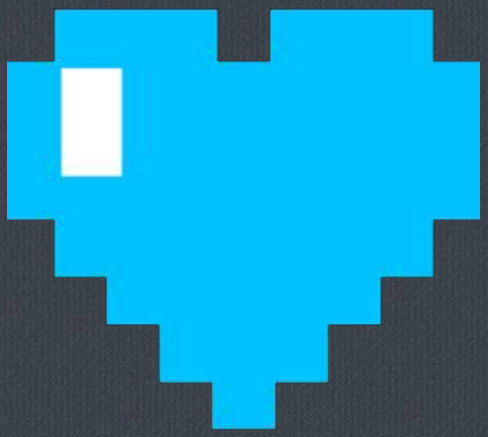
Videogame Developer's Strategy Guide



By Chris DeLeon

Edited by Laura Schluckebier

CHAPTERS OVERVIEW

Why a Strategy Guide?

Learning to develop videogames is itself an open-world adventure. Explore in any direction you'd like. Develop character skills that are broad, or specialized. Build a party, or go solo.

While this book can be read cover-to-cover - I've sequenced these chapters and sections to help that go smoothly - for most people it will work much better if used as a strategy guide.

If, in playing a game, you ran into trouble in the Lava World, or wished to learn about Enchanted Crafting, you'd head to that section of a guide. Similarly, here, if you're tinkering on a game type that has levels, jump to the Level Creation chapter for terms and processes. If you're in a group, browse some sections on Team Projects.

There's also one more way that this is like a strategy guide: tips in a guide only help if you're actually playing the game it's for. This guide is for videogame development. For these ideas to have context and purpose, you really should be actively making videogames of your own.

If you haven't yet started creating games: today's a great day to finally begin your journey!

CHAPTER 1

Getting Started

CHAPTER 2

Questions About Education

CHAPTER 3

Programming

CHAPTER 4

Get Motivated

CHAPTER 5

Game Design

CHAPTER 6

Level Creation

CHAPTER 7

Team Projects

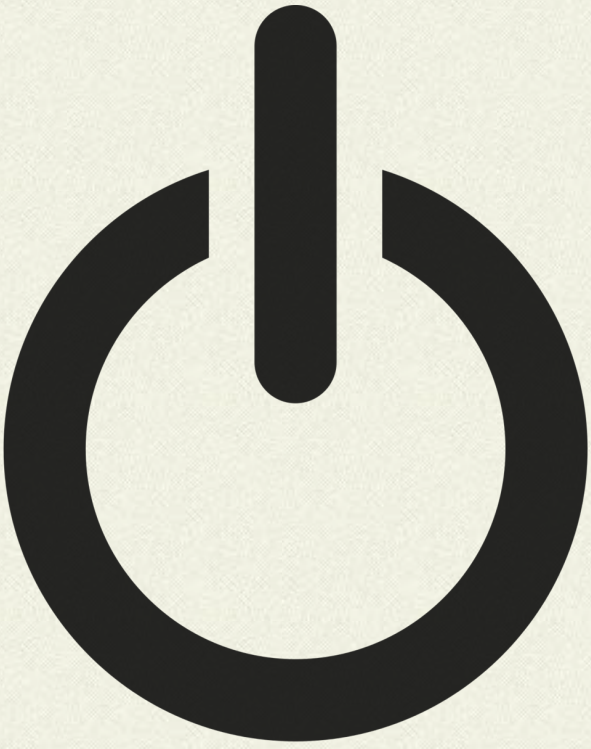
CHAPTER 8

Industry

CHAPTER 9

Game Analysis

1



If this is the first material that you're reading about videogame development: welcome! The way forward is long, difficult, at times frustrating... and totally worth it. You're on your way to learning how to create your own videogames! There's nothing better than homemade fun.

Getting Started

SECTION 1

MAKING YOUR OWN VIDEOGAMES AT HOME IS TOTALLY AWESOME

There has never been a better time than now to design and develop your own videogames. A typical home computer is all that's required. The ways to learn and share your games are already huge – and growing!

The original *Pac-Man* took a team of nine professional engineers a year to create. Although we tend to experience *Pac-Man* today as mere software, either downloaded or played in a web browser, the original was a 400 lbs. (180 kg.) painted wooden arcade cabinet of custom electronics that needed to be shipped all over the world.

Pac-Man's development and distribution required a level of labor and equipment in 1979 that could not have been achieved as a part-time side project.

Now in 2014 virtually any decently experienced amateur can remake a videogame similar to *Pac-Man*, working alone, mostly in an evening. It doesn't even cost anything to make.

Then you can distribute the game worldwide instantly for free.

What at one time revolutionized the game industry is now less complex than projects that we see made in weekend game jams.

Things got better. Much better.

BETTER FOR ALL OTHER GAME MAKERS, TOO

Advancements in development and distribution options means that we're seeing more kinds of games, released more frequently, from a larger number of creators.

This is very exciting, for players and game makers alike. But it's essential to recognize that this does not mean that making games *as a business* got easier, too. It's true that it's easier to sell an app on a smartphone than it was to sell a game for the old Nintendo Entertainment System, but that truth applies to every developer in the world. As of August 2013 there were 3 times as many new mobile apps published *per day* (~2400; VentureBeat) than the total number of unique NES games released *ever* (~800; Wikipedia).

The barriers to entry to making games aren't just lower for you. They are lower for millions of other people making games, too. *If your main goal is to make indie games professionally, know that you will be competing against more people than ever to do so*.

I'm not saying this to discourage anyone. I'm saying it to establish perspective of what to prepare for.

DANGEROUSLY MISLEADING MYTHS

There are some misleading stories out there which have tricked people into thinking that making videogames is a get-rich-quick scheme, or that quality and hard work aren't necessary to succeed.

The first misleading myth is that of the overnight millionaires. Some indies have fared very well. They are talented, and they put in the work, but their success appears to be overnight because you usually haven't heard of the first 30-50 games they've worked on, or seen what went into developing their hit for years leading up to launch. Most indie developers make very little money from their games, and many do paid contract work on the side to make ends meet.

The second misleading myth is that simple games that anyone

can make easily often explode into meme-like fame. The few games like Flappy Bird are the exception, not the rule. For every tiny, simple game you see that becomes a hit, tens of thousands of little games quietly flopped.

Tossing tiny, unsophisticated games into digital stores just in case one takes off is like buying time-consuming lottery tickets.

The vast majority of successful games were not simple projects or lucky breaks, but the result of hard work, talent, and experience. Nor does every game that arises from hard work, talent, and experience become a hit. No one gets any guaranteed profits or benefits from effort or seniority. Every game is a new, separate effort to excite and engage demanding audiences.

To make games professionally your work and skills need to be extraordinary. It'll require practice, persistence, experimentation, compromises, and humility along the way.

MOST CUSTOMERS AREN'T GOOD TEACHERS

The marketplace can be a brutal, risky, and unforgiving environment, which doesn't make it very friendly to beginner learning. People that insist on releasing their first games commercially just to see, and "maybe earn a little on the side" are seeing discouragement from dozens of dollars or less for months of their work. Would we really expect otherwise though, if someone just learning guitar sold their practice attempts on iTunes?

For this reason I advise people who are in it for the long-haul to at least begin by making games non-commercially. What can be done now without funding is incredible. It's a great way to start without betting the farm, nor mistaking a lack of paying customer response as a reliable gauge on whether you're learning and progressing.

Like a person going to Hollywood aspiring to be a movie star, if the goal is to earn a living making games as an independent developer, it would be prudent to

have a backup plan, a day job, and an understanding that it'll be a long, bumpy road to success.

BUT IF WHAT YOU WANT IS TO MAKE GAMES

If your near-term goal is to make games, rather than to make games as a business, then this is without a doubt the best, easiest, most exciting time yet to be doing it.

You have so many options of ways to learn, ways to create, and ways to share. I've actually heard this major benefit turned around and used by someone as an excuse: there are so many ways to make games now that they "can't figure out the best way to start." Absurd! There is no one best way to start, nor one best way to make games.

Anything that gets you making games and trying out your ideas is great. Within this book I'm going to give you the very best advice and ideas that I have to offer, but part of my advice is this: learn from any and every source that you can. I'm but one of many people who have been making games and is excited to help others. Not sure whether something is worth learning? Just learn it. Then you'll know if it was.

Along your way you will learn and use many different programming languages, tools, and changing development environments. Don't get stubbornly stuck on whichever you use first, and then it won't make much difference which it is.

Know too that if you want to make videogames, learning can't just be book smarts. Your attention has also be on *doing* a practical skill. Reading can be helpful, a good use of time, but reading alone is not enough, for the same reason that someone won't become a great painter without a canvas.

GET LOST IN FOLLOWING YOUR INTERESTS

If you wish to establish an identity of your own as a developer you can't get caught up in chasing trends. Find and develop your own voice by following your passion.

"Not amusement nor distraction, but the desire to effect some cherished purpose is the strongest motive that can move the learner."

-John William Adamson

In order to create your own little worlds, you've got to be ready to live a bit in your own little world.

Well before stressing about what the public thinks, perhaps focus first and foremost on impressing and surprising yourself with just what you can accomplish.

Pull yourself from distractions, zero in on developing the skills that you wish you have. No one can create your many games the way you want them but you.

Your games will exist. Make up your mind to figure out whatever you must as you go to make it so. Accept that your games existing in an imperfect way is better than them not existing at all. (Anything that ever exists is imperfect!)

ENJOY THE FREEDOM

We're so fortunate to be alive at a time when you no longer need anyone else's money or approval to make your game a reality, nor to get your game in front of other players around the world. Making an enjoyable videogame no longer requires special gear or privileged access to retail distribution. You can learn to do it in your free time, free from external pressures, and free to share however you wish.

You can bring new games and ideas into existence simply because you want them to exist.

If there turns out to be a smaller audience for it – whether a circle of your friends, a dozen people across the globe, or even just you as the creator – that doesn't much matter. A game developed for free doesn't have to sell 100,000+ copies just to justify its existence.

As developers we've never had this level of freedom.

It's a great time to be making videogames.

Pages Removed for the Free Preview

4

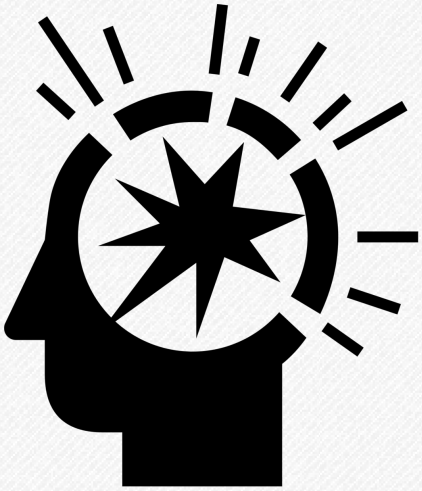
Even once you know what to do, you've still got to put it into practice before that knowledge can do you any good. Becoming good at anything requires practice. Let's look at some ways to start getting practice sooner rather than later.

Get Motivated

SECTION 1

STOP TRYING TO LEARN EVERYTHING BEFORE STARTING

Knowing about something isn't the same as knowing how to do it. The former comes from reading and discussion, the latter requires practice and experience. When you know enough to start... start!



Don't wait until you know everything there is to know before starting.

Learn just enough to get started. That includes having a rough idea of a realistic scope for a first project, so that you don't wind up lost on a fool's errand.

Then get started. Get your development environment set up and compiling empty, test, placeholder, or example code. Just get anything on the screen that runs, meaning an .exe if you're programming for Windows, a .swf if you're making a web game in ActionScript 3, etc.

At this point, you're already ahead of an untold number of people that have only ever thought about videogame development, but have been too caught up in waiting for the perfect idea, or trying hopelessly to fill their brains with everything ever written and said about it before actually getting started. So far so good.

Find a way in your programming language or environment to load an image file and get it on the screen, to respond to keyboard and mouse or whatever input is needed, to load and play back sound effect files and looping music. Depending on the platform, picking a library may be helpful for this (ex. SFML, SDL, Allegro, or XNA, if in C/C++), or this may be functionality built into what you're working with (Unity). Often the easiest way to get to this point is to just find some simple example code that already does these things, then twiddle settings in your development environment until you can compile and run it as expected.

Does the player character need to move like a truck? A spaceship? A tank? Mario? Get the input to move the player's graphic (probably an unanimated rough draft of the graphic at this point, concerned only with basic appearance and scale on screen) moving in the way that it ought to move.

Get the level structure put together to position visuals and handle basic collision against the player. Save/load the level structure in some practical format – never mind getting caught up on the theoretically optimal compression of that data, as level files are virtually always tiny, and if that becomes a problem later, cross that bridge when you get there. Level files initially saved in ASCII can make debugging significantly easier anyhow. Is the game world based on freestanding obstacles? Grid-based tile collision? If the camera needs to pan, add some offsets to the player draw position and level draw origin to achieve scrolling, or a global transform if you're in 3D or using hardware acceleration, and update those offsets to move the view based on player position.

Add enemies (if needed), items (if needed), trigger puzzle elements (if needed), special powers (if

needed), ammo limits (if needed). Add nothing that isn't needed. If you're not sure what else the basic engine might need – and I'm only using the word "engine" here only in the most minimal sense, meaning the game's core code, not the try-to-support-every-game-imaginable Titanic-seeking-an-iceberg undertaking – start putting together playable level content and see what additional features or process improvements you find yourself wanting while doing that.

The game may not need anything else.

If and when a situation arises for which you're not sure how to proceed without digging and experimenting for solutions, that's fantastic, welcome to real programming.

If and when a situation arises for which you have to make a tradeoff between burning an uncertain amount of time on figuring out something tricky, or working out how to cut that feature from the design without the game completely falling apart, that's super cool, and welcome to production.

Once you finish the game and people don't feel about it the way you hoped they would, first off: that's awesome, because hey, you finished a videogame. Congratulations! Next time you'll be positioned to make incrementally more informed tradeoffs with consideration for implementation realities, production compromises, and potential impact on user experience. It's only at this point that someone is beginning to really do videogame design, as opposed to talking about videogames, reading about programming, or cloning someone else's work as an exercise. Note too that design of a full videogame is different than level design. Though the two are sometimes conflated, since they are often done by the same people, when

done in isolation from other issues and options about a game's development, level design is another field of content creation, another skilled technician's craft like animation, dialog writing, or sound editing. Those are hard things to do, deserving of respect and worth doing well, but they are not videogame design. Designing a videogame is different than designing for a videogame.

Learning is great. Books are important. Some amount of learning does need to happen before starting, but the amount of learning needed tends to be far less than people seem to assume. Learning, for this type of material, has to be situated in a context, demonstrable, useable, and practical. Contrary to the lay consumer impression that making something is a mere variation on being able to judge something, a maker has a fundamentally different set of concerns and practices than a critic.

Roger Ebert was brilliant, but wasn't suited to making a decent film. (Not hypothetical - see: *Beyond the Valley of the Dolls*.)

If your goal is to be a critic: spend all day studying the form and discussing it. If your goal is to be a maker: start making things, keep making things, and finish making things. Being reflective on practice can be helpful, but that requires actual practice to be reflective about.

Learning everything there is to know about programming or videogame development before starting is impossible. It simply won't – can't – all fit inside one head at the same time. Even if that information somehow could be learned, memorized in the abstract, detached, and rote sense, it would not be of any use without experience working through real problems with it.

This is not a decision between learning versus doing. What I am proposing is that, within this

context, learning without doing isn't really learning. Meanwhile doing without learning is impossible, because doing will demand learning.

If you make some wrong assumptions early on, they're often easily corrected. That's the nature of digital stuff. We're not laying railroads or building skyscrapers, we're not cutting someone open while they're under anesthetic – we are dealing in digital text and other easily modified, easily backed-up files. The rework time is nothing compared to the vast but invisible damage from never starting, waiting an infinite amount of time until everything knowable is known. That rework is even when the best learning happens, because the desire to not lose that amount time on the next attempt will help lead to a deeper understanding of what caused it, so that it might later be avoided.

Be wary of excessive preparation serving as a disguise for procrastination.

Pages Removed for the Free Preview

IMAGE CREDITS

Cover and Chapter Icons

Cover Heart designed by Raji Purcell from The Noun Project

Ch.1 Power by Jardson A. from The Noun Project

Ch. 2 Teacher by Juan Pablo Bravo from The Noun Project

Ch. 3 Architect by Augusto Zamperlini from The Noun Project

Ch. 4 Tips by Lemon Liu from The Noun Project

Ch. 5 Gears by Edward Boatman from The Noun Project

Ch. 6 Mouse tools icon image in public domain

Ch. 7 Talking by Juan Pablo Bravo from The Noun Project

Ch. 8 Business Connection by Francisca Arévalo from The Noun Project

Ch. 9 Magnifying Glass by Yazmin Alanis from The Noun Project

Page Background Textures

Page texture from subtlepatterns.com
CC BY-SA 3.0 - Subtle Patterns © Atle Mo.
Chapter background pattern made by Olivier Pineda.

Getting Started

Making Your Own Videogames at Home is Totally Awesome - Working At Home by Rutmer Zijlstra from The Noun Project

How Long Does it Take to Learn Game Programming? - Calendar by Phil Goodwin from The Noun Project

Hobby Game Development: 20 Questions - Quiz by Rémy Médard from The Noun Project

Beginners Shouldn't Start with a Design Document - Document by Piotrek Chuchla from The Noun Project

Clone Videogames to Learn Real-Time Videogame Design - Space Invader by SuperAtic LABS from The Noun Project

General Concepts for Beginning Developers - Sketchbook by Edward Boatman from The Noun Project

Learn Videogame Development Like Woodworking - Tools by Dmitry Baranovskiy from The Noun Project

Fan Habits and Focus are Not Developer Habits and Focus - Crowd-Surfing by Hum from The Noun Project

Questions About Education

Advice to a New Student in Videogame Design - Education by Berkay Sargın from The Noun Project

Questions from an Elementary School Class - Students by Piotrek Chuchla from The Noun Project

Class Questions About Game Development Career - Image in public domain

Math for Videogame Making (Or: Will I Use Calculus?) - 3D by Michael Senkow from The Noun Project

Question About Comp Sci and Game Development - Network by Bruno Castro from The Noun Project

The Ways of Self Education - Education by Pete Fecteau from The Noun Project

Programming

Game Programming Fundamentals - Utility Knife by Hayden Kerrisk from The Noun Project

How Programmers Program - Code by Nikhil Dev from The Noun Project

Position and Speed Variables - Directions by Pedro Ramalho from The Noun Project

Float and Int Variables: Casting and Other Issues - Image in public domain

Hack then Refactor - Arrows by Juan Pablo Bravo from The Noun Project

Basic Real-Time Videogame Artificial Intelligence - Robot by Edward Boatman from The Noun Project

Quick and Dirty Backups - Image in public domain

Steps in Programming a Simple Real-Time Strategy Game - Image in public domain

Get Motivated

Stop Trying to Learn Everything Before Getting Started - Mind Blowing by Luis Prado from The Noun Project

"Overcomplicating Everything" - Confusion by Kelcey Benne from The Noun Project

Think by Building, Build to Answer Questions - User by Wilson Joseph from The Noun Project

Your Attitude Matters Even When Working Alone - User by Mundo from The Noun Project

Don't Wait for an Event, Job, Contest or Assignment - Waiting Room by Luis Prado from The Noun Project

The Brain is Not an Emulator - Thinking by Dirk Rowe from The Noun Project

Stop Arguing About What Makes a Better Game - Argument by Michael Thompson from The Noun Project

Start Before You Have an Idea - Image in public domain

A Little Planning Can Go a Long Way - Gantt Chart by Jeremy Boatman from The Noun Project

Game Design

Modest First Projects and Incremental Learning - Process by Joel McKinney from The Noun Project

Bottom-Up vs Top-Down Design - Create Database by Ilsur Aptukov from The Noun Project

Photographer's Algorithm - Camera by Okan Benn from The Noun Project

Doing More With Less: Short Videogame Design - Image in public domain

Colorful Oceans and Chunk Sauces - Noodles by Anna Wojtowicz from The Noun Project

Genres and Conventions: Known Patterns of What Works - Image in public domain

Level Creation

Anti-Design / Backwards Game Design in Goldeneye - Gun by Simon Child from The Noun Project

Level Design Concepts - Architect by Joel Burke from The Noun Project

Level Design Process - Mindmap by Leslie Tom from The Noun Project

Level Design Q & A - Image in public domain

Non-Essential Level Art is Essential - Graphic Design by Anna Weiss from The Noun Project

Visual Language in Super Mario Bros Level Endings - Flag by Ashley van Dyck from The Noun Project

Team Projects

Videogame Project Management for Hobbyists and Students - Meeting by Ainsley Wagoner from The Noun Project

Communication is a Game Development Skill, Part 1 - Image in public domain

Communication is a Game Development Skill, Part 2 - Image in public domain

Establishing a Videogame Development Club - Image in public domain

Industry

A Frank Look at Making Videogames Professionally - Image in public domain

Indie Game Development as Career - Art Collector by Piotrek Chuchla from The Noun Project

Influence of Business Models on Game Design - Pay Per Click by Siwat Vatatiyaporn from The Noun Project

What's Japan Doing Differently? - Japan by James Christopher from The Noun Project

Intellectual Property and Hobby Game Development - Copyright by Thomas Hirter from The Noun Project

Should I Release a Game as Soon as It's Done? - Checklist by Aaron Dodson from The Noun Project

How to Get the Most Out of Your First GDC - Conference by Wilson Joseph from The Noun Project

Game Analysis

Why Minecraft Worked in 2011 - Cubes by José Manuel de Laá from The Noun Project

Reflections on 3 Specific Hobby Game Projects - Thinking by Michael V. Suriano from The Noun Project

Quit Smoking: Effect of Interaction on Interpretation - No Smoking by Rahul Anand from The Noun Project