Project Evaluation

Conduct a customer satisfaction survey and reflect on the results.

- Was the project finished on time?
    - Yes, in fact it was done before the deadline.
- Did our product do what you wanted it to do?
    - Yes, the product met all of our requirements.
- Were you satisfied with the product?
    - The product satisfied what we set out to make.
- Communication between you and the team was satisfactory?
    - As the development team was also the customer, the communication was great!
- Were you satisfied with the quality?
    - The game ran well and looked good, so yes the quality was good.
- Were you satisfied with the reliability?
    - The game ran as expected, and the team worked hard and was reliable to complete the project.
- What was a feature you thought was really well done?
    - The animations look really good.
- What features do you wish were implemented?
    - I wish there were more mechanics in the game, like picking up items/ using items. This was never a requirement, but looking at the completed game, it would be a nice addition.
- Were your expectations met?
    - Yes.
- On a scale of 1-10 (1 being worst) would you recommend our company?
    - I'd say like a 9.
- How likely are you to do business with us again?
    - I would work with this team again to have products made.
- What would you say about us to someone who asks about us?
    - I would say that the team was able to deliver the product that was set forth and further explained in the requirements.
- Do you have any other thoughts?
    - Nope, I don't have anything more to add.

Reflect back on each document that you created and make a statement about its usefulness and accuracy.

- Process Model Justification
    - I don't feel that this document helped us much because we talked about what model we were going to use, and then we never really came back to it. It wasn't very accurate. A waterfall model was closer to the style of our model.
- Project Proposal
    - I think that this document was useful. It allowed us to brainstorm some ideas and start to flesh out a project that we wanted to accomplish. This document was useful because it allowed us to look back on what each of our roles are, our goals for the project, and a nice reminder that everyone was on board to work on it. I don't know how accurate the team roles were due to such a small team. Everything else was pretty accurate though.
- Team Member Assignments
    - I feel this document was pretty useful as it outline's each individual's assignments. While it was useful, I don't feel that we held to it very well because we had a small team and we were completely sure how long tasks would take.
- SRS Documentation
    - The requirements section in the SRS Documentation was very useful because it gave us objectives to work towards. This was important because it allowed us to see the progress being made by checking things off of a list and moving to the next item. We stayed pretty close to these requirements as well, and we only failed at one non-functional requirement.
- Software Design Documentation
    - This document was useful for starting to make a general outline of our code, but it was hard to actually tell how our code would be structured until we started actually coding it and ran into some roadblocks. The other parts of this document weren't as useful. They just seemed like fluff to help us maybe understand some topics from the course as opposed to actual documentation. The other useful section would have been the architectural design section, which helped us finalize how our project was gonna be structured. I don't know how accurate this document turned out to be due to our code naturally changing as we learned more about cocos-2dx and roadblocks we encountered.
- Team Journal
    - The team journal helped us keep track of what we were doing last time, what assignments were made, what to cover next time, etc. It was actually

a nice refresher for each meeting. As this document was just info for the team and meeting notes were kept each time, I think that it was an accurate document.

- Testing Documentation
    - This would probably be more useful if we had a client that was actually paying us to ensure quality and error free code. The problem was we were the clients and and to make the scenario up. During the building of the project we tested each time we made a change to make sure it didn't introduce any errors. However, we did test everything in this documentation.

Look closely at your SRS. List all requirements and state whether you met or didn't meet the requirement.

Functional Requirements:

- Player Control
    - We met this through keyboard input 'A' for move left, 'D' for move right, and 'spacebar' to jump.
- Graphics Rendering
    - This was done through using sprites and the Cocos2d-x rendering api.
- Gameobject Updates
    - To update objects, we were able to use event listeners, and the Cocos2d-x update/tick function.
- End Objective
    - The goal was to make it through all five levels of the game, and once you finished the 5th level there was a screen saying you won!
- Menus
    - We have a few menus within our game. We have a main menu which starts the background music and has a play button, and a level selector that allows you to choose the level you want to play.

Non-Functional Requirements:

- Programming Language (C++)
    - The product was programmed in C++.
- Game Engine (Cocos2d-x)
    - Cocos2d-x was the game engine used, and the included frameworks were used as well.
- Music

    - Starting from the Main Menu we have a song that plays while the game is going. We figured it sounded better to just use one song than to switch the song with levels or from the menu to playing the game. The song is an open source song from youtube's audio library.
- 5 Levels
    - The game does indeed have 5 levels. They were made using Tiled and the xml is parsed into the game.
- Publish on App Store
    - This was the only thing we weren't able to accomplish. We were however able to build an application for macs.

Reflect on your experience as an educational experience.

- Adam Maxwell
    - I learned a lot about the process of software development through this project. Where we planned for what the product would consist of, it allowed us to make a finished product easier because we had an end goal in mind. I enjoyed learning about a game engine and how they work compared to the simple ones I've written. The last thing that was really good about this project was that it helped develop my C++ skills.
- Bryce Mecham
    - This helped me learn some good practices about making good software, or atleast better software. I picked up some tricks to help me debug better, and the importance of writing documentation that others can use. I learned a bit about making a game, and if you have a game engine, you can actually do some pretty neat stuff. I'd say this is the most "finished" project i've had to complete here at SUU, and I thought that was very nice.
- Yucheng Long

If you were to do it over again, what would you do differently?

- Adam Maxwell
    - I think next time I would want to make a utility application and not a game. I've made a lot of games for school projects, and it would be nice to learn more about different frameworks for maybe web apps or desktop apps. Something with a frontend and backend would be a fun project that would teach me a lot.
- Bryce Mecham
    - I would probably want to do a different project next time. I think doing a project that wasn't as complex would allow me to focus on the software engineering part of the class. Which is the main point of the class. While making a project was fun, sometimes I was more worried about the project itself than learning about the practices of software engineering.
- Yucheng Long