Poor Yorick Write-Up

Max Goad, Jace Bayless, Apurva Rai, Tri Pham, Meet Kapadia

For project 2, we were given another group’s project 1 and we were required to expand upon that. The group we ended up inheriting from was TGHET. When we looked into their code, it involved libraries we have never heard of, and their C++ code was heavily dependent on GTK libraries which are used to make the GUI. Furthermore, in their README it stated that it will not compile on Microsoft platforms, but we were able to get that fixed. Lastly, when we did compile it, the GUI worked, but when we attempted to play the game, it did not work at all (went straight to the board without ships being placed and whenever we clicked a square it always said hit). At first, we tried to fix this issue, but after giving it a serious try, we realized that was going to be extremely difficult and we ended up redoing the whole project 1 in C++. Afterwards, the two additions we made were adding several themes/colors and a scoreboard. Here are our meeting logs that Meet has been keeping track of since project 2 started:

#1 9/30/19 In-class Eaton 2 (everybody present)

Download a bunch of stuff because the group used a bunch of complicated libraries. Decided to add themes and sound effects and fancier screens.

#2 10/2/19 In-class Eaton 2 (everybody present)

Decided to try to get the original version of the project to work. If it doesn’t work by Sunday, we meet Monday to get it done only using the C++ aspects of the code.

#3 10/4/19 In-class Eaton 2 (everybody present)

Got the original version to work on one laptop, now going to try it with other laptops

#4 10/7/19 In-class Eaton 2 (everybody present)

Still deciding whether or not we want to do it with the original libraries, but new additions we are going to make are themes, improving the GUI, and sounds effects and graphics

#5 10/9/19 In-class Eaton 2 (everybody present)

Decided to meet up Saturday, Sunday, Monday, and Tuesday and see where we go from there

#6 10/12/19 Spahr Alcove (everybody present)

Try to understand the broken code, get code to work on everyone’s computer, and get rid of junk on GitHub.

#7 10/13/19 Spahr Alcove (everybody present)

Tried to get the original, GUI code to work but it’s really, really, really, hard to fix but making progress

#8 10/14/19 Spahr Alcove (everybody present)

Tried to get the original, GUI code to work again, but this time we gave up and instead are just doing it in the terminal using C++ as Dr. Gibbons already permitted us to do.

#9 10/15/19 Spahr Alcove (everybody present)

Got the game to work with the best of our ability and finished the AI and started to add colors. Also had a lot of memory leaks had to fix that

#10 10/16/19 In-class Eaton 2 (everybody present)

Got the features approved by Dr. Gibbons. Features are themes and a scoreboard. Maybe meet today but for sure meet tomorrow.

#11 10/19/19 Spahr Alcove (everybody present)

Started working on our additions.

#12 10/20/19 Spahr Alcove (everybody present)

Added finishing touches to the project and finished documentation.

Max was the one who figured out how to compile the program on all platforms. He, together with Tri, even made instructions for us on how to get it to work so we can do it on our own computer and when we officially meet, we can just continue the project. Max also did a lot of the work on the GUI and tried his best to get it to work. Once the GUI seemed hopeless, he was responsible for re-doing the Game.h, Game.cpp, Point.cpp in C++. Once the game started to work, Max was then involved and helped the rest out in other classes and implementations. He made the colors and themes in the terminal, assisted Meet with the scoreboard, and helped create the AI as well.

Apurva did a lot of the AI aspects of the code. He was responsible for implementing the AI difficulty levels and just like Max he did a little bit of everything. Apurva made most of AIOpponent.h and AIOpponent.cpp and helped integrate that into the project. AIOppnent was then merged into Game.h and Game.cpp, but the AI part was done by Apurva. Apurva also helped with problem solving as he is the idea man of the group. Apurva also helped with the scoreboard and colors.

Jace, being really experienced in Java, made a terminal version of our project one. We used this outline he had made, and we converted it into C++. Jace also played a major role in cleaning up the terminal code in general as well, making it look presentable. Furthermore, Jace did a lot of problem solving as well and helped test the program and successfully broke it. He also helped with the scoreboard and colors as well. Jace also helped fix a lot of memory leaks with the help of Max, Tri, and Meet.

Meet did the Ship.h, Ship.cpp, and Point.h and was in-charge of everything a Scrum Master would be in-charge of. He tracked and set up all the meetings and did majority of the documentation as well. The works cited, the write-up, and the code documentation was done by Meet. He made all the comments in the .h files which were then used for the official documentation. He also helped test the code, break, and then fix it. He also did a significant chuck of the scoreboard code, assisted with colors, and memory leaks.

Tri was in-charge of making the GameBoard.h and GameBoard.cpp class. Tri’s computer is the only one that could run valgrind so he used that to search out all the memory leaks and helped get rid of those. Tri did a lot of work on the memory leaks. Tri helped stress-testing the program and adding constraints to the user inputs. Tri also helped with the colors aspect of the code, made sure the code format looks up to standard, and assisted Meet with the write-up as well.

The biggest challenge we faced early on was getting the original code to compile. Due to many different libraries, it was extremely complicated to get the code to work on any platform, and their directions were extremely vague as well. Max looked at the makefile and tried to understand what exactly they were doing, and it took a few days to figure out what was going on. Once we all got it to compile on the computer, another huge challenge we faced was that the code seemed like it did not function at all. Once the application ran, we could only hit the start button and that was all we could do. The game could not be played whatsoever. We initially tried to get the code to work within their parameters and the GUI, but that turned out to be time-consuming and complicated due to GTK libraries’ dependency (the straw that broke the camel’s back was when we tried to call a function to a class called “GtkWindow” and were given an error saying “no conversion from GtkWindow to Gtk::Window”). So, we ended up nuking the code and re-doing project one in the terminal. We talked to Dr. Gibbons about this and he said it was okay. Those two things were our biggest difficulties as it took a lot of time just to do them first let alone starting project 2.

There are a lot of crazy features we wanted to implement but we could not due to the lack of time. Some of them were the GUI, teams, and a battleship battle royale. Since we could not get their GUI to work, we decided to do the project in the terminal, and then we were going to make our own GUI. However, we spent too much time trying to get their code to work, so much to a point where we had to finish the project in the terminal. We also wanted to add teams. So, it would be two players versus AI match where two players can attack the AI at the same time, so they have a better chance of beating the AI on hard difficulty. AI then randomly picks either of the board to fire at. We didn’t do this because it got too complicated before the deadline. Lastly, we wanted to do a battleship battle royale. It was going to work like Tetris 99 where there are X number of “players” and the user picks a random spot to fire at, and if that spot is on any of the others’ board, the hit goes there. If it is multiple boards with the same spot, the hit is assigned randomly. There is a 5 second buffer after each fire and the user’s board can get hit the same way. We didn’t do this because again we ran out of time.

One thing we would have done differently is to allocate time a bit better. We spent a considerable amount of time trying to fix a program that was arduous to fix in the first place, and that took up two weeks of our time. We initially loved the challenge and our group, as a whole, welcome it, but the time was not in our favor. Thus, we had to quickly switch to plan b. Since we did our project 1 in Java and using a GUI, we had to make the terminal program in C++ from scratch.

Overall, we had a lot of fun with this project, even though it took a lot of time and effort to get it the way we wanted it to be; however, we all did our part and as a team were able to finish this project. Even though we decided to clear the original project a little late, it still gave us enough time to do what we mostly wanted to and get it to look up to our standards.