The Wright language with the wrong tools

It all started with a simple question:

“What would it be like to write a game using LabVIEW?”

Then came the choice of the game. “Quake 3” wasn’t a good idea, because I wanted to write the game without disturbing the entire LabVIEW community and the LabVIEW development team. So I went for and old favourite of mine, “missile command”.

I started messing around with the picture toolkit to see how I could implement the “objects” of the game. After a while, I came up with one good way of implementing them - a mixed approach of pure dataflow and object oriented programming.

It’s very simple: the properties (information) of the object would be represented by type definitions, and the methods (actions) would be encapsulated into subVis. All of the information is carried from one iteration to the next by shift registers. I considered using the encapsulation of the information in global services, but there are two significant drawbacks to this method, the first being the non-negligible overhead introduced, and second the hiding of important information. When you are using a graphical (visually appealing) language like G, you want to use it to it’s fullest potential, and hiding information is not a good idea.

The biggest challenge in building this game, was certainly the lack of support for those kinds of applications. Just imagine what we could do if we had the same libraries of drawing function available for C++, like OpenGL or DirectX? Because when it comes to data manipulation, my opinion is that LabVIEW is by far the best - and what are games, other than big data manipulating engines...?

- Dominic Lavoie - 2001