Levi Ellis

Response: HEY! YOU! Make your indie game properly

This week I read an article by game dev Dan Marshall which touched on his solutions meant to optimize the impractical nature of common game design methods within the indie sphere. He explains that much time is wasted on things like changing or adding game mechanics unnecessarily, focusing on beautifying a graphical environment before completing its functionality, and creating time-consuming sections like cutscenes early in development. According to Marshall, the best route to create a shippable game under time constraints and with few contributors is to prioritize the completion of the game by sticking to three specific rules.

Marshall's first rule is to aim to have a playable game before you even have a single finalized asset in place. His logic is that fleshing out mechanics and implementing them successfully should be the first phase of the indie game development process. Things like assets or cut scenes are subject to change as things like mechanics and world rules are worked out and programmed, and saving asset integration for last prevents major level design revisions. I have tried to follow this advice as I have been working on my game. So far I have built a temporary environment in which I can test my mechanics, and I plan to create a much more complex environment during the next stage of development.

In his second point, Marshall does point out that graphics are the defining feature of the game's "juice and feel," and explains that although graphics should not be put in too early, they should also not be put in too late. Once core mechanics become functional is the best time to go about bringing in assets. Polishing the game is the best part of the process according to Marshall, and staging the design process so that each major component is built to be operational and then polished and finalized only once all aspects are operational is the most optimal way to work. In my test environment, I did attempt to make it look visually appealing and ultra-realistic. Although Marshall might have recommended I avoid spending any time on graphics at my current stage, I am fairly new to my engine of choice, Unreal 5, as I have only been working in it for four months. My idea was to spend about two weeks getting as close as I possibly could to having a rough but well-rounded demo to measure my capabilities against my vision and develop an achievable end goal and scope.

This ties in to Marshall's final point, which he puts simply: "know your scope, and halve it." As any game designer has experienced, he discusses the fact that games can always take more time and resources than anticipated to reach completion, and not having a strong understanding of scope and contingency can waste time and money, resulting in a delayed or scrapped game. Being new to Unreal Engine 5, I want to be sure I understand my current capabilities and base my project's scope on my quickly developed proof of concept. I do anticipate that I will become much faster with the engine and will have a greater skill ceiling by the end of thesis, but I plan to limit the scope of my project according to my current abilities in order to leave room for any unexpected issues as I go about creating a shippable, cohesive game completely on my own in less than one year.