***Copperware!* Introduces**

Guess v1.4 for Windows

Chris Uhl
342 J.J. Thiessen Way
Saskatoon, SK
S7K 5P4
Fax: (306) 934-0004
BBS: (306) 652-2487 14.4K N81

As *Copperware!* introduces Guess for Windows, so too do I introduce *"Copperware!*" *Copperware!* is my company name which I will use for all of my programs. The name "Copperware" stems from some realizations I have come to. First, my name Chris Uhl. Few people know this, but copper's chemical symbol is Cu. Hence copper. Second, since I wanted to have it a computer oriented name, I thought of "ware". So, "copper" + "ware" = Copperware. The exclamation point is my personal preference.

Now, about me. My name is Chris Uhl. The last name is pronounced Ühl (as in you´·el). The proper German way of pronunciation is Ool (as in fool). I prefer the Ühl, although it isn't spelled with the two dots "Ü", that is there for those of us who are phonetically proper.

I was born in Regina, Saskatchewan on February 23, 1975. So, for most of you "he's a young'n". That's true, I am young, but I don't think that should be held against me. I graduated high school in '92, have had a very small job at the corner store for a while, and program when I'm not working.

The language I have chosen to program in is Visual Basic Pro 3.0. This is not a cop out. Some are dead-set against people using VB, because it is so easy to use. They think we don't earn what we make. I say to heck with that. It's true, I don't "C", so that disqualifies that. I have been taught in Pascal — wonderful language — but I didn't quite catch on. I can program the simplest programs in Pascal, but it wouldn't give me the options presented in Visual Basic Pro 3.0. I'm not here to debate *how* I program, I'm only here to present my artistic expressions. Some people paint by numbers, others paint by memory.

Now I have talked about the past, lets talk about the future. As I know it, a new version of Windows is coming out. Windows 4.0, codenamed "Chicago" by Microsoft, would effectively be a Windows NT Lite. Since I have not got a hold of a beta version (yet), I cannot assume anything. If this program works with Chicago, that's great, if not, then I'll release a new version.

I have a lot of ideas in the works. A game of Black Jack, an educational program to help students solve triangles, and others. These will be released sporadically, as I am coding each program myself.
Now that the past and future have been taken care of, lets move on to the present. This programs name is called, quite simply "Guess". For those of you who think this is too simple, you may call it, "Guess the Number". Anyway you look at it, you are trying to guess a number.

The object of the game

The object of the game is to try to guess a randomly generated number in as few tries as possible. The possibilities of the number range from 1 to 10,000. Now I know what you're thinking, it will probably take you 9,999 tries to guess the number. No, that would be boring — boring is what I detest. When a number is picked, it is hidden from you (duhh!). When you enter a guess, the computer will tell you if the number is higher or lower than your number. Through the process of systemized guesses and *luck*, you'll get it.

How about a challenge

That’s easy, right? Of coarse it is. Now for those who want an added challenge. To explain this, I’ll have to get technical, so bear with me…

Our number system is based on the number 10. And why not? Ten is a nice round number, easy to comprehend, and even easier to use. But what would happen if the number system used by Guess was based on the number eight? The same random number would be generated as a “ten” based system, and then converted. Here’s an example. Let’s say that the number picked by the computer was 6765. Anybody could figure out what the number was providing the computer gave it clues. No problem. Now, lets translate that number to the “8” based system. The number would now be 15155… well, wait a minute here, I’m confused.

The ten based system uses the following principle: numbers start from 1, and go to 10. The following chart will show this:

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

easy enough. Let’s now look at the eight based system:

1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14.

Now do you see where it gets interesting? This forces the brain to think on a different level. This task can be accomplished — not easily, but with a lot of practice.

The resulting number system will be from 1 to 23,420 not 1 to 10,000. The “8” based number system is scientifically called the Octal system. Our common “10” based number system is called the Decimal system. Think of it this way, an oct’opus has eight tentacles, so too does the Octal system have eight numbers. A decade has 10 years, as well as the Dec’imal has 10 numbers.

Now that you know more about the Octal system, you are less intimidated by it. You just learned something! Give yourself a pat on the back.

But just a sec. The Decimal system and the Octal system are not the only systems I use. I also use Hexadecimal and the Binary systems. Some of you teckies are revving up your mind to tackle these systems. Obviously, these are the two most difficult systems to master, with Binary topping it off as the most difficult to learn, and Hexadecimal being the hardest to use.

Using what you have learned from the Octal system, apply them to the new systems: Hexadecimal:

1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11, 12… (on the number 16)

Binary:

1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011 (on the number 2).

The Hexadecimal system, when compared to the Octal system, is as easy to learn. Although it encompasses characters we are not familiar with: “A, B, C, D, E and F”. Think of these numbers as an extension of the Decimal number system, as that *is* what they are. Computer gurus thrive on Hexadecimal numbers. Some of them know the ins and outs of it, but most of us wouldn’t know where to begin. Here’s an opportunity for the gurus to *prove* it. Don’t be surprised if they back down. It’s a hard challenge.

The last number system is the Binary system. The hardest to think in, but the easiest to comprehend.

(Well, maybe not the easiest to comprehend, but after you understand the rules you'll get the idea.) The computer knows the ins and outs of this system. It has to. This is the simplest number system as it pertains to the number of numbers in a system. A computer bit has two possibilities: on or off. Or in electrical terms "charged" or not charged. Since a bit has only two possibilities, the computer must think in a Binary way. It can't think in a Decimal way, because a circuit only has two positions. A computer must translate everything back to Binary, work with that, and then translate back into Decimal.

I still havn't told you *how* it works, I have only told you *why* it works and why it was created in the first place.

Ok, here's my attempt to explain *how* it works. Let's see… Imagine each digit has a predefined number for that position. The position furthest to the right has the value of one. The position immediately to the left of that, has the value of two. So in these terms, the number "10" has the value of two. Confusing? Here's another example. The number "11" has the value of three. Get it? You simply add up the values of the positions with the number "1" in it. In the third position from the right, is the number four, and the fourth position has the number eight. The Binary number of "1111" has the Decimal number of 8+4+2+1. Add this up to get 15.

Do you notice the pattern emerging here? Each number is double the number that preceded it. So the pattern is: 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096 et cetera. If you have been around computers for a while, you will know all of these numbers are *special*. A sector on your hard drive is typically 2048 bytes or 2K (**kilo**bytes) large. When one has a **mega**byte of something, they don't just have a million of something. They have exactly 1,048,576 bytes, or 2 raised to the power of 20 (2^{20}) or, to put it in laymen's terms 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2.

Lets give you some examples, so you can extrapolate.

	Decimal	Octal	Hexadecimal	Binary
1.	10	12	A	1010
2.	15	17	F	1111
3.	100	144	64	1100100
4.	128	200	80	10000000
5.	180	264	B4	10110100

Wow. Lets look at number four and five a little closer. The number 128 has already been described as a "special" number. So the resulting number in each system is also special. The Octal number four is a nice round number. Coincidence? I think not. The Hexadecimal number four is also a round number. Coincidence? Nope. So too with the Binary.

Lets forget about the Octal system for a while and concentrate on the other three. The Decimal number 128 is special, but why? Two raised to the power of seven is 128. (2^7 = 128, or 2 x 2 x 2 x 2 x 2 x 2 x 2 = 128.) Good, but what about the Hexadecimal and the Binary?

I'm glad you asked me that, because this one is neat. Here's some more technical stuff, so bear with me…
Take the Binary number "10000000" and break it up into two groups of four. You get "1000" and "0000". Now, using what you learned regarding the Binary system, translate 1000 into Hexadecimal. What did you get? Did you say eight? That's right. Now translate the second four digit Binary number into Hexadecimal. That was easy. It's zero. Now put eight and zero together to get 80. Now look up at the chart under Hexadecimal number four. It's 80! This is not an isolated incident! Now try that with

number five and see what you come up with. This one is a little harder, so I'll explain it.

"10110100" breaks down to "1011" and "0100". Translate them to *Decimal*, then to Hexadecimal. 1011 is eleven in the Decimal system, and to translate count from one and go to F. You get B. (1, 2, 3, 4, 5, 6, 7, 8, 9, A, B…) Now do this with the second one. That number boils down four. Put B and four together and you get B4. The answer to Hexadecimal five is B4. Did someone yell Bingo? I thought so.

Some of you are saying "Well, wait a minute here. Didn't he just contradict himself?" In a way, I did. I cheated for brevity. In the first example I asked you to translate 1000 to Hexadecimal. But in the second example, I asked you to translate from Binary to *Decimal*, then to Hexadecimal. It is harder to translate directly from Binary to Hexadecimal than from Binary to Decimal, then to Hexadecimal. Since the Decimal number eight is the same as the Hexadecimal eight, I omitted that step.

Therefore, since brevity is the sole of wit… I will be brief. (Polonius 2.2.90-92) But I digress.

Here is a chart showing the level of difficulty associated with each system. (Whereas the number three being the most difficult and one being the easiest.) The Decimal system was left out for obvious reasons.

	Learning	**Using**	**Thinking**
1.	Octal	Binary	Octal
2.	Hexadecimal	Octal	Binary
3.	Binary	Hexadecimal	Hexadecimal

Apparently the Hexadecimal system is the most difficult to use and think. Why? Well because it encompasses characters we are not familiar with. It's kind of like using numbers to spell, or symbols to speak. (☎+📫=☺)

Surprisingly, the Binary system is the easiest to use. Why? Because it has the thinking built right into it. The Binary number 100000 is twice the amount of 10000. You can use it to make the system work for you. Don't tell anyone this, but when you get good at the Binary system, you can actually guess the number faster (and more accurately) than the Decimal system! The Octal system is easier to use than the Hexadecimal, because it only *excludes* numbers, not includes other characters.

The On-Line help does not tell you any of this, because that would almost be cheating. To those who read the documentation come great reward.

Why in the heck would anyone confuse the issue by making <u>more than one number system</u>!!
I'm glad you asked me that. No one in thier right mind would actually learn Hexadecimal, Octal or Binary without a good reason. It's kind of like learning Swahili just to impress your friends. The fact is, no one really *invented* Hexadecimal, Octal or Binary; they were, quite simply discovered. A while ago, I told you about a special relationship between Hexadecimal, Octal and Binary numbers. This is by no means a coincidence. There are complex relationships between all three. I showed you examples on this, but I didn't get into any specifics. It is beyond the scope of this document to give you all the juicy details, but in a nutshell, here how it goes.

The Binary system is based upon the number two. Remember that. The Octal system is based on the number 8. Ditto. The Hexadecimal system is based on the number 16. Right now, can you give me any connection these number systems have? 2 x 8 = 16. Good. But that is just scraping the surface. The complex relationships that occur happen because of this. You can place two Octals in one Hexadecimal, and four Binaries (for lack of a better word) in one Octal. Because of this, their formulas for converting from Decimal are remarkably the same.

I have evaded the question again, haven't I? Ok. The reason humanity uses Hexadecimal, Octal and Binary rather than just Decimal is because computers use them exclusively. The computer does not use Decimal per se, it does so because the user does. The Decimal system is remarkably inefficient (to the way computers think). And, the Hexadecimal, Octal and Binary systems are Swahili to us. So, where should we strike a balance? How much Swahili do you actually need to know? My answer is: who cares! But opening your mind to another number system won't hurt. You just wait. Your boss will bring you into his (or her) office tomorrow and say you are going to Zanzibar to start relations with the Bantu people. There's a Swahili to English dictionary at Coles (although it may be on back order).

Afterward

Now that I have taken over twenty-five hundred words to tell you about myself, and our wonderful number systems, I will tell you why I made this program, and why it is free.

Why I made this program

When I got Visual Basic, Guess was the first program I made. It was purely for me to experiment with all the bells and whistles that came with VB. When I get to know it better, I will start my larger programs: Black Jack and Triangle. These two programs will most likely not be free. However, I will price them reasonably, so they might seem as though they were.

Why I made this program free

It is my contribution back to the computer world. In reality, would you pay for a simple program like this? If yes, here's where you can send your blank check… no, just kidding. I'm not allowed to solicit money from you, so I won't. All I would like for a payment is a letter, a postcard or a thank you card. Not much.

Other boring stuff you can read

Included with this package is an ASCII document called README.TXT. It contains all the boring legal stuff that is almost always mandatory when a program is distributed. I encourage you to read it, but I don't expect you to.

Copyrights, trademarks, registered trademarks and other legal stuff

This document is © Chris Uhl, 1993. The README.TXT contains more information about your rights, and mine. Please read it thoroughly before you do anything else.

The name "Copperware!" can be considered a trademark. However, it is not a registered trademark, which means if anyone else decides to register it, I'm out of luck. If any lawyer wants to generously donate their time so I can have it registered, I would be eternally obliged.

Name Entering

This game has a simple algorithm for weeding out bad names. This includes names shorter than three characters or longer than twelve characters. It also looks for vowels. If it doesn't find a vowel, then it will reject the name. However, if you have an uncommon name that does not contain a vowel, simply put an underscore *before* your name "_". This will supress error checking of your name. However, your name has to be shorter than eleven characters.