Newsletter :: Volume I, Number 1 :: May 30, 2005

In This Issue

Staff

Co-Editors-in-Chief.....
Kerm Martian
Jon Pezzino
Reviewers....
Kerm Martian
Jon Pezzino
Pending Members...
Alex10819
Wizard3246

The BASIC Elite Inaugurated

By Kerm Martian [kerm_martian@yahoo.com]

May 17, 2005 - At Cemetech, it was a relatively quiet month. Doors CS was nearing completion, the site was undergoing several updates, but otherwise nothing major was happening. The first event occurred when Kerm Martian received an email from a Jonathan Pezzino, offering some suggestions and bug reports for Doors CS 5. He noted the suggestions, fixed them, and emailed Jon of this news. He soon received another reply offering further feedback, and before he knew it, Jon was a full-fledged member at Cemetech, posting at a nearly unheard-of rate. After several days, Jon presented Kerm with an idea he had come up with.

"I was thinking: There is so much BASIC c**p out there that we should start a club/group/organization/association/whatever-you-want-to-call-it of top-notch BASIC programmers so that if we see a program by them we will know it's not total c**p. Obviously, it would be composed of people such as yourself who really know what they're doing in TI-BASIC.

"The group wouldn't require too much effort: you'd merely have to create a group on the Cemetech forum that required a password issued to accepted applicants. The club could publish a newsletter/review that lists good BASIC programmers and reviews of their programs. We could also provide support for developers such as a manual of 1337 [Elite] BASIC tricks. Additionally, we could publish information as "Informational Texts" on ticalc.org.

"To qualify for this exclusive club, applicants would have to send in a selection of their finest BASIC programs to be

reviewed by a panel or moderator along with a paragraph that shows them to be an intelligent and respectful human being. Members could also be recruited from the ranks of ticalc.org based on program releases.

"This group would be good because it would increase the amount of high-quality BASIC stuff out there, provide support for competent BASIC programmers, and promote Cemetech as well. What say you?"

I said yes, and thus The BASIC Elite was formed. Soon I had created a logo, shown at the top of this document, made a subforum on the Cemetech forum (http://cemetech.designerz-core.com/forum) and wrote several news articles. Jon busied himself writing an inaugural version of The BASIC Elite Tips and Tricks guide.

As it stands now, the Elite (as it is called in short) has three main functions: a Programmers' Guide, a newsletter, and a programming consortium. The Guide is a single document, updated each time a new item is added, that contains the largest quantity and variety of programming tips and tricks yet compiled. As new items are suggested, they will be added; so far the document is in its fourth revision. This newsletter will be a semi-monthly publication containing the latest tips, upcoming projects from each of the member organizations, and reviews of outstanding community BASIC programs. The third and final section, the BASIC Elite Programmers' Consortium, will be a standard to ensure the program you are downloading is a quality BASIC program, not a five-minute math class creation. Each program will be individually certified and given an image of approval to include in its documentation.

On May 24th, 2005, The BASIC Elite began accepting applications for membership. The application form can be found at http://cemetech.designerz-core.com/projects/basicelite/application.html; if you submit it, we will try to process it within 24-48 hours. Even if you have very little experience, give it a try!

The BASIC Elite Member and Certified Logos

By Kerm Martian

The BASIC Elite has among its functions the role of a Programmers' Consortium. Members of this group will retain their individual programming groups but will also be able to display the BASIC Elite member logo:

Posted in the pages of any calculator website, it will tell visitors that the site or group creates quality programs and is not one of the dime-a-dozen programmers who clog ticalc.org with literally hundreds of quadratic equation solvers. Each program that a member group creates can be submitted for verification; when at least three independent members have agreed to the high quality of the program or project, it can be given the following logo:

This is merely a template, however; actual valid programs will contain a unique bit-coded sequence in the bottom gray bar, similar to a barcode, identifying the program, its author, and the group that wrote the program. One you or your group has joined the Elite, simply post a link to your program on the forum to be considered for certification.

Latest BASIC Elite Members

(alphabetical by last name)

Kerm Martian: The co-founder and co-editor of The BASIC Elite, he is also the president and founder of Cemetech. Kerm has been programming for six years; among his more than three hundred BASIC projects are an office suite, 3D racing games and a 3D first-person shooter, an AOE clone, and a Risk variant. He has begun to work in ASM over the past three years, yet he still frequently utilizes BASIC as a powerful and versatile tool.
URL: http://www.Cemetech.tk

Jonathan Pezzino: The co-founder and co-editor of The BASIC Elite, Jon acts as the driving force behind the Elite. He is the primary maintainer of the Elite's programming guide and is himself an accomplished programmer. An avid user of the object oriented languages Java and C++, he has programmed a slew of innovative works that stretch the power of BASIC and bring an OOP flavor to it. Jon is currently unaffiliated with any programming group other than Cemetech and the Elite.

Daniel Thorneycroft (TI-Freak8x): One of the original Cemetech members back when Cemetech was in its first Geocities domain, Daniel is primarily known under his programming alias TI-Freak8x. He is the webmaster and founder of a growing

programming group, TI-FreakWare, and is noted for quality BASIC programs that he improves and revises extensively before he releases them. Among his most popular projects are the Age of Darkness series, TI-City, and extensive RPG, and Star Trek, a complex and graphical strategy/action game. TI-Freak8x is also a frequent poster on the current Cemetech forum, often helping BASIC programmers in distress looking for assistance. TI-FreakWare is the second group to be accepted into the Elite.

Program Reviews

Filename: **Rally Champion 9.7**
Author: Jose Sousa
Email: ze888Ø26Ø5@hotmail.com
URL:
http://www.ticalc.org/archives/files/fileinfo/278/27897.html
Released: August 22, 2ØØ3
Reviewer: Kerm Martian

Review:
This was one of the first programs I downloaded from ticalc.org years ago, and I immediately realized it was one of the better BASIC racing games available for the TI-83 series. It claims to be in 3D, and the first thing that strikes the user when playing it is the detailed dashboard of the first-person view. A road defined by two lines stretches out to the horizon in front of you, and you begin. The object of this game is to reach the finish line in the least amount of time while staying on the road as it twists and turns through a randomly-created course. It also supports fixed tracks, and includes an editor to create your own courses. The controls are straightforward: [<] and [>] to steer, [up] and [down] to accelerate and brake. The 3D graphics are nothing to write home about, but they do add an enjoyable twist to the game. The real treasure of this game, however, is the way damage is expressed. If you stray off the road, cracks appear in the windshield. Do it again, and more cracks appear. If the windshield breaks, you lose. I found the level of realism an innovative way to keep track of damage, far more interesting than a simple health bar. I had only two gripes with Rally Champion 9.7, its size and its speed. I runs at 1.5-2 frames per second, since it needs to update the track, draw it, and render the dashboard. However, I felt this was not unreasonable given the level of detail. Even the steering wheel turns when you steer! My second problem was with the size. Weighing in at around 11K, the bulk of the size comes from the numerous picture variables necessary to achieve the graphics at the speed this game runs. Filling up nearly half the average calculator's memory, it's large, but I feel well-worth it.

Ratings:
Size: 4/1Ø
Speed: 6/1Ø
Originality: 9/1Ø

Execution: 8/10
Graphics: 8/10
Ease-of-use: 9/10
Replayability: 7/10
Overall: 7.3/10 Very Good

Filename: **FishTank**
Author: Andrew Lundeen (Andrus Programming Group)
Email: bigalthethird@yahoo.com
URL:
http://www.ticalc.org/archives/files/fileinfo/370/37027.html
Released: May 26, 2005
Reviewer: Jonathan Pezzino

This program looked very intriguing from the start. Although there are a plethora of aquarium screensavers for the computer, I have seen few for the calculator, and so I decided to take a look at this one, especially since the authors provided an animated screenshot, allowing me to preview it without going through the hassle of downloading it first. From the second I saw the screenshot, I could tell that the authors had done a superior job and put a lot of work into this program. They designed their own custom GUI, created absolutely beautiful graphics, wrote a professional-looking & informative readme, and based it all on a fairly fresh (though not necessarily original) idea. If there is one thing that makes this game, it is the stunning GUI and graphics. The logo for the program is very suave-looking, a real eye-catcher especially since it is white on black, a rare feature in BASIC programs. I was impressed to see that the authors had designed their own font and menu system with a creative fish cursor, a feature you will rarely see even in professional computer programs of this type. The programmers even took the time to design their own text-input system. I cannot say it was the most efficient of systems, but it did its job and added to the program's elegant aura. This is also one of the few programs to effectively utilize the clock function on the 84+ series calculators. The ability to monitor time has vast untapped potential, a concept that the authors seem to have realized. Unfortunately, this program did have a few con. First off, I was unable to actually play the game because I kept getting an "ERROR:DIM" message. The programmers were clever enough to lock the program, so I was unable to find the source of this error. Because of this error, I was unable to actually play the game; I was restricted to marvelling at the amazing GUI/graphics. My guess is that the authors forgot to initialize a list or array because they had created it on their calculator manually, not realizing that users would not have

this variable properly set on their calculators. My only other gripe with this program is the authors' claim that the program "naturally" slows down because it is in BASIC. Not so! Clearly, the authors had not read "The 1337 Guide to TI-BASIC," in which it is explained why this occurs and how you can easily prevent it! Oddly enough, despite the lack of playability, I still deem this program superior because of the authors' obvious effort and absolutely stunning graphics and GUI. I am sure that this file will be updated in the near future to fix these problems, and I can only assume that the game experience will be as high-quality as what I have seen of the program thus far leads me to believe. Andrus Programming Group: Congratulations on a job (almost) well-done!

Size:	5/1Ø	(Uses almost all pic variables, but they can be archived)
Speed:	7/1Ø	(The menus ran quickly and seamlessly, but one must take into account the slowdown mentioned by the authors)
Originality:	4/1Ø	(An old idea on a new platform)
Execution:	1Ø/1Ø	(Awesome job!)
Graphics:	1Ø/1Ø	
Ease-of-use:	5/1Ø	(Two programs - ugh)
Replayability:	9/1Ø	(Fish are intended to last many weeks)
Overall:	7.14/1Ø	(This rating system does not do the program justice! I hope to see this program updated so that I can improve the score.)

Latest Tips and Tricks

By Jon Pezzino

Every issue, we will be featuring tips, tricks, hacks, and programming strategies for TI-BASIC that will help you improve both your programs and your overall programming style. Some may be as simple as low-level optimization, others complicated conceptual strategies, but whatever the case, all of our tips are guaranteed to help you become a '1337' TI-BASIC programmer.

This week, we will be presenting two tips, a graphics trick and a programming strategy, both featured in the The BASIC Elite's "1337 Guide to TI-BASIC."

Displaying Text on Grids

Have you ever tried to display text on a grid on the graph screen? Did you ever notice that sometimes letters appear to overwrite grid lines for no reason whatsoever, while next time you run the program, it may not happen at all? Well, don't take chances that this random error will plague your program: eliminate the risk of this aesthetic nightmare by redrawing surrounding grid lines every time you output text to the graph screen. Here is an example from a chess game where a rook is being drawn (assume each grid space is 5x5 pixels):

```
:Text(1,1,"R
:Line(Ø,Ø,5,Ø
:Line(Ø,-5,5,-5
```

As you can see, the grid lines above and below the space in question are redrawn just in case the error decides to strike. I am not sure what exactly what causes this error to occur, but I can tell you that you will be happy when you do not have to worry about it!

Avoid Memory-Leakage through the Proper Use of Control Structures & Program Calls

This might seem like a fairly subtle and unimportant programming strategy, but it is vital to a program with lots of iterations. Failure to follow this rule is the cause of so called 'program fatigue' where the program runs slower and slower as time goes on. Failure to follow this rule is also the cause of most of the mysterious 'ERROR:MEMORY' messages you

might receive when your program is running.
    The rule is based on the principle that every time that the TI-OS detects the beginning of a control structure (While, For(, Repeat, If <condition>:Then) or a program call (subprogram, subroutine, function, whatever you want to call it), it allocates memory space to remind it to search for the End that accompanies the beginning of these control structures. If, however, you exit the loop before its corresponding End has been reached, you will leave that little piece of memory dangling and doomed to clog your RAM until the program exits. The most common way to exit a loop is use of the misunderstood Goto statement (though I suppose horrendous use of the Menu(function could also qualify). An example of poor use of control structures is provided:

```
:While 1
:If getKey=45
:Goto 1
:End
:Lbl 1
```

    Notice how the End that complements 'While 1' will never be reached? TI-OS doesn't know that and as such will wait for that End to be reached at some point in time, using up memory. The above example can and should be replaced with something that does not cause a memory leak:

```
:Ø→A
:While not(A
:If getKey=45
:1→A
:End
```

    Note that this is not necessarily the best or most elegant solution, but it readily demonstrates the principle of memory-leakage through improper use of control structures.
This rule might also lead you to infer another rule: generally avoid the use of Goto and Lbl. These conventions are inefficient, confusing, obsolete, and ineffective the vast majority of the time and are usually avoided by skilled programmers in any language. Overuse of Goto will create so-called 'spaghetti' code, where reading your code becomes akin to untangling spaghetti because it is so difficult to constantly follow the path that the gotos will take you. Also note that Goto will slow down your program as a whole because the program interpreter has to scan the entire program to look for a label when it encounters Goto. This is much slower than other control structures, which the TI-

OS is able to handle much more efficiently.
    Another common cause of memory-leakage is a tangled program/subprogram hierarchy. Avoid recursive programs (programs that call themselves) because if they recur too much, they will gobble memory up like none other, causing the original program to crash. Also, make certain that programs have an established hierarchy so that when a subprogram is called, it is guaranteed to return to its parent program. For example, if you were to have a program whose purpose was to take in input, use subroutines to display the output, and start all over again, you would want to make certain that the parent program is handling everything, unlike in this diagram:

    Parent Program -> Subprogram 1 -> Subprogram 2 ->
    Subprogram 3 (displays answer) -> Parent Program...

    As you can see in this diagram, Subprogram 3 does not ever call the original parent program, instead minting a new 'instance' of the parent program, dooming the original parent to needlessly gobble up memory. Instead, the diagram should look like this:

    Parent Program -> Subprogram 1 -> Subprogram 2 ->
    Subprogram 3 (stores answer, then returns, traveling back up the program hierarchy to Parent program)