RESPONDER: Zhongyi Sun

AUTHOR: William Batu

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 goal of this software project is to develop a rudimentary robot simulator in which robot behavior is visualized within a graphics window. "***

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.)

***"In this iteration, the robot simulator is still like a video game, but moving towards autonomous, intelligent robot behavior. " As a design document, I didn't think we needed to show such details about how we move on and construct this project. What we should do is to tell the readers, what this current project is doing and how this document helps readers get involved as soon as possible.***

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."

***Those sentences are related to the design. He used the observer pattern in this project.***

***"Whole system using observer pattern to organize information flow and control flow. "***

***"Observer is the mobile entities because sensors of arena mobile entities are the one really do the update state computation. "***

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.

***He wrote those simple sentences to explain and show what he had done, but it might be not sufficient to the new readers. It's better to explain as an***

***overview, such as what mobile entities are specifically, which sensors they have, how the job is done by each sensor, what the reaction would show up after the state computation, and so on. And then he could write the more details in the following.***

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?

***From my perspective, the component design part is underspecified. He didn't show more details about what those entities are and how those entities work. The entities are primary object in this project because the changes that we only can see are via those entities' state and those entities are good carrier for the message which is delivered from the arena to the sensors.***

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.).

***As for the content part, the level of details is needed to be improved. Such as the details in the component design part were provided in this document were underspecified. The arena entities are essential, so he may need more classes/ objects descriptions and discussion about the entity (mobile entity, immobile entity). Moreover, he could also explain to the related content in more details, not just simply using the diagrams because I think a detailed and concise document could let the new readers save a lot of time.***

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?

***Arena class. From the system overview in the writing. "In this system arena is the subject because arena can access all the information of this system.***

***Observer is the mobile entities because sensors of arena mobile entities are the one really do the update state computation. " That lets me know that the arena is the most essential part in this project because it is a subject and it has all the information which will be used by observers.***

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.

***He didn't provide the documentation in more detail in that class.The only one primary method is shown in the writing is updateEntitiesTimeStep(). Therefore, it is not sufficient description in that class to the other methods. But for this method, the logic is very clear and absolute correct, and I think it is the most excellent and valuable method in this class. This method used 4 for loops, and in each loop, it checked one event then pass this event to the current entity, so it looks like simple but it is indeed powerful and easy to understand it. This piece of code is wonderful.***

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

***When I looked at his document, I saw many small UML diagrams to show the what current part was doing. That could helper reader understand the structure of code fast. In my document, I just wrote many words without the diagrams. That was not a concise and wise way to give descriptions after I read his document and would make reader feel boring. But he didn't give enough descriptions for those classes and he also skipped many classes which were important but were not shown in this writing.***

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.

***My eyes went to the position around between arena and player, which is the central and the bigger part when I first looked at the UML diagram. And those classes are placed aside the edge jump out of my eyes. The arena and player classes are really important classes in this UML***

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?

***He added some comments, such as "pass to check collision", "including", and so on, between different classes, That helped new programmers understand the relationship, like composition, inheritance, aggregation, and the functionality of some important classes quickly. I just have one comment in my UML, so I should add some comments like him to indicate the purpose and relationship of some classes briefly too.***

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 relation (composition, inheritance, aggregation, and so on) and organization between different classes are the most helpful things. That can let a programmer know the basic structure for each component and its adjacent component, then this general overview can become to a "big picture" in his/her head. For example, the base event class has derived classes, and the sensor also has derived classes. Events are generated by arena then they are passed into entities. Then those mobile entities receive those events and pass them to their sensors by calling Accept functions. Therefore We could find that those classes were connected together in some way. Moreover, it is better to adjust the lines by changing the position of classes because it looks a little bit messy. That could mislead the attention of programmers and this is the only thing needs to be reworked.***