thinks he can build a chip which will replace 1 point of Complexity with 1.3. His Electronics roll is a 17, and Aaron finds out that his motherboard is so strangely configured that it cannot be removed without scrapping the whole deck. Because of this, the total amount of Complexity available for future attempts is reduced by 1 to 2 (of the possible 3 for a deck of Complexity 4).

For the assembly roll, subtract TL, gadgeteering penalty, and twice the replaced Complexity. A critical success allows the replaced Complexity to be used in subsequent attempts (while retaining the fractional bonus). Critical failure means that the faulty component looks fine, and can be installed without problems -- but it may eventually cause internal damage or system failure at the worst possible time . . . .

Example: Aaron asks a chipmaker to burn a faster coprocessor for his cyberdeck. After haggling the specs and cost, they decide that the component they need is Complexity 2 (of 4 total) and will increase his deck's Complexity by 0.1. The penalty to the conception roll is -3 (starting penalty), -8 (TL), and -4 (base Complexity), or -15. The penalty on the manufacturing roll is -3, -8, and -4 (twice the replaced Complexity), also -15. There is no gadgeteering penalty for the 0.1 increase since this is a Simple improvement. The engineer makes both rolls and delivers the merchandise into Aaron's greedy fingers.

Setting up the new hardware requires two rolls: an Electronics (Computers)/Engineering (Electrical Work)-3 roll (for the physical installation) and a Computer Programming roll (to resolve any compatibility problems between the new component and existing software). Subtract the working TL, the system's base Complexity, and the gadgeteering penalty, but add the amount of the Complexity which is being replaced. Both attempts require the use of a suitable tool kit or lab. While a critical success on an assembly roll makes a task shorter (see below), critical failures on these rolls can cause physical damage or data loss.

Example: Aaron wishes to rewire his cyberdeck with the new chip. To successfully upgrade his deck, he needs to make Electronics and Computer Programming rolls with a total penalty of -8 (TL), -4 (base Complexity), +2 (the Complexity replaced) -10 for each. His modified rolls are 6

and 13 -- although he installed the new component correctly, it creates a software glitch that must be resolved in another attempt.

A computer can be modified more than once, provided that at least 2 points of the system's original Complexity remain, and that critical failures have not made available points off-limits. Since the maximum amount of increase is 0.4, a system can only be modified no more than four times. This limit includes failed attempts and attempts by more than one engineer.

Example: Aaron has tried to modify his Complexity 4 cyberdeck twice. His first attempt involving 1 point of Complexity failed critically, while the second replaced 2 points with 2.1. He has 2 points of original Complexity remaining. But since one must never be replaced (by definition), and the other cannot be replaced (because of the critical failure), there's nothing left to customize. Had his first try not ended so badly, he would still have 1 point available for one more upgrade. He's left with a deck of Complexity 4.1 that cannot be improved. Now it's time to look at his software . . . .

The cost and time to required to make upgrades can vary greatly. One rule of thumb is to have the base price equal to the replaced Complexity squared in thousands of dollars, subject to whatever other modifiers the GM thinks apply. Each action governed by a die roll takes (TL x Base Complexity x replaced Complexity x 60) minutes to resolve, divided by how much the roll succeeded -- so if Aaron succeeds by five on one roll, that step takes (8x4x2x60)/5 = 320 minutes. A critical success on the assembly rolls halves the time to perform each job.

Optimized Programs

Programmers can try to write new software with lower Complexity, or improve existing programs. The former requires more knowledge and time, since the program may delve into other fields of study; the latter may involve cryptanalysis and bypassing security routines just to examine the code! Customizing programs is cheaper than installing new hardware -- the cost is the purchase price of the program to edit, or the freelancer's fee.

New Software

Writing an optimized program from scratch uses the inventing rules at B186. A programmer can attempt to write entirely new software, or custom versions of existing programs. In addition to the concept and assembly ("compiling") rolls, the programmer (or his assistants with the relevant skills) must make success rolls for any other skills appropriate when creating a specialized program (a Personality Simulation requires the assistance of someone with Psychology skill, an Expert System requires an expert in the underlying knowledge skill, etc.).

Each roll is at -3, with the following modifiers: twice the program's base Complexity; and the penalty for the gadgeteering improvement (Simple to Amazing). As with hardware, a critical failure in the design stage means the programmer thinks the software he examined cannot be optimized; similar misfortune in the compiling stage means that he thinks the program is functional, when in reality it will eventually crash.

Example: After encountering polymorphic ice on a harrowing run, Aaron thinks he can incorporate some of its features into a Webster program (base Complexity 2), which should result in an improved Complexity of 1.7. This 0.3 improvement is Complex, carrying a -6 penalty. The GM mercifully decides that his familiarity with the Net is sufficient to not require any other skill rolls, but since he has no access to Webster software he needs to write one off the top of his head. The penalties to his rolls are -3 (the normal penalty), -4 (twice the base Complexity of a Webster program), and -6 (Complex task) or -13 total.

Version 2.0

Instead of writing new software, Characters with Computer Programming skill can try to improve existing programs. Getting to the actual code can be difficult; erstwhile pirates will first need to detect and bypass security measures. A Computer Hacking roll, minus the program's base Complexity, reveals any such safeguards. A second roll, either Computer Hacking or Cryptanalysis (if the program is encrypted) at the same penalty will circumvent the protection -- unless the GM uses more sophisticated rules for this type of activity.

The penalties for writing from scratch apply as well to optimization. Programs can only be modified once, unless someone achieved a critical success during a previous edit. In either case, the gadgeteering penalty is based on the reduction in the program's original Complexity, as opposed to its current Complexity. For example, improving a program from 4.7 to 4.6 is just as difficult as going from 5 to 4.6, since the designer probably used his best tricks in a previous rewrite.

Example: Looking to free up some disk space, Aaron prints out his deck's Success program and pores over it for extraneous bytes. Its current Complexity is 2.9, the result of a lucky all-nighter a few months back. After several pots of coffee, Aaron bets he can shave another tenth off to a new size of 2.8. The penalties to his roll are -3 (the usual penalty), -6 (twice the base Complexity of 3) and -2 (from 3 to 2.8 is a 0.2 improvement, an Average task) for a total of -11. On a critical success he may make yet another attempt in the future.

Other Applications?

These rules were intended as incentive to seek advantages over opposing hackers, by using computer-related skills as opposed to combat. However, they can easily cover repairing damaged hardware or corrupt files. If the amount of physical destruction can be expressed as an amount of Complexity, and the extent of rebuilding as a gadgeteering task, the hardware customization rules can be applied verbatim.

Example: Aaron's deck is damaged by gunfire, after narrowly escaping from goons in an internet cafe. Campaign house rules state that 10 points of damage destroys 1 point of Complexity. His deck took 34 points of damage, so he needs to replace 3.4 points of Complexity. Some of it is garden-variety circuitboard, wires and plastic casing, which the GM decides is easy to find and replace -- 1.3 points of Complexity and a Simple task. The remaining 2.1 points was his new chip. Aaron sighs, and ducks out his back door to see his fixer once more.

Complexity may also describe components or systems in other fields, allowing players to improve and repair them in the same fashion. Automotive parts in an Autoduel setting, jump drives in ***GURPS Traveller,*** and security systems in an espionage campaign are three obvious candidates for this treatment (although at higher TLs the GM way wish to remove the -3 penalty to every roll). GMs must decide what bonuses fractional Complexity provides, and assign Complexity ratings to whole systems and component parts. Then mechanics and hackers alike will have more things to do during the down time between adventures.

Article publication date: June 7, 2002