Video Game Lessons for Business Strategy

entrepreneurship methodology

Wed Jul 15 11:22:57 -0700 2009

I see video games as a petri dish, a place where the challenges and rewards of the real world are simulated in a simpler, more discrete fashion. Often the challenges faced remind me of the same challenges I face in my work as an entrepreneur. In the spirit of Scott Berkun's Management lessons from Gears of War 2, I present you three business lessons I've learned from video games.

Acting vs. Reacting

Pax Galaxia is a casual strategy game that I found extremely fun. When played in realtime mode, it gave me some perspective on the subtle yet significant shift in attitude between acting to reacting.

When things are "under control," I'm analyzing the board, fitting the current state into my overall plan, and taking steps to advance that plan. But if I mess up - if the enemy breaks through my lines, probably in a place I wasn't watching closely - I shift into crisis mode. The plan is forgotten and all I can think of is damage control. I'm consolidating my forces, trying to cut loses, trying to re-establish a solid battle line. There's no planning for the future, just struggling with the now.

You can't win a game in crisis mode, all you can do is try to not lose. If you gain control of the situation again, you can formulate a new plan and return to acting on that plan instead of doing nothing but reacting to the biggest fire.

Sound familiar? Companies go through this same shift, periods of acting and reacting. I don't think reacting can be avoided completely if you're aggressive in seeking growth and success. But if you find yourself spending long periods of time in reaction mode, you're in trouble. You don't win by reacting.

Planning For An Uncertain Future

Puzzle games often revolve around a match-similar-items game mechanic: for example, gems of a certain color. Puzzle Fighter is a classic in this genre. My strategy in these types games is to build almost, but not complete, groups of matched blocks. I'm trying to set myself up for future large combo chains, by having all the pieces poised and ready to go.

Planning for the future makes perfect sense, of course, but what makes this interesting is that you don't know what pieces you're going to get. So you have to combine your knowledge of what pieces are likely to come later with what you currently have; then take an educated guess as to how to arrange the board. You're under time pressure, so you can't do a full statistical analysis for every move, only rough guesses in order to make your decision quickly and move on.

Sound familiar? Companies, particularly young startups, are doing this very thing every day. You don't know what opportunities or challenges are coming next in the marketplace, but you know what you have now, and you can try to align them such that you'll be poised to take advantage of the somewhat guessable future.

Experimental Validation of Theories

"Platformers" is the genre title used for exploration games like Super Mario Brothers, Jax and Daxter, Prince of Persia, or Rachet and Clank. In these games you encounter new environments with combinations of enemies, surfaces to climb or jump on, things to ride, etc. You explore the environment and form a theory about how to solve the puzzle presented by the current location, with the goal always being to progress to the next location.

There's two parts to this: forming a theory on how to overcome a certain obstacle, and execution of that theory. You have to get both parts right, but if your plan isn't working, it's hard to tell whether it's your theory that is wrong, or your execution.

You could have the right plan in mind, but you're failing to execute it well - not timing your jump right, or waiting for the enemy to be in the right place. Or, your plan could just be wrong - no matter how well you execute it, that ledge is out of reach for the jump you're trying to make. But when you're failing, there's no way to be certain whether it's your theory or your execution that is at fault. Whether to give up and try something else, or keep trying to perfect your implementation, is a question of judgement.

Sound familiar? This is the deepest question for a company building a product. When failure happens (and nothing succeeds without many large and small failures along the way), is it because you're not building a solution that fits a true market need? Or are you barking up the wrong tree completely, and no matter how good a solution you build, it's solving a problem that no one really cares about? There's no way to prove it one way or another, except by succeeding.