3CG Postmortem

Marcus Ochoa

Learning Process

Working on this project was easier than expected in some areas and much more difficult than expected in other areas. I started with the functionality established in the solitaire project and worked from there. However, after working on the solitaire project I felt that much of the functionality of the game was not flexible enough for my liking. decided that the functionality of the card stacks/containers in the new game should be more generic and game state should be more easily accessible. As a result I ended up generalizing the behaviors of all card containers and storing game state in a global board instance. This was simple enough to implement and prevented excess specialized behavior from popping up, however I experienced major difficulty in deciding where these generalized behaviors should go in the large exposed and interdependent system I ended up creating. My greatest regret working on this project is not considering the dependencies of the system as much as the flexibility. That being said, implementing the general behavior of the game was much easier in this more flexible architecture than that of the solitaire project. Interfacing with the global board made changing game state much easier and calling general functions for card containers made implementing unique card behaviors quite simple. Overall I felt that I did fully consider flexibility against efficiency or modularity when starting the project, which resulted in issues down the road with messy architecture. I also feel as though I did not conduct enough tests and often did too much work at once which made debugging more difficult. However after working on solitaire I was much more comfortable in lua and LOVE2D, which made the process much smoother in general.

Project Experience

Working on this project required me to put effort into developing a more complete game architecture than most of the other projects I have worked on. On other projects I have worked on specific features or sub-architectures rather than entire game systems. The few projects I have done the entirety of programming for have been systemically simple and did not require me to pay as much attention to proper architecture and use of best programming patterns (although looking back I certainly see how these could have helped).

Highs and Lows

Overall I was most proud of the flexibility of the project's architecture. It is far from perfect, but it is functional and decently expandable. I thought it was really cool to make my own classes for UI and such, which might come as prepackaged components in an engine such as Unity. As mentioned before, my biggest annoyance or failure was simply not considering the efficiency or dependencies in my code as much as I should have. Too many classes are exposed and able to interact with each other. I should have better restricted what depends on and can interface with what. And efficiency in general fell by the wayside.

Reflection

I would like to work on another game where I have to implement slightly lower level game architecture that might not be present in the larger game engines with all the bells and whistles built in. I enjoyed the challenge and flexibility of defining my own systems. I am unsure whether I would like to continue using lua and LOVE2D mainly since I find it difficult to work in untyped languages and the lack of proper classes leaves some things to be desired. However, I highly enjoy how little bloat the engine has when compared to more mainstream options. Generally I have gained more respect for larger game architecture and am excited to work on something on this scale again, trying my best to utilize the best programming patterns and design practices.