Ben Gaudreau

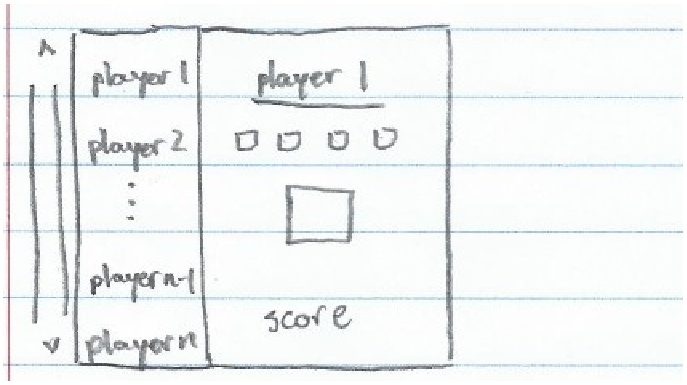
Professor Levine

COS 420

February 21, 2025

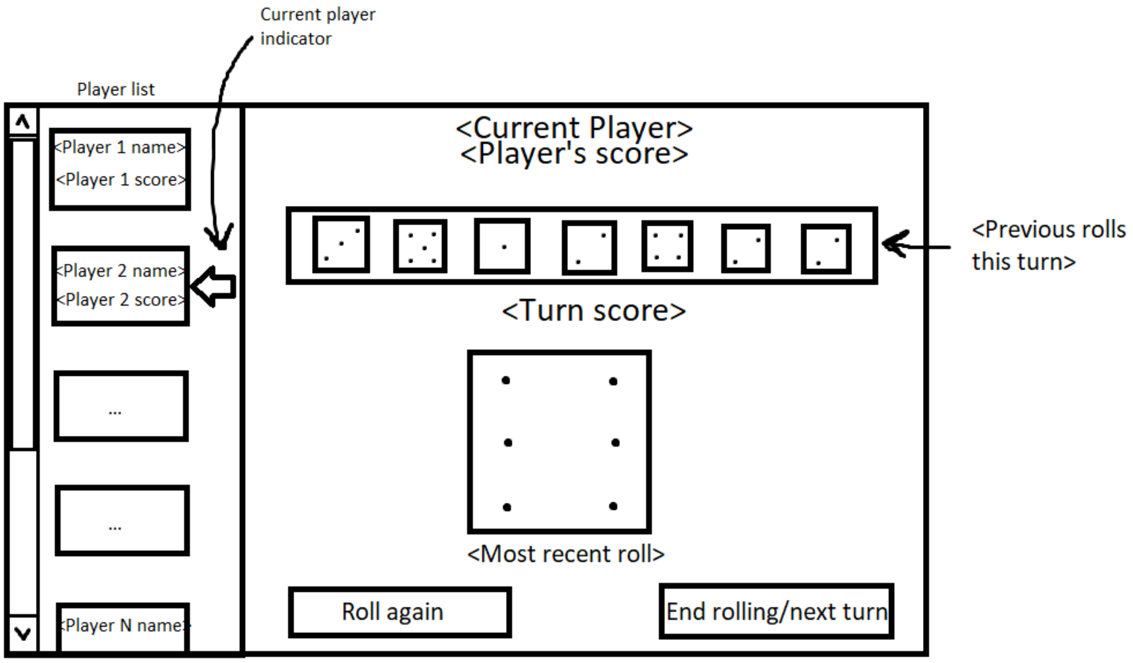
Making a pretty-looking GUI is hard work. Sure, anyone with a pencil and paper can sketch up an amazing concept, but the transition from design to reality highlights the flaws that our "perfect" ideas have. This truth was reinforced to me in my journey of bringing Bulldog (read my previous article for details) from the terminal to the application window. I have made a few simple GUIs in my time, but this was my first time getting my hands dirty with Swing. As such, much of my learning beyond surface level was done on the fly and with many visits to Stack Overflow. However, I can safely say that I am proud of my current product, despite the several bugs and additions that still need to be hammered out.

Naturally, in order to create a GUI within a program, one must have some idea of how it should look. A physical drawing is best for this, and within 10 to 15 minutes of brainstorming with another classmate, I had a decent starting point to my program:

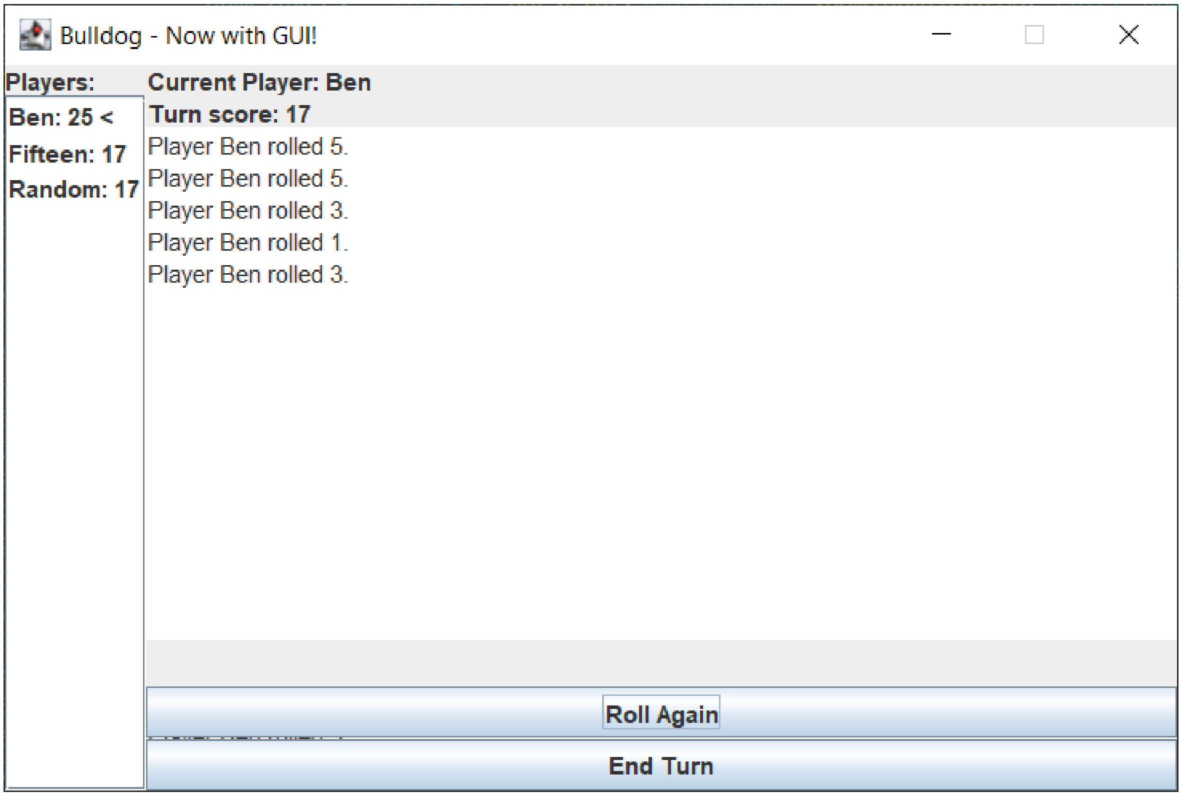


Attributions: Ben Gaudreau & Gabrielle Akers

Notice that I do not say “decent-looking”. Shortly thereafter, I made the mistake of believing this sketch of the GUI to be set in stone, and immediately moved to make a cleaner and more descriptive version of this notebook sketch in Paint:



Looking back at these images, I now realize exactly why software companies would want to avoid using computer drawing tools to make their sketches. It looks clean, but look at all these elements! There are already a lot of moving parts here, and up to this point I had not taken the time to consider my limitations. Could I even make this? Short answer: no. Here’s how my final program looks:



```
Players:      Current Player: Ben
Ben: 25 <     Turn score: 17
Fifteen: 17   Player Ben rolled 5.
Random: 17    Player Ben rolled 5.
              Player Ben rolled 3.
              Player Ben rolled 1.
              Player Ben rolled 3.
```

Let’s take a step back and see how we got here. Since my knowledge of Swing was rather poor at the onset of this assignment, it was clear that I was going to need some help. Given that I am studying the effectiveness of artificial intelligence in programming applications, my work-friend ChatGPT-4 would be the one to provide the assistance I needed. Unfortunately, OpenAI does not currently support sharing conversations that include uploaded images. Therefore, a summary of my conversations will have to suffice.

Over the course of the past week and a half, I had two conversations with ChatGPT pertaining exclusively to the Java implementation of my sketches into a GUI implementation of Bulldog. At first, I attempted to have the LLM code the vast majority of the program for me, like I had done for the first AI programming assignment. Following some comments and a conversation with my professor, David Levine, however, I realized that the complexity of this problem compared to the previous ones necessitated more active participation on my end. At a program of even this limited scope, I found it difficult to get the LLM to add, remove, or fix sections of its own code that it had generated. For example, ChatGPT constantly tried to invoke methods that it assumed existed, despite having been given all the code and therefore know that said methods did not. I assume this is a memory problem within the model itself, rather than a prompting issue, but in the end it did not matter as I could make these kinds of corrections by hand.

In total, I spent about two hours prompting ChatGPT, and another four to study, learn, and modify what it had given me in order to reach the finished product. I am not certain whether I could have completed this program sooner had I not prompted an AI tool at all, but I do believe that my initial attempt to have ChatGPT do the heavy lifting code-wise slowed down my understanding of the concepts employed in Swing and GUIs overall. Going forward, I may try generating a small, “Hello world”-style example through AI, to get a baseline understanding, before having it generate a solution to my task-specific problem.