Reviewing Mark Bunday's Documentation

1. In the first paragraph (or first section) of your design documentation, what is the most effective sentence with respect to orienting the reader to the project?

   "The purpose of this Software Design Document is to provide users instructions on how to interact with the Robot Arena Freeze Tag program but also to give them some insight into how the program works."

   The very first paragraph is in the "Purpose" section, and it's actually only one sentence long. Therefore, by default, it's the most effective sentence of that paragraph. It gives a description of what the document will be about (the program it's describing).

2. Identify a sentence in the first paragraph that needs to be reworked and state what you think is problematic about that sentence. (Do not edit it.)

   "The purpose of this Software Design Document is to provide users instructions on how to interact with the Robot Arena Freeze Tag program but also to give them some insight into how the program works."

   Again, the first paragraph is actually only one sentence long. That being said, I would suggest that this be made into two or three separate sentences. Also, maybe outlining who the document is intended for would be a good idea, since the section this sentence is in is called "Purpose".

3. Identify a sentence or two in any of the paragraphs that provides the "big picture" with respect to the software, design, or class structure, AND is accompanied by low-level details that help the reader better understand the "big picture."

   "The design of the Robot Arena Freeze Tag program encapsulates all functionality within the arena in which the player and all other arena entities, such

as Robots, Obstacles, the Recharge Station, and the Home Base interact. The arena is the mastermind behind the game.“

It’s a bit tough to find one or two sentences that satisfies this condition, but this one is pretty close. It makes clear that at a high level, Arena is in charge. At the same time, it also gives the names of some entities in the Arena, which gives some concrete detail about the structure of the program.

4. Comment on the effectiveness of this technique in the example from (3). If it is effective, analyze why you think it works here. If you think there are other details that would be more elucidating, state those.

   I believe it’s reasonably effective for someone familiar with this project, but not so much for anyone else. Throwing around words like Robot, Obstacle, etc. without describing the structure of the program may confuse people. When introducing Arena, I think it may be helpful to not list any specific entities until later.

5. Identify a topic in the writing that is either underspecified or is discussed too in-depth. If underspecified, what is the most important idea that is missing? If too in-depth, what can be removed?

   The Architectural Design section seems to mix up the description of the inheritance structure and the actual operation of Arena. I would make a seperate section describing the inheritance structure, and then talk about how Arena uses the resulting entities. Otherwise, there isn’t enough information here to figure out how Arena actually works with the Entities.

6. What do you think would be the single most impactful change to this document - in other words, what would you recommend to the author as the one area on which to focus? It could be related to the content (e.g. level of detail, more or less technical information, highlight more or fewer classes, etc.) or to the writing (e.g. reorganize paragraph or sentence order, condense text, improve sentence structure, etc.).

   I think the document tries too hard to structure itself like the example document we were given; that structure is not actually very good for the project we’re doing.

Since it is very difficult to write a full description of code this complex, my suggestion would be to write the document in a looser fashion. That is, sections that feel necessary should be added and some redundancy is ok.

7. As a programmer new to this project, which class do you think the document is emphasizing as the place to begin to engage the code? This might be explicit or implicit. What part of the writing made you think you should start with that class?

   The document is definitely emphasizing Arena as the starting point to understand the code. The System Overview section begins by saying Arena is a the mastermind behind the game.

Now explore the documentation of the classes. Go to the class that you identified in (7).

1. What do you consider to be the best and worst documented method in that class and why. OR, if you think they are all of equal quality, comment on the level of detail provided in the documentation. Is it sufficient, clear, and correct? If it is excellent, state what makes it excellent.

   The Reset() method in Arena is not documented. It's not very clear what Reset is actually doing (namely, resetting everything in a recursive process). The CheckForEntitiyOutOfBounds() method in Arena is documented nicely. It gives a brief description of the method which provides good detail but doesn't go overboard.

2. Skim through all the brief comments on the main classes page. What strikes you as you look at the collection? Is there an effective pattern in the comments? Is there something consistently lacking?

   It may sound kind of silly, but punctuation seems to be lacking throughout the doxygen comments. Since these comments actually show up in the doxygen files, they should be punctuated correctly.

Now look at the UML - be conscious of your first reaction!

1. Where did your eye go? What jumps out at you on the page? Is this an important element, thus warrants the attention? If not, offer a suggestion on how to make it less visually prominent.

   What really jumps out at me is arena_mobile_entity because there are so many arrows going in and out of it (and because the arrows are a bit jumbled). I would say this is definitely an important class, though I feel Arena is more important.

2. What did the author do in her/his UML diagram that you would like to incorporate into your UML? Why do you like that part of the UML and how does it differ from what you did?

   The author included a legend for the different arrow types. Technically this is a standard and most people should understand the meaning of those arrows, but it doesn't hurt to include that. Overall, it just makes the diagram slightly clearer.

3. Try to recall your sense of your first attempt to engage the base code, and think of how it is even more complex now. Keeping that in mind, what do you think was the most successful part of the author's writing (in doxygen and UML) with respect to helping a programmer get acclimated to the code? What do you think could be very helpful but needs some rework?

   The author's UML was complete and used proper UML conventions which definitely would help to make sense of the code. Likewise, the doxygen was comprehensive. My main criticism is of the UML; technically everything is connected correctly, but no time was spent cleaning up the arrows. As a result, it's almost impossible to actually follow the inheritance structure without already knowing it. That being said, I know from first-hand experience how difficult it is to find an arrangement where arrows don't overlap each other. In my UML, I went to great lengths to prevent overlapping and ended up with some questionable placements. It's definitely a process where trade-offs must be made.