Hung Bui

Musical Madness Report

Platform: Windows & Mac
Duration: 7 weeks
Team size: 6
My roles: Project Leader, Programmer, Artist
Software used: Unity, C#, Blender, Git

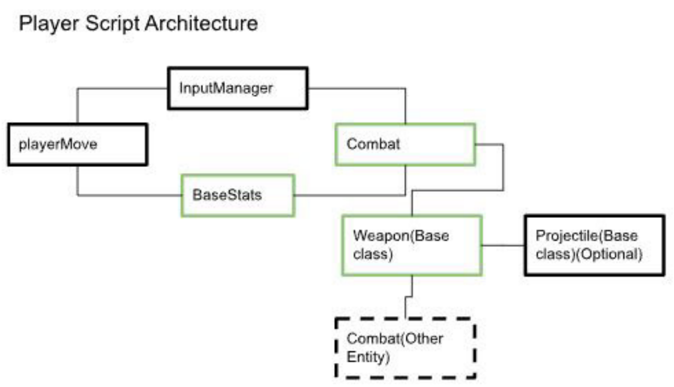
Leading the project

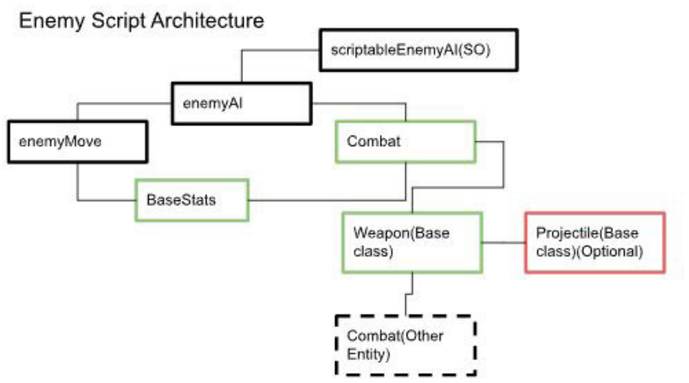
Leading my first game project at first was a bit daunting, but it felt much more natural as time went on. My co-leader and I facilitated the meetings so that each person could present their progress and gain feedback and suggestions when needed. Doing this, the design of the game was very much a collaboration, and arts, music, and programming became very cohesive. Though the meetings were not as quick as they could have been, we learned how to reduce time while also staying efficient.

Architecting scripts for Reusability

Coming from another project where code lacked architecture, it became my goal for this project to maintain structure so as to make code efficient and non-repetitive. One example of programming for reusability is that the player and enemies share the same scripts for stats, combat, and weapons, with only the "brain" differing(enemy AI vs player input). Furthermore, I was in charge of designing the enemy AI. I created a script that read from a scriptable object with parameters stating where enemies moved when they would attack, when they would pause, etc. By doing this, several scriptable object instances with different parameters could create wildly different enemy behaviors.

Unique scripts[Black](Used uniquely for one purpose)
Shared scripts[Green](Used for multiple purposes/entities)
Red scripts[Not yet made]





Implementing arts that create energetic, fit-for-purpose gameplay

For this project, I was the only visual artist and learned a lot about Unity's particle system and when to utilize that versus custom drawing animations for visual effects. Many of the player and enemy animations had to encompass what the enemies represented. For example, I at first designed the drum enemy -- meant to be harsh, abrupt, heavy-- to bounce when idol. However I realized that -- as satisfying as it was, it took away from the identity of the enemy and made it less readable.

Applying discrete mathematics knowledge

To create procedurally generated levels, we took advantage of what bit strings can do to denote the types of rooms(as there were 15 types of rooms), and to generate rooms that could be easily checked if compatible with other rooms due to bit operations. After reworking the code to use bitstrings, there was a massive reduction in the amount of code and a speed-up of the algorithm.

Conclusion

In the short time that our team worked on this project, we learned the importance of working with intention. At first, we had a large scope and spent ample time discussing different musical themes, powerups, and weapons in the game; as time went on, this discussion did not become relevant as we had to reduce the scope. We realized that, for this game, polish and quality were more of a priority than variety and quantity. We also learned that discussing and reviewing each other's progress helped us catch many bugs, and create ideas we wouldn't have otherwise.