COMM 362 – Assignment #3

Dear Jim,

I'm writing to let you know that I received your request for the updates to your department's new Scratch game. Unfortunately, some aspects of the request were a bit vague and, frankly, contradictory, so I'm hoping to clear some things up today by walking through what I think you mean and providing you with some hypothetical options to choose from for how to move forward.

First of all, you mentioned that you want a "super fast program with no lag" that has "as few lines of code as possible" and "the ultimate production value." You also mentioned how "optimal usability" is a must and "perfect readability/maintainability" is expected for a company with as many employees as yours. These are great goals, Jim, and I wish our IT department could deliver on all of them to the caliber of your expectations, but, similar to house hunting, I don't think we'll be able to get you everything on your wish list (at least to the degree that you desire). Even if your budget was unlimited and you were willing to wait a long time for us to get everything up and running, I don't think we'd quite be able to pull off what you are asking us to do. It just wouldn't work with the way algorithms and software operate. Allow me to explain.

As one of my professors used to say in college, algorithms are like a program's underlying strategy, so they can be judged as "good" if they achieve their goal; however, sometimes these goals conflict with each other, and that means that decisions have to be made about what is the more important or more desired among the various goals (Sandvig). Unfortunately, that is where we find ourselves today, Jim. But before we get into that, I want to make sure we're on the same page. Essentially, you said you want a program with increased speed. On our end, that means we need to make an algorithm that reduces the time spent calculating whatever requests are made of it, or whatever "thinking" it has to do. You also want to reduce complexity, which could look like condensing the program into fewer variables or fewer lines of code. I'm assuming that "optimal usability" refers both to computer resource use as well as the game player's ease of use, but I'd like to hear which one you were specifically referring to, since those can be a bit contradictory at times. For instance, if our department reduces the computer's resource use (such as by making fewer variables and instructions), that doesn't necessarily mean it will be easier for your users to play the game.

Another contradiction I found with your requests is that you want "perfect readability/maintainability" for the code, along with all of the other goals I've mentioned so far. Wanting very good readability and maintainability for the code is an important goal, and I think you should choose that route over the others in this case, but I want you to know that if you agree with me about the importance of this goal, our IT department might not be able to accomplish three of your other goals to the fullest of their potential. I know this can be upsetting, so let me walk you through why I think this is (and please don't bother asking Mark for more money on this project – I don't think we could make it work, even with more money or time).

First off, improving code readability involves white space and lots of notes, which means your code will have more lines. In order to see what each variable or statement does, it makes sense to often put them each on their own line with white space, rather than cramming what should be several lines of code into one line. Additionally, because coding and algorithms can be so complex (and it's literally kind of like a foreign language), it helps to include many hidden notes about what certain pieces of code do (which I have done for your game, "Gilly's Great Gem Quest," leaving copious amounts of notes so that the code is very readable and maintainable to those in your department). But this, of course, would also add lines to your code. Furthermore, and this is getting a bit nitpicky, it is very helpful to readability and maintainability if variables and such are named in ways that the humans reading them can understand and identify. For instance, when making your Scratch game, I chose to name some of the sprites things like "Lava rock" or "Lava fissure." While this would potentially make the code very slightly longer/wider (asking the computer to read a bit more), I felt it was important to do rather than naming them short things like "s3" or "s4" (for sprite 3 and sprite 4). (But don't go thinking I'm making your code longer with every decision I make! I have optimized some of the algorithm. For instance, I put some of the things that Gilly, the sprite, says into a single list in a command string that can pick randomly from that list, instead of programming Gilly to say a certain phrase at each of those intervals, which would have required six different algorithms instead of one.)

Why I bring up this adding of lines of code and such is that by making the code longer with these added things that aid in maintainability/readability, I am potentially impacting the speed of your program and the complexity of your game's code, and I am especially impacting computer's resource use. As mentioned before, a good way to increase speed and reduce the computer's resource use is to have less that the computer needs to read (perhaps even less lines within the code). If we were to allow for all of the necessary white space and comments necessary to make the code very maintainable and readable, we would have a lot more lines within the code than we need, and that would mean the computer needs to use more resources and time to read it.

In short, you just can't have the maximum amount of fulfillment in both of these areas. Our team can certainly try to optimize the fulfillment of both requests, but you will need to pick one or the other for us to focus on, as the two contradict each other and simply can't be fully realized if the other is also being fully realized. I hope that this memorandum has made that clear enough, and so now I would like to ask you how your department would like to proceed. Would you like us to write shorter code that is faster and uses less resources but potentially does less and is less readable/maintainable, or would you like the added usability and production value (such as with more sprites that do more things) to detract from the speed and require more resource use? I fully understand if – like our competitor, Netflix – you choose not to use the best algorithm because it uses too many resources, but I hope that you are at least able to make an informed decision now that I have explained the way these systems work (Sandvig).

I look forward to hearing from you.

[ANONYMIZED]