Jonathan Walsh

1/17/15

Keep Talking and Nobody Explodes Module Solver:  
Project Review

Time Spent: 15 – 20 Hours

Lines of Code: 1227

Introduction:

About two weeks ago, my friend and I tried the game Keep Talking and Nobody Explodes. KTANE is a video game made for 2-4 people in which everyone has their own screen and one person looks at a bomb while the other 1-3 people use the online Defusal Manual to help solve the different modules on the bomb. There are a variety of modules, from wire cutting to a maze to a password that can only be solved with good communication between both the person looking at the bomb and the person(s) reading the manual. My friend and I played for a few hours, but got to a point where, no matter how quick we were, we felt that with two people it was extremely difficult to diffuse the bomb. Thus, I decided to write a program to help us solve the modules more efficiently.

What I Learned:  
 This was my first major programming project, and I learned a lot about independent projects, program design, and Python. I have used programming in a few classes throughout high school and college, but as a newly transferred Computer Engineering major I’ve decided to start working on personal project, and this was where I decided to start. The project is not perfect, but I am absolutely happy with how it turned out. While I had taught myself Python last year, I had not really worked on anything with it. I had to refresh myself on quite a bit of the language, but I now have a solid grasp on it’s basic aspects(syntax, conditional statements, lists, dictionaries) and plan to stick with Python for many of my future projects this semester. I also learned how to approach personal projects – by answering a question that I have. My question for this project was “Could we defuse these bombs more quickly?” and so I decided to try and do it. Finally, I learned a lot about program design. My project is long and only contained in one file, and that is one of the things I most regret. Next time, I would separate the modules into different files and even split some of the modules into more than just one subroutine (which would solve my consistent use of while statements and repeated segments of code). Other things like consistent commenting style and the use of parameter passing (nonexistent in this program) are ways to write better and cleaner code in the future. I also taught myself a little bit about how to use Git/Github (as evident by the existence of this project) and will use it to keep track of my projects and save my revisions in the future.

What I Think I Did Best:  
 My favorite part of the project was writing the different algorithms for the modules. I don’t have too much experience in algorithms, so while I may not have solved my problems with the fewest lines of code possible I did solve them in my own way. None of the modules were too difficult; however the one I was most proud of was the maze module. I have not learned too much about artificial intelligence yet and it was really interesting to figure out how to “teach” the computer where dead ends were and see how, although it would make wrong turns in the maze, it would get closer and closer every time. I was also proud of my user design. While this is just a Python script for personal use, and I do not have nearly the documentation nor information for it to be very useful for other people, I did make it relatively easy to use/read. Additions like a list of all the modules, examples of correct inputs when a wrong input is given, and a way to quit from almost every step of the program were ways I tried to make it more feasible for both me and others to use. For other projects that I design for use by other people, I know that thinking about these kinds of things now will make my future projects even better.

What I Think I Did Worst:

There were definitely a few things that I would change if I rewrote this program. The first, which I touched on above, was that all of my code was in one file. This made for unnecessary complexity and could have been easily solved by splitting up the modules into separate files instead of just separate subroutines. I also could have broken down the modules into more than one subroutine each, as this would have reduced the amount of lines of code as well as the redundant copy and pasting in a few of the modules. Another thing I would change is my use of global variables. I was taught to try and NEVER use global variables, and I still did because it was easy. I think I could have fixed this by using parameter passing in the modules (such as the “return” for the battery, serial, and port modules) or finding a way to share the variables exclusively between the modules that use them. I am sure there were a few more problems that I saw but have now forgotten, so my goal for future projects is try to write down the things I do best/worst as they occur and not at the end of the project.