What it Takes to Make a Game

Part 2: How do you Start?

I'm glad you asked, Conner. The answer is deceptively simple:

You make a game.

That's it. There's no different angle, no clever solution, no trickety-trick that's gonna make you a game designer. You just have to *make games*.

It's hard work, and like ***every single artistic endeavor***, you will suck at it when you start. But sucking at something is the first step toward being sorta good at something (I'll just shotgun cartoon references at you, if you don't mind, Patricia).

As I covered in Part 1, the most difficult part of the process is never the vague concept. "It would be cool if..." is the easiest part of the process by a huge margin. So what should you do after you've come up with the idea? I also alluded to that in Part 1, but in this word-vomit, we'll dive deep into the answer. And for this topic, we'll pretend that you're 100% a solo dev who has to do everything on your own (which means having a bit of a rigid structure to avoid hopping all over the place and then getting lost in the weeds).

The next step is assigning discrete mechanics for each thing you want the game to do. This is "easy," in that anyone can do it with sufficient theoretical knowledge. If you've ever played a game, you can identify most of the things you want your game to do. You'll miss some, but that's fine for now – you'll catch those gaps later (trust me, for the first few iterations, every designer has holes in almost all of their games as a solo dev).

It's not just about imagining the main interactions, you have to figure out how literally every single part of the game works with literally every *other* part. Naturally, you start small. Both mechanically *and* as an overall design. You start with a "character;" a controllable element. Then you make that character move in some/all directions. Then you make that character interact with something. Then you make that interaction matter in some way. That's the basis of most games and it's the most important thing to do when you're starting out.

"That sounds boring, I wanna make a *good* game!" That's a good point, but shut up, Sven.

If you want to make a good game, you have to make some bad games first. If you try to make a good game out of the gate, you may actually have a good idea and some good mechanics in mind, but because your technical skills aren't there yet, you'll have a mixture of cool ideas and solid effort that probably comes out looking and playing like garbage. That's when we get disheartened as fledgling designers: when we pour our heart into something and it ends up being *awful*. It's even worse if the execution is bad enough that we can't salvage anything or save the game, and it all has to get thrown out. So the first step is to make games that you *know* will be bad. And you have to be okay with that. Then you'll make another and it'll be bad too. It still hurts, but less!

But don't worry – they won't be bad forever. And when you're good enough to try your hand at something you might sell or leverage for a job or pitch to a publisher, you approach the project with a slightly different first step. Eventually, your goal when making a game isn't starting with basic functionality; it's starting with one main mechanic that answers the question, "why am I making this game? What does it do differently that justifies its own existence?" And then building your game outward from that one mechanic. Everything else in the game will usually be informed by how that single mechanic works. It's your hook; it's the thing that makes someone look at it for five seconds in the Steam shop and say, "oh, that actually looks dope."

This is where things get hard. You have to design the elements needed to get the first functional test. Just something to make sure it works. Next, you have to actually *test* the game, which sounds easy, but it's secretly really hard for three huge reasons. The first is psychological – you have to be able to distance yourself emotionally from your own creation and be able to assess it on its own merits. The second is analytical. You have to be able to assess some critical things quickly and firmly ***without*** losing faith or momentum in your process (yes, even if the game is bad, Eric). And the third is practical – you have to be able to make the thing that you can then test. For programming, that takes a lot of time and work! I learned that process with card games and board games, which are much faster to design and test, since there's no technical production work. But once you're ready to test, here are the core things you'll be looking for. They'll be asked sequentially as you complete and test more of the game over time:

1. Does the game do anything that other games don't do?
2. Does the central mechanic have legs? Can you turn it into a solid gameplay loop; **[***See Part 3 – a deep dive on Gameplay Loops!]**
3. Does the game control well enough?
4. Is it fun/does it feel like it *will* be fun as it starts coming together?
5. What could be cut without demonstrably dropping the quality?
    a. What could be cut to *improve* the quality?
6. Which mechanics feel poorly balanced compared against each other?
7. Does it take the desired emotional, physical, and temporal investment to play?

Each of those steps, you also have to be able to analyze ***why*** something isn't working, ***what*** you can do to fix it, ***how*** to fix it, and then you actually have to follow through with that and start back over until every light so far is green.

And then you have to do it again. And again. Then you have to put it in front of people who don't know its design in and out so you can figure out some *more* stuff:

1. Does the average player understand it easily or intuitively?
2. Does the average player need additional guidance for how any mechanic or interaction works?
3. Does the average player have fun?
4. Does it *still* take the desired emotional, physical, and temporal investment to play with the average player?

I bet you're exhausted, Jessica. But I have fun news: now you start over from the beginning and start iterating on each of those things, even if they're functional. Over and over. You want your game to be ***crisp***. And this'll take time. A lot of time. ***A lot of time***.

“What am I even looking for when I’m testing? How do I know when something’s good enough?”

And here we finally come to the secret sauce. The thing about mechanical design that grinds a lot of games to a halt or makes them come out as bad games that the designer doesn't know how to fix:

Most of good gameplay design can be boiled down to numbers. ***You***, Doug, have to be able to identify, translate, and fiddle with those numbers at a ***very*** granular level. This is what makes a game *feel* good in a hard-to-explain way. Let's take an example of a super well-rated game and chunk it down to nothing but numbers. Slay the Spire:

1. How many characters are there
2. How many starting resources does the player have (cards in deck)
3. How many starting resources are unique (to that character)
4. How many variable starting game state options are there (characters, starting buffs from Neow, difficulty options, alternate game modes)
5. How many different acquirable elements or resources are there (gold, cards, potions, artifacts)
6. How many options does the player have after starting during each micro-loop (how many map nodes there are each step)
7. How many different node types are there overall
8. How many of each node type is there per act (yes, this can't be totally random or else the game *can* feel awful sometimes)
    1. What’s the maximum
    2. What’s the minimum
9. What’s the average ratio of value for each player stat (buffs, debuffs, attack, defense, and HP)
    1. Repeat for enemy stats
10. What's the average ratio of enemy stats to player stats(this value has to be re-examined every act) – This will need revised if certain stats are more or less valuable depending on the player’s starting state
    1. Repeat the process for mini bosses (this also has to account for *some* unspecified, not-necessarily-predictable amount of resource acquisition)
    2. Repeat again for bosses (this also has to account for some unspecified, not-necessarily-predictable amount of resource acquisition with greater variance than mini bosses)
11. What's the average resource to cost ratio in combat (mostly for cards, but as you get deep enough in design, you’ll realize that stats like HP are also a resource)
12. What's the rate/percent chance of resource acquisition (for gold, potions, cards, and artifacts, respectively)
13. What's the average value of a single acquired resource (for gold, potions, cards, and artifacts, respectively)
    1. What's the maximum value
    2. What's the minimum value
14. What’s the average value of a normal fight
15. What's the average value of each '?' tile
    1. What's the maximum
    2. What’s the minimum
    3. What’s the ratio of positive returns vs negative returns
16. What's the average value of a rest stop
17. What's the average value of a resource acquired from a mini boss
18. Repeat for boss

19. What's the ratio of resource value to difficulty (some cards, potions, and artifacts are way better the higher your ascension level)
20. How long is an act
21. How many acts are there
22. What's the scaling value between acts (not just the aforementioned ratio between enemy stats and player stats, but also the rate of resource acquisition, the value of resource acquisition, the ratio of resource to cost, etc)
23. How much variance is there between runs that does not fall into the scope of starting states
24. What's the rate of meta-progression
25. What's the ratio of starting meta-progression vs meta-progression to be completed
26. What's the ceiling on difficulty

Almost literally every single mechanic in a game can be boiled down to numbers, and numbers are ***not*** the thing most people choose to get good at. Most people do not find numbers *intuitive*. Your job as a gameplay designer is to understand those numbers and intuitively recognize when they're not balanced against the other numbers ***or*** against your overall design goal. That's not easy! It's an acquired skill like any other, but the main way to acquire it is to play a lotta games and start paying attention to all the things that can be broken into numbers and then compared against each other. Start seeing those connections the designer made.

As an added benefit of understanding the numbers for design and balance reasons, nearly all of the programmatic parts of game design are ***also*** laden with numbers, so you're actually killing one and a half birds with that stone, Shirley. And while I don't condone avicide, I *do* condone efficiency.

"Okay, you exhausting windbag, I've spent months making and remaking and tuning my game. It actually feels pretty good, so what now?"

Well, A. That's hurtful, Jeremy. I expected better of you. But B. Now you do ***all the other stuff!*** *Hooray*! Writing, art, sound design, music, marketing, production, and business!

Now hold on a second, Josie, lower the gun, I know I just hoisted a lot on you, but it's not all that bad.

As a solo dev, you need to identify which of those things you *are* good at and which ones you're *not*. For everything you're good at – great! Hopefully our guides will help you understand the way to get started building and integrating those elements into your nice, new game. For everything you're *not* good at, there's a lot of design resources out there for us. Most of it is pretty cheap or even *free*. Check out our links section, which will have a little page for resources from every major field of game design! **[RIGHT, BRIAN? ಠ_ಠ]**

For marketing, production, and business, you may, unfortunately, have to try to find a publisher who's into your game and wants to do that work for you in exchange for some of the moolah. And once you get into trying to find a publisher, there's a whole new efficiency element in play. Because game design ***isn't*** just a solo act, a lot of work needs to happen at the same time across teams. As a result, it can be very efficient to develop a presentable part of your game as quickly and as polished as possible so producers/marketers have something to work with. This is called a "vertical slice," and it's crazy important once you're actually trying to make a living like this. It also serves as a stable "home base," where all of your teams can start from. But that's a topic for another day… Wesleeeey.