CREATING A GREAT DESIGN DOCUMENT

I've got to get product out. In the panic and dizziness, my head smashes against the CRT and next thing I know this genie whiffs up out of a virtual bronze-texture-mapped lamp and offers me three wishes. Without missing a beat, I answer, "I need...

- A great team of talented, skilled, and dedicated engineers and artists (including a very understanding wife) with strong interpersonal skills.
- Enough time and money to allow for a mess-up or two.
- A first class design document.

Once upon a time, when coding a game involved one programmer (and *maybe* an artist) with a take-it-as-you-go budget and a loose deadline, documentation didn't need to be taken so seriously. You knew

BY TZVI FREEMAN

what you wanted to make and you made it. If there were a few major changes along the way, the only one to complain was you. Nowadays, a thorough and readable document can mean the difference between a swift descent to budgetless Hell and a smooth ride to shrinked-wrapped Nirvana.

How the System Works

Most games go through three development stages, from concept to design to production. Think of them as "flash," "paper," and "grind."

In the first stage, the concept paper acts both as a letter to yourself — setting out your goals clearly so you won't lose sight of them — and as a sales tool for whomever takes the product to market down the road. Sometimes, this stage involves a working mini-prototype as well, which gives you a chance to experiment and revise your ideas.

The intermediate stage of design involves a lot of discussions with artists, animators, musicians, and engineers — trying things out, and finding ways to organize and set down your ideas.

In the final stage, production management is often left up to some expert in moving trains and tracks without major collisions. The original designer may be an integral part of the team, but in many cases — especially in large companies — the designer ends up as a kind of outside consultant.

Without question, the design document is where the original parent of the project exercises the most influence on how this little baby is going to grow up. Even if you, the designer, have decided to double as project manager, don't delude yourself into thinking that you hold all the reins. A complex project involves many talented people. Skilled programmers and artists tend to have minds of their own. While you intend to create a horse, the artist may be envisioning a unicorn and the programmer a highly efficient camel. A good document ensures that you are all planning to make the same thing. A *great* document ensures you all have the same feel for the inner soul of this thing. Think of it as a big band jazz score — it puts everybody's mind in the same place, even when there's still plenty of room for the stars to improvise.

Your document is a sort of intermediary between your mind and the real world. It ensures that what you have in mind is something that the real world is able to handle, and that what you end up with will be what you originally had in mind.

Finally, remember the adage to which any salty gamer will attest: "Great art is in the details." Brilliant details flow naturally from the general gestalt as though they were present in that first flash of inspiration. But once you get into the hands-on implementation, it's easy to lose that spark.

The Challenge

Prototyping parts of the project yourself is definitely a good idea — make whatever rough sketches you can. But again, it's those details that count. The more details your imagination can hold, the greater a masterpiece your work will be.

Working from a document has a flip side, as well. Developing an exciting game has to be exciting. Some of the best parts of many projects were discovered in the heat of last-minute deadline panic. True, the pressures of time and cost budgeting don't allow for perpetual reiteration of concept, but you simply cannot expect a killer game to come out of dry, predictable work. The challenge is to create a design document that will allow your project to tolerate surprise adaptations without losing the integrity of its original direction and scope.

Ten Points for a Successful Design Document

1. Describe not just the body, but the soul. If game development was just an automated input/output issue — something like writing code and being able to predict how it's going to work — you could get by with a dry, descriptive document. The reality is that development is done by people, many of them creative people, who have their own minds; most will want to leave a stamp of that mind on everything they do.

It works like this: You provide specs to the artists and discuss with them what to do. You then visit the programmers and go over their specs. Both groups nod to everything you say.

That night, around 2AM, just as the constellation of C++ is rising in the west, the programmer reaches a midlife crisis and begins to think, "What, a geek programmer the rest of my life? Is this what my mother expected from me? Why, I can design a game just as well as anybody else!" And the hands keep typing code.

Around the same time, the artist has just woken up before his machine, having fallen into a deep stupor while waiting for a complex 3D rendering to finish. Unsure and not really caring

TABLE 1. *The three stages of documentation.*

Documentation Stage	Contents	Purpose
Concept Paper	Genre; target audience; description; most compelling features; market information; cost and time to develop.	It defines the concept, scope, worthiness and feasibility; sells the idea to your client, publisher, employer, and venture capitalist.
Design Document	Description of the body and soul of the entire project, with all the details, and the method by which each element will be implemented.	It ensures that what is produced is what you want to produce.
Production Documents	Time-management charts (Gantt, PERT, and so on); task database; budget spreadsheet; technical specifications; Q/A database.	It implements the design document on time and within budget.

whether he's dreaming or is actually getting paid for all this, immersed in that wild world of artistic genius where fantasy and reality blend as one, the phosphors come together in ways previously unimagined — certainly not by you.

By the next morning, your horse has become a unicorn with two humps. With creative people, instructing is not good enough. You need to inspire.

In your design document, don't satisfy yourself with a detailed description of every article and nuance. Take time to describe the feel that the game should have, the purpose behind each element, the experience each user will have, and any other aspects of the game's soul you can envision and describe.

For example, say you're designing a shooter. You want to train your players to deal with certain challenges before they actually meet them, so you place less lethal mini-challenges a few steps in advance. You're going to have to explain that to everybody on the development team, so they'll understand why certain things are where they are and why they work the way they do. That way, even if (read: when) your team toys with and mangles your ideas as they exist on paper, you can still harbor hopes that the outcome will have the same or similar overall effect. Or maybe even better.

2. MAKE IT READABLE. Go ahead, provide your people with full pages of 10-point, sans serif, 80-characters-per-line text, and demand that they read it. You may want to bundle Advil in the package — for those who actually take the pains to obey orders.

I try to follow at least some of the guidelines of good page layout.

- Plenty of white space
- Serif font for body text
- Bold headers
- Spaces between paragraphs
- Short lines of text
- Direct the eye towards important material
- Use a hierarchical, "2D" format (see what I wrote about outliners in the "Design Documentation Tools" sidebar)

Many instances call for a table, spreadsheet, or chart. Use them and make them sensibly attractive.

3. PRIORITIZE. Now that you realize that you're working with other conscious egos, you'll appreciate the urgency of tagging certain game elements as sacred. True, there are no guarantees, but if you use the tag sparsely enough, it may get some respect. But don't stop there. As long as you're tagging ideas, you'll also want to distinguish between things that you intend to do and things that you'd *like* to do if time, budget, skill sets, and technicalities permit.

Even when you provide animations and prototypes, put the concept into words as well. True, an animation is worth a gigabyte of words, but words can communicate in ways that animations can't.

Then there's the trash bin — things that sounded great, but were trashed for good reason. Refer to them explicitly and explain the reason that they were trashed. If you don't, I can almost guarantee that they will be resurrected. Here's your list of tags:

- Indispensable
- Important
- If Possible
- Rejected

You may wish to use visual symbols to represent these. Don't rely on color, since documents aren't always printed in color.

4. GET INTO THE DETAILS. A document without details is useless. Generalities can be interpreted by anybody in any way that they like. "Thou Shalt Not Kill" meant one thing to Moses and another to a Spanish conquistador. Detailing whom you shouldn't kill and under which circumstanceswould have been more helpful. The same holds true for your document: Once you've described some practical details and given some examples, your idea becomes more concrete — and harder to shove around.

For example, don't just say, "Bronze bird is invincible." Describe exactly what happens to this creature in each possible instance of its being hit, and how it recovers afterward. True, if the animator has any spunk and artistic dignity, you can rest assured that he won't follow your specifications. But at least he'll have a clear idea of what you want to achieve, and his modifications won't seriously alter the related portions of the game.

Don't just say, "At this point, users will have to press the jump key with the arrow key to climb the wall." Describe what will happen if a player tries anything else. Explain why you think users will be able to figure out the combination that you've provided. Explain what about the environment suggests that it's possible to climb this wall.

Again, your artist will come up with something else, perhaps something even more suitable than what you originally conceived. That's real success: When your developers' results come out even closer to your original flash of conception than what you were able to descibe on paper. But this won't happen unless you first lucidly describe your concept.

Don't just say, "Bongo Man is stronger than Bongo Boy, but Boy has faster reflexes." Use tables, spreadsheets, and charts to assign real values to the character's speed of movement, how many hits the character can take, how much damamge the character's hits do, how many cels it takes to animate a hit, and so on. This sort of spreadsheet is invaluable in the Q/A and tweeking stages of production.

Don't just say, "Most people will figure out the whole game in a few days." Make a chart of predicted product life in different households, indicating at which points in time you expect various features to be discovered. User testing will later provide valuable feedback for designing your next game.

5. SOME THINGS MUST BE DEMONSTRATED. Sometimes a few rough sketches are enough, but if the idea is truly important to your concept of the project, you may want to make a rough animation yourself. When behaviors of elements become too complex and ambiguous to describe on paper, you'll want to make a prototype. A side benefit of prototyping is that this practice often leads to a simpler, more elegant solution.

Even when you provide animations and prototypes, put the concept in words as well. True, an animation is worth a gigabyte of words, but words can communicate in ways that animations can't. Words also clearly spell out the vital nuances that may be missed when watching the animation.

6. NOT JUST "WHAT" BUT "HOW." In the real world, the "how" determines the "what." For example, suppose you've opted for claymation. Work out the process of how the images will be captured and *document everything*. What material and what color should the backdrop be? What camera should be used and why? What are the steps for processing the captured frames? And on and on. If you've tried it, you'll know that any one of these factors can have a serious impact on the end result.

Or suppose you're working on a motorcycle racing game. You state that the motorbikes must be balanced by their differing pros and cons. You even provide a chart that shows how balanced they are. Then you state that tweaking will be necessary. State *how* you plan to tweak — what is the process? Suppose the main character in your game is the Phantom of the Opera. Describe how theplayer's keyboard is mapped as a pipe organ. Provide a map of each key. Specify how many channels of sound will be available. Talk it over with your programmer and work out every detail of *how*. Then document it. Two different "hows" can mean two very different results.

7. PROVIDE ALTERNATIVES. Project managers spend a lot of time with their Gantts and PERTs. Personally, I can't really say that this stuff is effective for game development — principally because there are just so many unknowables. The more radical and pioneering your game technology, the less predictable the development stream is going to be. The best thing you can do to ensure that your team reaches your milestones on schedule is to provide more than one way of doing things.

Lets go back to the keyboard as pipe organ example. Your engineer describes to you the ultimate method of getting awesome and funky results with tremendous power and depth to the user — at a cost of about 50 person-hours to implement. As with everything else we've discussed, you document the whole thing.

You can't stop there. You've got to ask, "What would it take if we just wanted a trimmed-down, eight-channel pipe-organ? And what will we need to achieve the bare minimum? And what if we just had some assistant doing this?" And then you document all that as well. When the FedEx truck is on its way over for the final daily pickup, you'll be able to save your skin with a simple, "OK, do Plan C."

One of the biggest mistakes I've made in product design is asking engineers, "Can it be done?" Unless you're asking a first-class programmer, the question is useless. More specifically, responses fall into one of three categories:

(Lousy programmer) "Sure, that's no problem."

(Mediocre programmer) "Nope. Can't be done."

(First-class programmer) "I could do it like this and it'll take two weeks. Or I could make a slight modification like this and it'll take five hours."

Always ask for more than one alternative and an estimate of how long each will take. Then indicate your preference — do this is if we have time, or this if we don't.

8. GIVE IT A LIFE. I've already warned you against strangling the inspiration and spontaneous creativity that comes with the excitement and fun of seeing ideas become living objects in your hands. You've got to allow your document to tolerate change — by you or by (hopefully intelligent) others.

Design Documentation Tools

1. Lots of paper and pencils (nothing digital has yet replaced paper and pencil for hacking out ideas).
2. Even more erasers.
3. A good integrated document application that includes features such as:
   - **a.** A good outliner. Most decent word processors have an outline mode. It's amazing how few people ever try out this feature. Some word processors will even automatically number and sub-number your items for you on the fly. An outliner makes it much easier to create documents in a nonlinear fashion, jumping around and inserting and moving information from place to place. An outliner also makes it easy to produce what I call "two-dimensional documents" — meaning documents that can be read both vertically and horizontally. Moving the eye vertically, a reader gets a feel for the contents and major concerns. Moving horizontally provides the reader with detailed information.
   - **b.** The ability to link illustrations to text.
   - **c.** Cross-reference updating.
   - **d.** Spreadsheets.
   - **e.** Tables.
   - **f.** Hot links are great.

   I recommend Nisus Writer (Paragon Concepts) as the most powerful document processing tool, but ClarisWorks (Claris Corp.) is still the best integrated application available for both Mac and Windows. There are all sorts of project management tools available. I reserve those for the next stage of development.
4. A prototyping tool. mTROPOLIS, from mFactory, is a killer for determining behaviors of objects and getting things to happen really quickly. Macromedia Director is still the standard for quick and dirty animations. You may also need a 3D modeler, such as 3D Studio.
5. A basic library of sound effects. You don't have to make the final decision, but you'll probably want to try a few things yourself and give your audio engineer some idea of what you want.
6. Some pieces of art representative of the style that you intend to use. You'll probably need to make a list and use it to work with an artist while you are composing your document.
7. A knowledgeable friend to read the thing over thoroughly once it's finished and point out the remissions, contradictions, and ambiguities that every author fails to notice.

I first learned this lesson as a music composition major at the University of British Columbia. With much toil, I had written a neo-renaissance brass quintet of which I was quite proud. My professor liked it, too. When we brought it to the college's star brass quintet for rehearsal, however, I passed through several stages of horror, disbelief, indignation, and clinical depression within ten seconds. The quintet began to play, then stopped on signal from the tuba player. The fellow took out his pencil and began to change a few notes, and then everyone continued. It happened more than once.

My professor, noting my sudden faint state of health, turned to me and commented, "Don't worry, they did that to Mozart as well. And they're usually right."

The fact is, no matter how good something looks on paper, the greatest expert still modifies things when it enters the concrete world of objective perception. Nevertheless, you don't want to witness the ruthless rape of your design and dreams. Rather, you're hoping for a kind of organic growth — ideas growing naturally out of the seeds you've planted without needing foreign limbs and bodies grafted onto them.

Here are some tips for creating a document that can tolerate change without corrupting the original idea or sabotaging the development process:

- Make certain to engrave in stone those aspects that are so essential to the game concept that they must not be changed.
- Make certain everybody understands the feel that the game is supposed to have and the purpose of each of its details.
- If information is repeated, it must be cross-referenced. Otherwise, if there are changes, you can end up with contradictory instructions.

And here are some tips for the actual implementation stage:

- When a change is suggested, check back in your design document and see if it is in concordance with the "soul" of your game.
- Check whether this is just an isolated change, or it's of major global ecological impact (see "The Ecology of Improvement"). If it's the latter, save it for your next project.
- Update the design document and include the reasons for the change. Or if you didn't make the change, say so and explain why it was rejected.
- Changes, deletions, and rejected ideas should be retained in a master document to avoid discussing the same thing twice.
- Everyone must be working from the same version. Past versions should be destroyed.
- *Crucial, Vital, and Urgent:* The design document must be maintained under *one person's supervision only*.

9. Nobody Should Be Able to Say, "I Did It That Way Because I Couldn't Find Any Reference to It in the Document." I've seen documents that didn't even have

The Ecology of Improvement a.k.a. The Freeman Elegance-Gestalt Principle

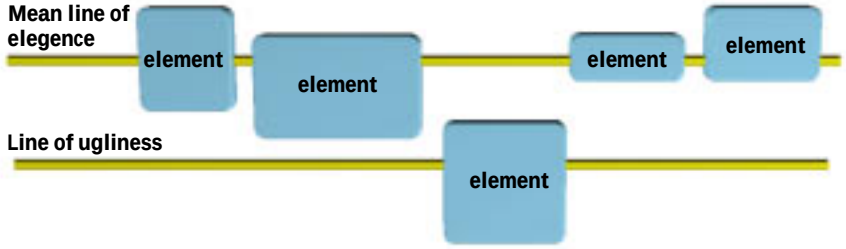
Improving the elegance of one part of an entity without addressing the gestalt of the whole will negatively effect the perceived elegance of every other part of that entity.

For example, you just had the leaks in your car's brake system patched, causing such pressure that your whole worn-out brake system blows out on you. Or you just bought a new pair of jogging shoes, which make your previously sufficient blue jeans look plain tacky.

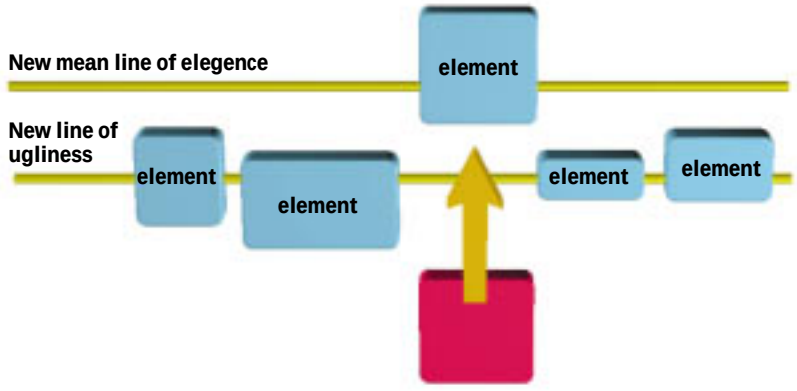
Or you just found a way to add realism to the movement of one of your characters, rendering all other characters jerky and sick in comparison. "Amazing," you mutter, that pale look of what-are-we-going-to-do-now-really-quickly all over your face. "They all looked great yesterday."

The lesson: Don't consider change in one area without first considering its ramifications on every other part of your design.

1 One element of your game is conspicuously inferior.



2 Somebody figures out a way to wildly improve that one element. Now many elements of your game appear conspicuously inferior.



Anatomy of a Design Document

Here's a suggested structure for your document. In actuality, you'll need to tailor the structure to suit your particular project. But my guess is that this example is pretty close to the industry standard. If you're working with a team that you have a history with, or if the game is following an established precedent, your document may be as short as 30 or so pages. That's short. Often, these documents can end up in the hundreds of pages. They're getting longer and longer as the industry matures.

Overview (sort of a recap and revision of the original concept paper)

The User Experience

- To what genre does this game belong? (We haven't really defined genres yet in this industry, as they have in Hollywood, so you'll have to be a little more specific. Best to describe similar products.)
- What part(s) of the brain are you attacking? (Reflex response? Imagination? Problem solving? Strategy?)
- What are the most compelling aspects of this game? (Give this section much consideration. It is the core of your "mission statement.")
- How deep is the product? (Is this a one-shot deal for buyers, or something they can keep getting into for a while?)

The Platform

- Arcade, home, or school?
- PC, Mac, console, or multiplatform?

The Users

- General audience
- Base target audience
- Describe typical users

Time

- Game-play time (This is one of the most crucial, decisive issues in the design of any game. It's impossible to make meaningful design decisions without establishing predicted maximum, minimum, and mean game-play times first.)
- Product life (How many days/months/years do you expect the user to keep coming back to your game? This period is usually based on how long it will take users to figure everything out and master it.)

Basic Concepts (gives a feel for the game, why things are the way they are, and what the essential, indispensable elements are)

Storyline

- The background story (if applicable)
- Storyline or object of the game play
- Rules of the game (Justify these rules and explain what you expect to see from them.)

Heroes and Villains

- Include biographical information and descriptions, even if this won't be implemented in the software. This will be of great help to the artists and animators.

Novelties and Compelling Features

- This is your chance to state the things you could not bear to see disappear from this project, and justify your emotional attachment to them.

Navigation Chart (an illustration of how parts of the game link to each other)

Entry and exit
Main menu
Level movement
Access to preferences and credits

Global Behaviors (ensures that your game will have a consistent feel to it, avoids serious run-ins with the programmers)

Run through all the standard elements of your project (sprites, buttons, life-bars, input devices, and so on) and describe their behaviors in every circumstance you can imagine. Your programmers will shower you with wreaths for this one. Later, you can go change things on them — as long as the objects remain consistent, and everything is justified.

Illustrate the motion of each animation, at least in stick form. For 2D scrollers, fighters, and the like, you'll want to describe things cel by cel. With other projects, rough sketches of general movements and their approximate duration in microseconds may be enough.

If you are relying on a specific input device, justify your button-mapping and button-combination decisions.

Scenes and Action (In an adventure game, this will take up most of your document)

Include preferences, credits, and main menu.

In subchapters, lay out consistent behaviors of local elements.

Very often, a storyboard (that is, a series of panels illustrating each scenario) is provided. In many projects, however, this is clumsy and impractical.

Lists of Resources

You'll have to go over this with a fine-tooth comb to make sure it's thorough. Leaving out even a few items, or failing to describe them clearly, could prove a major source of exasperation later on.

This section comprises detailed lists of animations, sounds, music, narration, sprites, backgrounds — everything that needs to be created besides code.

Cross reference each item to its main reference in the document.

the pages numbered. And then they complain that people didn't follow instructions. Every good word processor will auto-number pages and print the date and title in the header or footer of every page. Some will even allow you to change the header at new chapters. Use bold text to direct attention to important material. Repeat yourself in different parts of the document as much as you like, as long as you cross-reference so you can update everything together as well. Make a thorough Table of Contents.

You may wish to write your document using HTML and provide hot links. Some progressive word processors provide hot link capabilities without HTML. But remember that more often than not, people prefer to work from a hard copy. (That way there's something to read while rebooting after the hourly system crash.)

10. DELIVER IT IN GOOD CONDITION. After all this, you need to do whatever you can to facilitate everyone actually reading and using the thing. A pile of papers doesn't get read — it doesn't look important enough. Only things with hard covers look important.

Create a list of everyone who is supposed to have a copy. Keep the list. Print out the whole thing with the date in the header of each page. Have holes made and put it in binders. Label the spine and cover of each binder. When there are updates, provide *everyone* with the revised pages. At some points, you may need to provide new books *and throw out the old ones.*

You may wish to write your document using HTML and provide hot links, but remember that more often than not, people prefer to work from a hard copy.

In Sum Check...

Movie makers use move scripts. Architects use blueprints. Musicians use a score. According to ancient hearsay evidence, even the Cosmic Creator created a design document — which He later let a few prophets take a peek at — before the primal "Let there be light!" So game developers, following their Supernal Role Model, can certainly do the same. Do it right and it's smooth sailing the rest of the way. ■

Tzvi Freeman teaches Game Design and Documentation at Digipen School of Computer Gaming in Vancouver, British Columbia, Canada. He has designed several commercial games and has acted as a consultant on many others. He can be reached at TzviF@aol.com.

ADVERTISER INDEX