Subject: Jinx
Posted by Wayne Parham on Mon, 30 Aug 2021 15:51:48 GMT
View Forum Message <> Reply to Message

This might be a fun distraction for you. :)

You might remember a few months back, I posted about old early microcomputers, those made prior to the IBM PC:
Microcomputer History
I have a handful of old computers, and I'm starting to do write-ups of each of them. An RCA 1802, an Altair 8800 and the one I started with, the Synertek System's Sym-1.

In the process, I wrote the machine-specific assembly-language routines for the cc65 compiler. That provided a C compiler for the Sym-1. It's being reviewed and merged into the official distribution into that compiler right now.

So anyway, after doing that, I started writing some C programs for the Sym. I wrote a version of John Conway's Life simulator and a fixed-point integer math version of a Mandelbrot set renderer. But the reason I'm posting this here is that I wrote an old-school text adventure game for the Sym-1. That might not, in itself, be interesting or useful for anyone here but I did compile it for several platforms other than the Sym-1, including the Apple II and the IBM PC, both the original one and modern systems with Windows. It can run on any PC from the first MS-DOS to the latest version of Windows 10.

It's "old school" so if you're into the latest and greatest graphics, don't bother downloading this game. But if you like text-based adventure games like Colossal Cave and Zork then you'll probably like this too. If you have no idea what I'm talking about, read on.
Jinx Adventure Game Download
Here are some screen shots, so you can see what the game is all about. No cheats though - I don't want to ruin it for you - but I will give you one little hint, directions how to make your first score.

Here's the first screen shot, the entry page. This is the first thing you'll see when you start the game:

Pretty cool, huh? It's not like the new games with stunning graphics, but rather more like reading a book. You interact with your environment by typing instructions.

You can give directions to go wherever you want to go. You're standing just south of the house, so you can approach it and enter by going north. Or you can go other places to the south, west, east or northeast.

As you can see below, I chose to go west. That took me to an old school, which I tried to enter. It appears that the school is all boarded up. That's true of many places in this game, because the town is said to have been abandoned. But there are a lot of places that are still open too. You'll

have to find out which ones are which.

After I went to the old school, I then went north, then east and then north again. There I found a coin!

Checking my score, I see that obtaining that coin earned me five points. That's great! So next I tried to go west, but found that wasn't an option. No path to the west, no way to go that way. Instead, I retraced my last steps back to the house. I went south, then west, south again and then back east. That brought me back to the house.

I went inside, examined my surroundings by typing "look" and then placed my coin in the living room. Checking my score again, I see that the points I earned by getting the coin had doubled. That's the goal - To get resources and bring them back to the house.

There are actually other houses you can use as "safe houses" too. You can choose any of them you want, as long as it isn't boarded up. If you can enter a house, you can make it your own and put your stuff there.

Most things are just resources you'll need to stockpile to survive. A few items are useful during your travels, for things you might need to do in the game. But in the end, you'll want your stuff in a house.

When you find an item, be sure to examine it and/or read it. If it's valuable, you'll want to get it and eventually take it home. Even if it's not valuable, it might be worth taking. You certainly want to examine it and/or read it because it may have clues that will tell you what you might need to do. Some of the items are just interesting, maybe telling you something about the area. Other things tell you stuff you might want to know.

I won't give you any more details, because that could spoil the game for you. But I will leave you with this one last screen shot, a place in the game far from your home. :)



Happy adventuring in Jinx!

---

Subject: Re: Jinx
Posted by gofar99 on Tue, 31 Aug 2021 02:12:01 GMT
View Forum Message <> Reply to Message

Hi Wayne, brings back old memories. I remember programing in SPS, Fortran, various Basic and

even in machine code.  Mostly for IBM mainframes at the time.  Folks don't know it how good they have it now.  Back them if there was a coding error the program just didn't work.  Generally no clue why.  Even with some error codes like "division by zero" which popped up when you never did any division were not much of a clue to the error.  Folks running the main frames were not to happy if you caused an endless loop either.  8o

---

Subject: Re: Jinx
Posted by Wayne Parham on Tue, 31 Aug 2021 23:02:25 GMT
View Forum Message <> Reply to Message

I hear you, Bruce.  My career has spanned those eras - Starting at a time when microcomputers were still just "on the horizon" and all business was done on mainframes and minicomputers from IBM, DEC and Data General.  Now days, everything is really different.  What was the best coding style back then is no longer seen as being appropriate in most cases.  But then again, in certain situations - like memory-constrained microcontrollers - what was good practice back then still is today.

So some of the ways programmers are taught to work now-days doesn't apply very well to a machine with little memory.  These newer technologies are optimized for abstraction, so that complex programs can be written in a way that's more readable and maintainable.  But that sacrifices efficiency in the form of larger compiled code and sometimes it's a little slower too.  Then again, back in the 1970s, systems were rated in thousands of instructions per second and one million instructions per second was a "holy grail."  Now days, an Intel i7 will run nearly 100 billion instructions per second.

It's interesting to work with both technologies - the stuff from the 1970s and the stuff from today - because of that juxtaposition.  I enjoy coding in assembler or using C, styled from the 1970s.  I enjoy the old BASIC programs with line numbers.  But I also enjoy modern C/C++ and Java tech stacks, with technologies like Spring and its dependency injection approaches.  They're as different as onions and orangutans.

---