Space Escape Post Production Notes

Our design underwent a series of iterations, each time resulting in a slight revision of the original plan. As we brainstormed new ideas and thought about various implementation problems and solutions, we were constantly in the process of bettering what we had. Throughout this process, we were also able to learn a few important lessons about software engineering and the languages we were using.

Our initial design was in general a close approximation to what we ended up with. The retro style of the game in its simplistic form was always an underlying ideal in our production. The reason for this was that we wanted to be able to focus on things besides graphics in making our game (which I consider to be a strong idea, since in true game development at least the generation of the graphics is left to an artist). This was also the reasoning in abandoning using a preexisting API to create the game, since it added complexity that we did not feel was necessary.

This can really be identified as the first true challenge; we started programming with EaselJS and quickly found that everything had to conform to the EaselJS standard. This made it very difficult to add customizable features to the game, which was really what we wanted. Also, the more we looked into the documentation, the more we got the feeling that Easel was actually slightly overpowered for our purposes. We were not using it in the ways it was intended to be used because our design was so simple. This led to the decision to break from that API and develop our own framework. The lesson here was that you really need a strong understanding of an outside resource before you decide to use it. While it looked good on the surface, we quickly found that it was not suited for our needs which led to problems quickly.

A gameplay aspect of the original design that did not come through was the intended story mode. At first, the multiplayer option was more of an additional feature, rather than the main attraction of the game. This slowly changed as the networking part of the team started doing more and more with the server side, to the point that we started focusing a lot on what we could do with the multiplayer modes. So from there, the decision was made to keep the challenge and time trial modes, and keep developing them, but to switch the focus more on fully developing meaningful multiplayer games. All in all, the team's response to this was fairly fluid. We were able to experience that in developing a large project ideas change and focuses switch, but we kept our eyes on the design specifications that we were given (in making a multiplayer game) and decided to cut part of the game to fully fulfill this requirement. Furthermore, it was not a bad thing to be ambitious at the beginning with all of the ideas we had, even though they were removed later on.

Another dynamic change we made was adding and subsequently removing the bullet feature. About halfway through the design, we found that an additional feature in the game would help make it more interesting, so we developed some simple bullets to fly at the ship. They were simply small red circles that we put in the time trial modes, and the functionality worked fine, however after experimenting with them, we decided that they were not a meaningful addition. The lesson here was another success on our part; the way we developed the bullets was to build it on the foundation we already had, and to separate it from much of the rest of the code. This worked extraordinarily well, as the only change we had to make to remove the bullets was to comment out three lines that generated the bullets. Thus any time we want to go back to that idea, we can simply add them back again and continue production.

Another difficulty was in our static screen design. The initial programming of the game assumed a constant screen width and height. What is meant by this is that no matter the size of the screen, this ship stayed the same size, as did the text. Furthermore, much of this placement of items was hardcoded in terms of pixels, not a relative value based on the screen width and height. In addition, hidden aspects such as the ship acceleration and the screen velocity in challenge mode rely on the screen width, and did such important aspects such as the scoring system. Finally, when positions were passed to multiplayer clients, the position was in terms of pixels, so players on screens of different sizes saw their opponent's ship drawn in the incorrect locations. The majority of the past two weeks of the design was focused on making the screen size change dynamically, and then fix the plethora of bugs that this update generated. This was a major design failure. In no way was this hardcoding in the generic style, and we paid the price for that.

A final challenge we experienced was in the menu design. As was explained in the presentation, our menu design went under many revisions and updates in an attempt to make it clear to the user. In the end, the confusion led to two of our team members meeting with an expert of sorts. We learned many things about menu design through that meeting, including to add more dead space and to limit the graphics that could distract the eyes. We were also able to see how to rely on people with more experience. The meeting was not an extensive use of time, but it generated more output that days of user tests could have done.

All in all, the team adapted to challenges, even where the roots cause was our fault, and learned some valuable lessons from them. In the end, I think we are all happy with the product we put out. The main lessons that we learned were: understand the languages and libraries you want to use before trying to use them, user studies are extremely helpful, just because we think the game is easy does not mean other people will think so, and to think ahead and not take the easy way out when fixing bugs.