Useability & Testing

Heuristics

: involving or serving as an aid to learning, discovery, or problem-solving by experimental and especially trial-and-error methods <heuristic techniques>
<a heuristic assumption>;

also : of or relating to exploratory problem-solving techniques that utilize self-educating techniques (as the evaluation of feedback) to improve performance

http://www.merriam-webster.com/dictionary/heuristic

Ten Usability Heuristics

by Jakob Nielsen

These are ten general principles for user interface design. They are called "heuristics" because they are more in the nature of rules of thumb than specific usability guidelines.

http://www.useit.com/papers/heuristic/heuristic_list.html

Visibility of system status
The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

Match between system and the real world
The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

User control and freedom
Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

Consistency and standards
Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

Error prevention
Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

Flexibility and efficiency of use
Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

Aesthetic and minimalist design
Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

Help users recognize, diagnose, and recover from errors
Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

Help and documentation
Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Documentation and Testing

Document Types for Software and Game Projects

Embedded Documentation:
The Code Comment

Code comments explain
the purpose of a section of code.

Most programming languages recognize
the two common comment markers:

- // for single line comments
- /* <comment here> */ for block comments

Clear commenting makes it easier
to understand what a program is doing.

It also makes it easier to have more than
one person work on a project.

Document outside code
embedded in projects.

Include the book or website of origin, and
any copyright or author information needed to contact and
verify the use terms of the software.

Some developers allow free use of their resources,
others do not.

The rules for use may change depending on academic
status, non-commercial use, or commercial use.

The Design Requirements Document

the term *requirements* refers to the general set of documents that describe what a project is supposed to accomplish and how the project is supposed to be created and implemented.

A general set of requirements includes documents spelling out the various requirements for the project — the "what" — as well as specifications documents spelling out the rules for creating and developing the project — the "how".

http://philosophe.com/design/requirements/

In simplest form, the design requirements document describes the necessary functions and resources needed to complete a project.

It may be brief - a page or two for a small project, or may be hundreds of pages long for a big project.

Use Cases & Test Cases

Use Case: A **use case** in software engineering and systems engineering is a description of a system's behavior as it responds to a request that originates from outside of that system.

In other words, a use case describes "who" can do "what" with the system in question.

Test Case: A **test case** in software engineering is a set of conditions or variables under which a tester will determine whether an application or software system is working correctly or not