Detailed Game Design Documentation Template

V2.5 April 17, 2017

Template

This is an outline or template for your design documentation – a blueprint for your blueprint. Use the parts of this that are helpful to describe and define your game and ignore the rest. Every game is different, and so each game will emphasize some areas more than others.

Importance of Documentation

Document your game design so that you and any other stakeholders in the game (the team, management, etc.) know in as much detail as they need what the game is, what is needed to complete it, and why it's worth building.

Your design documentation serves as the embodiment of the vision for the game shared by the team. It's also be a roadmap and guide for during development. This documentation covers the qualities of the player's experience, how the game behaves to support that, and the reasons why you made various decisions along the way.

Note that all this is necessary even if you are alone on the project: six months from now it will be very easy to forget why you wanted something a certain way. Having to re-think it later can lead you to poor decisions or design thrashing (where you bounce between two lousy options instead of finding an effective one).

In other words, your design documentation is both *backward-looking* and *forward-looking*: it records the decisions you made and their rationales, and provides a map and a guide for developing the game going forward.

Guidelines to Follow

Creating your design docs as a mixture of text, systems and UX diagrams, equations, etc., helps anyone who reads it understand your game. **You are encouraged to use pictures, diagrams, flow models, etc., wherever possible.** If there's something you can adequately represent with a picture or diagram rather than a block of text, do so!

Treat your design document like a blueprint for a building. Be precise in describing the design details. Avoid being vague at all costs, and state aspects of the design in present tense (the game *behaves* this way, not "the game will behave"). Avoid waffling words like "the game might…" This documentation is where you stake out the game in no uncertain terms. Except when describing high-level aspects of the design, do not use qualitative phrases like "this enemy moves fast" or "this results in a big explosion" or "this turret turns left and right."

How fast? How big? How far? Whenever a quantity is called for (number of skills or levels, amount of damage, etc.), supply a number, a defined range, or a reference to an equation or other determining factor. If a decision on a specific number has not yet been made, say so, and call it out as an issue to be resolved. Being precise in your writing forces you to eliminate the fuzziness that can accompany a design that is all in your head. It is far easier to discuss and find the holes in a proposal than an idea.

That said, here is a warning: creating too much design documentation can be as detrimental to your game as creating too little. If you know the details of a part of the game, document it using text, diagrams, mockups, etc. If you're not sure, document what you're trying to create, your best ideas for how to do it, and what paths you've discarded – and then create prototypes (paper or electronic) to test out the ideas. Don't waste time documenting a lot of different options, much less arguing about them: find the area of uncertainty and prototype it. Then once the path you want to take is clear, document that and why you chose it.

The Goal

By combining text, diagrams, mockups, and/or prototypes, your design documentation provides enough information for anyone concerned to understand

- what the game is
- what the player's experience is – why they will play it
- what your goals are for it – why you are making it
- what its current status is
- how, in precise detail, to build it

Anything that doesn't contribute to this is wasted effort to be avoided or removed.

Keep it Current

Any design documentation is only as good as it is current. Game designs are often referred to as "living" documents, in that they change as the game development progresses. As such, plan from the beginning to update your design as development proceeds. You will find new ideas, uncover new problems, and change the overall design as you make the game.

While your design docs need to be complete enough to guide the implementation of the game, do not treat them as a "design bible" that is written once and never changed. Doing so is the fastest way to make your design documentation obsolete and irrelevant, and leave the team without a usable plan for how to create the game.

Encourage team members to comment on existing design docs "off to the side" – this can mean different things depending on the technology you're using. If it's possible to have people comment on the same page as the design (as in Word, Google Docs, and similar programs) this can work well and allow for conversations about the design within the doc. Otherwise you may need to have a "shadow doc" that parallels the main documents but contains *only* the comments and discussion about the design docs they accompany. The design docs themselves contain only the current state of the design, not any conversations or commentary.

As part of maintaining the design docs, either include these in your version control system to keep a log of changes and dates, or insert a log at the top of each section to show its history.

Divide and Conquer

On a multi-person team, each designer (and sometimes producers and others) may be assigned particular parts of the documentation to maintain and keep current. It is typically a good idea to assign the more general parts of the documentation to the more senior members of the design team, and the more detailed parts to the more junior members. That is, a junior designer is

more likely to be assigned a particular feature or system and the Lead Designer or equivalent the responsibility of maintaining the game concept and vision.

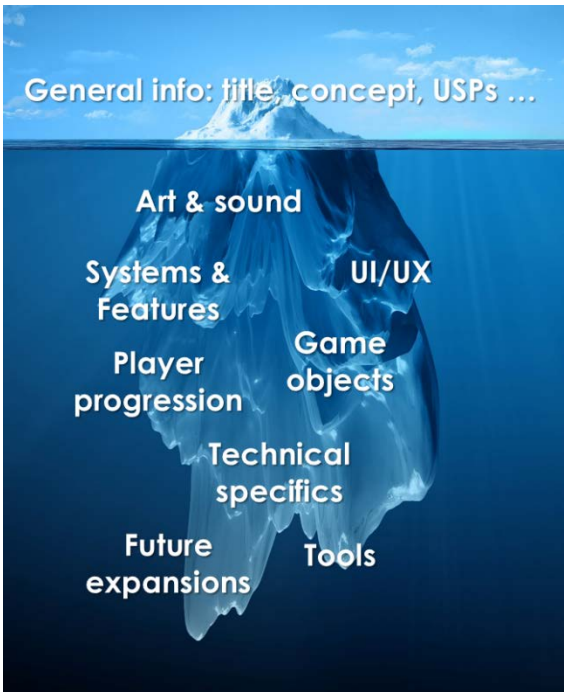
Organization and Adaptation
There is no one best method for organizing a game's design docs. What's presented here will probably work well for your game, but adapt it as necessary. Remember, the point is to give anyone who reads it the information they need in the most effective way: don't make them hunt for the particular information they want, nor be left wondering about some aspect of the game that concerns them.

It's useful to present your design docs so that the broadest, most general information is immediately apparent, and then progressively reveal more detailed information as needed. This is the model used here. The "main page" covered here gives the reader an immediate understanding of the game's purpose and current status. On this page links leads the interested reader to additional levels of detail. These are then separate sections that describe each part of the design in progressively more detail, with links between sections as appropriate.

Examples
There are many good example design docs online showing the level of detail needed. One good example is at http://www.gamasutra.com/view/feature/130127/design_document_play_with_fire.php. This shows the organization and level of detail needed in a production design doc.

The Design's Main Page: Tip of the Iceberg

Each section of the design doc is shown here in the template along with a brief description, starting with those parts that are on the main page.



The sections shown here are used in your design doc (as appropriate). The descriptions given with each section do not appear in your design docs. Use these as a guide for what information about your game to include.

Working Title
Your game's title communicates the gameplay and the style of the game

Current status
In a few sentences, state the game's current status: is it still being designed, are parts being prototyped, or is the whole thing ready for greenlight for production? Is it on schedule? Are there significant issues worth calling out as part of general status?

A useful way to phrase this is as the game's own Scrum report:

- What parts of the design are complete or have been reviewed?
- Which parts are being worked on?
- What if any blockers to progress are there
- What if any debt or major issues have arisen?

Concept Statement
In one or two sentences ("your game in a tweet," or "a few words that create a question") describe the most important aspects of your game. What is its theme and setting? What makes the game fun? How many players are there? Is it a quick pick-up game or a huge epic? Why would someone want to play the game?

Don't try to cram too much information into a quick statement, but don't leave out vital aspects of your game either. This can be a tough balancing act. Expect to spend some time iterating and polishing this.

Genre
While genre statements about the gameplay ("this is a bullet-hell platformer") are helpful to calibrate what the game is and isn't, avoid vague statements that don't actually inform the reader ("this is a Castlevania-like game") and that may hide lazy thinking about the details of the game's design (if it's "a Castlevania-like game" what exactly does that mean for *your* game?).

Note that single-genre games are often less interesting as the space has likely been well-explored. Genre hybrids or combinations can create interesting new angles on existing gameplay, but beware of a "something for everyone" design that contains multiple genres, as these often end up being a mishmash that's fun for no one.

Target Audience
Who is the target player for this game? What other games might they have played? What is their age range or other relevant interests or attributes? What is the desired ESRB rating?

Concept Paragraph and Unique Selling Points
Expanding on the Concept Statement, provide the reader with an overview of the gameplay. *Briefly* cover the game's core loops, player progression, setting, look and feel, and any other aspects that are necessary to understand the core appeal of the game.

In particular, highlight your game's *unique selling points* (USPs): what are the things that makes your game fresh, intriguing, new, and worth someone's time? Your game's USPs must be clearly understood by you and anyone who reads the design docs. If for some reason you can't state these clearly, or if they don't feel compelling, this is a sign that you need to rethink or refocus your design. By reading your game's USPs, a player (who isn't you!) can see why they would want to play your game instead of all the others like it that already exist.

A link to a current prototype, or even a video of a prototype, goes here once one exists.

Player Experience
Who is the player in the game? What is the setting? How long does the game last? What is the fantasy or aspiration the game grants the player? What emotions does the player feel by playing the game? What are the major phases of the player's experience in the game? (This section can link to "Player Objectives and Progression" and leave most of the detail to that part of the design.)

Key Moments
As part of the player experience, and to help others understand the design, *briefly* discuss a few "key moments" in the game. Typical examples include an initial positive experience in the first 1-5 minutes of the game; key struggles that must be overcome; and major victory, achievement, or resolution points the player experiences as part of their arc through the game.

Art, sound and music
This section contains high-level information and examples about the visual and auditory style for the game, as well as links to more specific information.

The high-level information here includes reference and concept art (and where possible, music) to give the reader an idea of the visual style and tone for the game. Reference art may be taken from movies, books, other games, etc., so long as it is not used in the game in any way. Concept art (if any) must be original to the game. This art gives the reader a clear idea of the environments, characters, opponents, objects, major locations, key events, user interface, etc. in the game.

A separated (linked) doc, usually in the form of a spreadsheet, delineates all known art and sound assets that are needed for the game. This includes all user interface assets, animations, environments, characters, etc, **in complete detail**.

Current Target Platform (and any system requirements)
What is the first or current platform for the game – PC, web, tablet, phone, or something else? Keep this to the current target platform; leave off any "future" desired platforms.

Competition
What games are similar from the player's point of view, or would compete directly with your game? Show their name, release date, any revenue and budget information you can find, and a brief description of each. Also include a brief description of what sets your game apart.

Monetization
If the game is "free to play" or has in-game purchases, these need to be wholly integrated with the rest of the design. This section provides an overview of the monetization philosophy and the major systems. It is heavily linked with the systems and features section.

Links to Detailed Sections
From the main page, the reader must be able to easily find the detailed information they want by following links to the following detailed sections. These appear separately, as on their own web pages or in their own chapters.

Player objectives and progression

- Who is the player in the game?
- Synopsis of what the player knows when the game starts, and any following in-game narrative
- Player's primary goals and how they progress to them, including which parts are linear or based on player choices
- Moment-by-moment gameplay (with links to UI section)
- core loops and outer loops
- reference back to Key Moments

Game world

- Backstory overview – important elements of the world fiction that set the stage for the game, but do not directly affect the gameplay
    - Link to full world fiction
- World organization: MVP levels, locations, zones, chapters – including introductory or tutorial areas
- Game world physics: running, flying, falling, etc., -- anything of note for the design
- How does the player move through the game in terms of locations, chapters, etc.
- Easter Eggs planned, if any

User Interface

- Specifics of all needed control schemes (selection, activation, radial or context menu use, tapping, dragging, etc.), ensuring that these are consistent across the game
- Screen flow mockups: each screen/menu in full functional detail (wireframe, no art), showing the player's flow through all aspects of the game.
- Mockups for on-screen controls (with links to individual features and systems)
- Help system details
- Game options: audio, visual, game save, etc.

MVP Systems and Features

- All major features such as character creation, crafting, combat, dialog, etc., including areas for monetization (if any), and specific puzzles or similar
    - Intent, the "why" behind the feature
    - The moment-by-moment gameplay for each feature or system
    - The monetization aspects of each system or feature (if applicable)
    - Lots of pictures and diagrams – give the reader the feel for the feature or system
    - Links to technical docs for programmer consumption
        - Math, data structures, format of external data docs
        - Naming schemes for external directories and files
    - Break down into Epics and User Stories for implementation
- Include a description of the UI for each (with links back to the main UI section)

Game Objects

- Listing of all NPCs and monsters, including

    - NPC backstory, narrative, dialog (links to specific dialog data files)
    - Attributes, attacks, special powers
- Game objects
    - Attributes and their values/ranges
    - Behaviors
- Link to external spreadsheet containing each object's data and attribute information

Localization

- Description of the initial localization plan: English-only, EFIGS, or other.
- Technical description of localization syntax for strings
- Technical description for localization string name conventions
- Link to external localization string files

Tools

A description of tools needed for designers, programmers, and others to create the game. This does not include standard word-processing or program editors, but focuses on tools such as:

- World creation and layout
- Object definition
- Missions/quests
- Others as needed to implement the game

Technical documentation

- Major technical areas and risks
- Target hardware and specific requirements/assumptions
- Infrastructure
    - directory structure and file naming conventions
    - data file format
- Server, client, and network architecture (as needed)
- Artificial intelligence and other autonomous/procedural systems
    - Procedural world creation
    - Pathfinding
    - Economy
- Technical (programming doc for each major feature or system, linked to the design doc for each above. These docs are for programmers, and focus on implementation details rather than the design itself or its rationale.
- Game cheats for testing

Ideas and Expansions

Non-MVP feature ideas for future expansion are kept here.

Unresolved Questions

As questions and concerns arise, they are listed here. When resolved, the question or concern is not deleted, but is moved to a "Resolved" section along with a link to the document that describes the design that answers the concern.

Prototypes

Links to prototypes not already referenced, if any, are here.