User Interface Design for Games

David Kieras

University of Michigan

Introduction: User Interface Design for Games

Desirable Properties of User Interfaces for Game Software

Attractiveness, enjoyability.
- E.g. quality of graphics, sound, animation, etc.
- Strong influence on marketability.
- Not further discussed here - I'm the wrong person!

Usability
- How easy is it learn?
- How easy is it to use?
- Related to "playability," but not the same thing.

Usability is a technical problem, and has technical solutions.
- Not an "art" any more than any other aspect of software.
- A large, active discipline within computer science.
- Well-developed concepts, theory, and methodology.

User interface field has many useful concepts and techniques for designing software to support work.

But what about games?

Overview of presentation
- Usability of game software versus work software
- Overview of usability principles and techniques
- Example for discussion

Software for Work
versus
Software for Fun

Work is less productive if the software is unnecessarily hard to use.

The user must get something done, so operating the software should be as easy as possible.
Fist-fighting with the software is pointless.
Save time and effort for *actual* work.

Games are not fun unless some difficulty is involved.

Something must be hard to learn, hard to figure out, hard to execute.
The user is seeking the thrill of accomplishment.

Consider a button labeled "Solve Problem."

Best possible design for work software.
Worst possible design for game software.

- At best, a "cheat" function.

Creative vs. Instrumental Work

Creative, computer-independent, non-routine - the point or goal of the work

Writer - thinks of the Great American Novel.
Researcher - chooses data analysis methodology and interprets results.
Programmer - comes up with program concept and design.

Instrumental, computer-specific, routine - the means of getting the work done

Inserting and deleting words and phrases.
Specifying parameters and data input for statistical analysis.
Editing source code, using a debugger.

Design Goal: Facilitate creative work by providing powerful and highly usable computer instruments (tools).

Fun vs. Instrumental Aspects of Computer Games

Fun, creative, challenging, interesting, enjoyable things to do.

Learning to fly a simulated F-18 in air combat.
Figuring out how to activate the Rocket Ship in Myst.
Planning next move in Spaceward Ho!

Interacting with the game software in order to have fun.

Hit Q to change radar mode, TAB to change radar range scale in F-18.
Where and how to click to move backward in Myst.
Executing sequence of actions to build and move a fleet in Spaceward Ho!

Design Goal: Maximize time/effort spent on fun, minimize time/effort spent learning and using the instruments of interaction.

Wrong Ideas about Usability in Games

Usability is not an issue for games - games are supposed to be challenging!

Games can be difficult for the wrong reasons

Player appreciates relevant difficulty, but not irrelevant hassles!

The user tasks are indeterminate - after all, people can and should do anything they want in a game!

Concept of the game implies that player will do certain activities.

- Move around, use objects in adventure games.
- Make moves, use information in strategy games.

Need to make sure required interactions do not impair fun.

We like it, so what's the problem?

Developer's intuitions about usability are usually wrong.

If you built it, you can't see it from point of view of actual user.

Sales are good, so we must be doing OK.

Effects of poor usability happen *after* the sale, not before - what do former purchasers think?

How much better would sales be if the game were more enjoyable?

Nobody's complained, so what's the problem?

People often have poor awareness of when their time is being wasted, and not just during entertainment!

People often don't bother to complain.

If game is good enough, usability problems might be ignored.

"Cognitive dissonance" effects might be especially strong.

Usability is a Scientific and Technical Issue

Not just a matter of opinion or "what you are used to."

Certain commonly used terms have no technical meaning in the science of usability:

"User-friendly "
"Look and feel"
"Intuitive"
"Natural"

"Usability" means:

Ease of learning

- Can the system be learned quickly and easily, either for the long term , or for immediate, "walk up and use," purposes?

Ease of use

- Can the system be used to accomplish work rapidly, accurately, and without undue effort?

These are the two major aspects of usability.

- Often do not have to be traded off against each other.
- Can be measured, quantified, and even predicted.

Some other aspects of interface design are not usability issues, but can be affected by usability:

Ease of implementation
Attractiveness
Marketability
Motivating value
Entertainment value

If a system is difficult to learn or to use, customers are likely to be dissatisfied eventually, even if it is a market success at first.

Suggested Approach to Developing Usable Game Software

Write out the Entertainment Goal.

What is supposed to make the game entertaining?
What needs to be difficult about it to make it fun?

Identify what is actually difficult about the game.

Apply standard usability techniques to:

- A particular version of the game concept itself.
- A particular version of the user interface.

Different techniques can use:

- Just a design.
- A prototype.
- Complete implementation.

Identify the difficulties that are not part of the Entertainment Goal:

Instrumental difficulties, not part of the game.
Interfere with the Entertainment Goal.
Should be fixed to maximize fun.

Fix the instrumental difficulties with standard usability design concepts.

How to Develop a Usable System

The focus: What do users need to do, and how can the system help them do it?

1. **Perform a task analysis.**
   What does the user need to be able to do?
   Identify what the user *really* wants to do, not just what commands the user will issue.
   - E.g. "conquer planet," not just "move ships."

2. **Choose system functions that will support the task.**
   How can the system help the user do the task?
   Don't default to "obvious" functions, often better choices.

3. **Choose and adopt some usability specifications.**
   Set usability targets - e.g. "Learn basics in 15 minutes."

4. **Choose initial interface design for chosen functionality.**
   Ease of learning, speed, accuracy of use.
   Usability guidelines, platform conventions, experience

5. **Evaluate the usability of the design**

6. **Correct any problems and repeat evaluation until**
   No more problems identified.
   No more time or money.
   A median 50% improvement per iteration.
   - (Landauer, 1995)

How to Evaluate a User Interface Design

User testing methodology

Have representative users do representative tasks.

- "Benchmark" tasks
- Include "start from scratch" learning situation.

Look for problems, measure time required, errors.

A well-established, effective methodology.

- Many good sources for specific suggestions.

Analytic model methodology

Simulate, analyze user procedures required by design.

Estimate how long it will take

- To learn interface procedures
- To execute benchmark tasks.

Identify problems in procedure learning or inference

- "Cognitive Walkthrough" methodology

Get some information without testing human users.

Fast comparisons of alternative designs.

Procedure Basics

The Procedures Required to Use a System are a Major Usability Factor

GOMS Models

Two Approaches for Evaluating the Procedural Quality of an Interface

The Keystroke-Level Model Method

Operators and Times for the Keystroke-Level Model

GOMS Procedure Model Method

Example: VCR Procedures

Methods for Setting the Clock

Methods for Recording a Program

Methods for Recording a Program (continued)

Methods for Stopping and Canceling a Recording

Methods for Stopping and Canceling a Recording (continued)

The Procedures Required to Use a System are a Major Usability Factor

Common view: User interface consists of the input devices and what appears on the screen.

More correct view: The interface also includes the procedures that the user has to follow to get his or her work done.

Example:
Q. How do I defrangulate a gizmo?
A. Step 1. Select the gizmo.
Step 2. Point to the "Action" menu.
. . .

Research results: A system with simple, consistent, and fast procedures is easy to learn and fast to use.

May be more important than any other single factor
- Tremendous learning and ease-of-use benefits.

Probably the actual reason for the success of the Macintosh
- A few simple, consistent procedures for the most frequent tasks are shared by all applications and the operating system.
- Careful detailed design of procedures, e.g. of defaults.

Simple, consistent, fast procedures can be designed for all platforms and all interface styles.

GOMS models: An approach to analyzing procedural quality of an interface.

GOMS Models

An approach to describing the knowledge of procedures that a user must have in order to operate a system.

Proposed by Card, Moran, & Newell (1983)

A GOMS model consists of a description of:

***G*oals**

What goals the user can accomplish with the system

***O*perators**

What basic actions the user can perform

***M*ethods**

What sequences of operators (procedures) the user should follow to accomplish each goal

***S*election Rules**

Which of several methods the user should apply to accomplish a goal, given a specific situation

Two well-established GOMS methodologies.

Keystroke-Level Model.
GOMS Procedure Model.

The Keystroke-Level Model Method

Analysis:

1. Start with a specified interface design.
2. Choose one or more representative task scenarios (use cases)
3. List the keystroke-level actions (operators) required in the scenario.
4. Insert mental operators for when user has to stop and think
5. Look up the standard execution time to each operator
6. Add up the execution times for the operators
7. The total is the estimated time to complete the task

Choose design with fastest predicted times.

Especially useful for:

Getting a quick, easily explained answer.

Frequently repeated activities, where even small time inefficiency can be a problem.

Operators and Times for the Keystroke-Level Model

K - Keystroke

Pressing a key or button on the keyboard
Different experience levels have different times

- Good typist (90 wpm): .12 sec
- Average skilled typist (55 wpm): .20 sec
- Average nonsecretarial typist (40 wpm): .28 sec
- Worst typist (unfamiliar with keyboard): 1.2 sec

Pressing SHIFT or CONTROL key is a separate keystroke

P - Point with mouse to a target on the display

Follows Fitts' law.
Typically ranges from .8 to 1.5 sec.
Average is 1.1 sec for large-screen text editing.

B - Press mouse button

Highly practiced, simple reaction.
Takes .1 sec.
Mouse button click (**BB**) takes .2 sec

H - Home hands to keyboard or mouse

Takes .4 sec.

W - Wait for system response

Only when user is idle because can not continue
Have to estimate from system behavior
Often essentially zero in modern systems

M - Mental act of thinking

Routine, brief pauses in stream of activity

- Not "deep thought"

Estimates ranges from .6 to 1.35 sec.
Use 1.2 sec if more precise information not available.

GOMS Procedure Model Method

Analysis:

1. Select a set of top-level user goals.
2. Choose a specified interface design.
3. Write out the general procedures that the user will have to learn and have to execute to accomplish the goals.
   - Can use a specified notation, e.g. NGOMSL.
4. Assess learning and execution implications of procedures.
   - Informal: Unneeded complexity, slow or difficult steps.
   - Formal: Calculate predicted learning and execution times.

Choose the design with the best combination of learning and execution times.

Especially useful for:

Assessing consistency and ease of learning.
Predicting execution time for a large variety of situations.

Example: VCR Procedures

User goals - a subset

- Set the clock
- Record a program
- Stop current recording
- Cancel previously programmed recording

Assumptions

- A cassette is loaded
- Power is on
- Desired channel is already selected

Methods and Selection Rules are somewhat simplified

Example uses NGOMSL notation.

Methods for Setting the Clock

Method for goal: set the clock
Step 1. Press CLOCK button
Step 2. Accomplish goal: set a day and time with the current day and time
Step 3. Press CLOCK button
Step 4. Return with goal accomplished

Method for goal: set a day and time
Step 1. Accomplish goal: set day value to the desired value
Step 2. Press SELECT
Step 3. Accomplish goal: set hour value to the desired value
Step 4. Press SELECT
Step 5. Accomplish goal: set minute value to the desired value
Step 6. Return with goal accomplished

Method for goal: set a value to a desired value
Step 1. Verify that display shows current value is flashing
Step 2. Decide: If display shows value same as desired,
   then return with goal accomplished.
Step 3. Decide: If display shows value smaller than desired,
   then press UP button.
   else press DOWN button.
Step 4. Go to Step 2.

Methods for Recording a Program

Selection rule set for goal: record a program

If you are present when the program starts
and you will be present when the program ends
then accomplish goal: record a program manually

If you are present when the program starts
and you will not be present when the program ends
and you know how long the program lasts
and the VCR clock is set
then accomplish goal: record a program with One-Touch Recording

If you will not be present when the program starts
and you know when the program will start
and you know how long the program lasts
and the VCR clock is set
then accomplish goal: record a program with Timer Recording

Return with goal accomplished

Method for goal: record a program manually

Step 1. Wait for program to start.
Step 2. Hold down REC button.
Step 3. Press PLAY button.
Step 4. Release both buttons.
Step 5. Verify that display shows "REC" and arrow is moving
Step 6. Wait for program to end
Step 7. Press STOP button.
Step 8. Return with goal accomplished.

Method for goal: record a program with One-Touch Recording

Step 1. Wait for program to start
Step 2. Press OTR button.
Step 3. Press OTR button.
Step 4. Decide: If time shown in display is less than length of program, go to Step 3.
Step 5. Return with goal accomplished.

Methods for Recording a Program (continued)

Method for goal: record a program with Timer Recording

Step 1. Press PROGRAM button.
Step 2. Verify that program number flashes
Step 3. Press SELECT button.
Step 4. Accomplish goal: set a day and time with the starting day and time of the program
Step 5. Press SELECT button
Step 6. Accomplish goal: select recording length
Step 7. Press SELECT button
Step 8. Accomplish goal: set channel value to desired value
Step 9. Press CLOCK button
Step 10. Verify the display returns to normal display
Step 11. Press POWER button
Step 12. Verify that display shows "TIMER"
Step 13. Return with goal accomplished

Method for goal: select the recording length

Only certain values are allowed, so special method required

Step 1. Verify that display shows "LGTH"
Step 2. Press UP button.
Step 3. Decide: If time shown in display is less than length of program, go to Step 2.
Step 4. Return with goal accomplished.

Methods for Stopping and Canceling a Recording

Selection rule set for goal: stop an on-going recording
There are subtle cues on the display that could also be used to select the method
If you remember that the recording is manual
then accomplish goal: stop an on-going manual recording
If you remember that the recording is a One-Touch Recording
then accomplish goal: stop a One-Touch Recording
If you remember that the recording is a Timer Recording
then accomplish goal: stop a Timer Recording
Return with goal accomplished.

Method for goal: stop an on-going manual recording
Step 1. Press the STOP button
Step 2. Verify that "REC" disappears and arrow stops moving.
Step 3. Return with goal accomplished.

Method for goal: stop a One-Touch Recording
Actually behaves differently depending on time delay since OTR button last pressed, but this method always works
Step 1. Press OTR button.
Step 2. If display shows a time greater than zero, go to Step 1.
Step 3. Wait several seconds
Step 4. Verify that VCR turns off.
Step 5. Return with goal accomplished.

Methods for Stopping and Canceling a Recording (continued)

Method for goal: stop a Timer Recording
Step 1. Press POWER
Step 2. Verify that display shows "REC" disappears and arrow stops moving.
Step 3. Accomplish goal: cancel a Timer Recording
Step 4. Press POWER
Step 5. Verify that VCR turns off.
Step 6. Return with goal accomplished.

Method for goal: cancel a Timer Recording
Step 1. Press PROGRAM
Step 2. Press SELECT
Step 3. If Display does not show "LNGTH" go to Step 2.
Step 4. Press DOWN.
Step 5. If Display does not show zero, go to Step 4.
Step 6. Press CLOCK
Step 7. Return with goal accomplished

Cognitive Walkthrough Technique

The Cognitive Walkthrough Technique

See Wharton, Rieman, Lewis, & Polson (1994)

Based on cognitive analysis of human-computer interaction.

Represents both problem-solving activity and procedural requirements.

Basis: Can user figure out how to use the system?

A good interface will require little learning - it will be obvious what to do.

What will guide user to do the correct actions?

Can the user learn the methods just by interacting with the system?

Especially, what can the user see in the interface that will suggest what to do?

How to do a Cognitive Walkthrough

See Wharton, Rieman, Lewis, & Polson (1994)

Start with:

A proposed user interface design - can be on paper.
A defined user population.
A set of representative tasks.
The correct sequence of actions required by the design for each task.

Procedure:

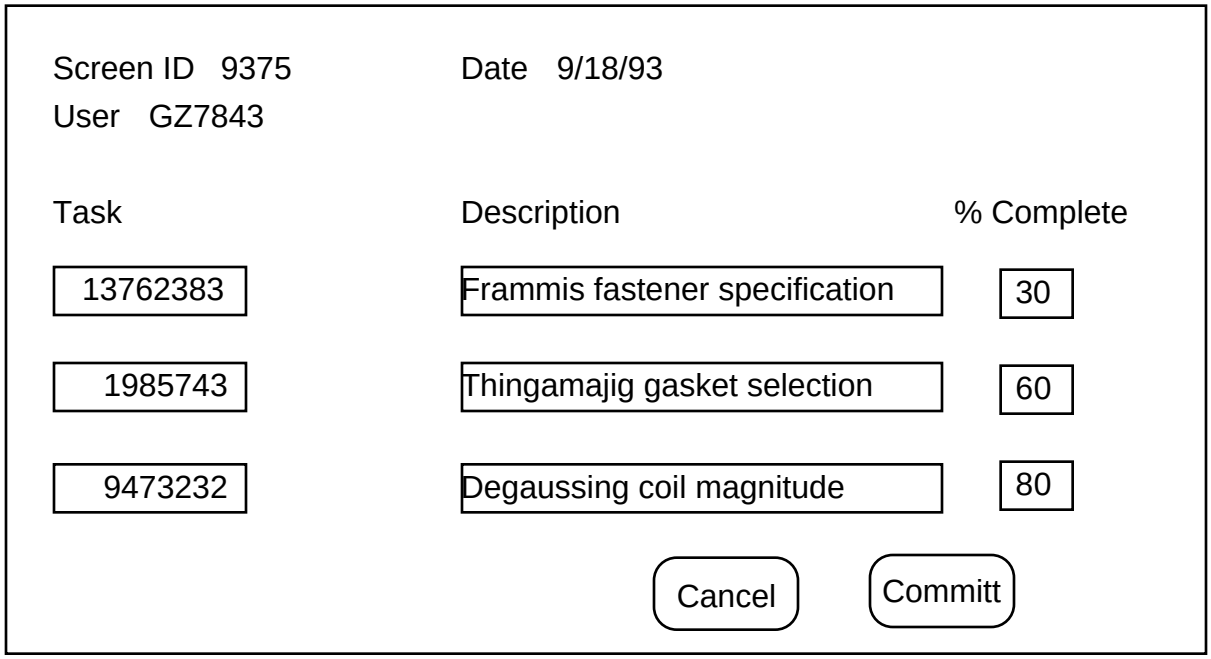
1. For each action in the sequence, make up a credible story about why the user would know to perform it.
   - How does the user know what the next step is?
   - If no credible explanation, means that there is a problem in the interface.
2. Focus on answering four questions about each step:
   - Will the user being trying to produce the intended effect?
   - Will the user be able to see the control for the action?
   - Will the user be able to recognize that the control produces the intended effect?
   - After taking the action, will the users be able to tell from the feedback that they succeeded?
3. Examine the stories - sources of the problem suggest fixes:
   - Better labels to suggest what to do.
   - Eliminate the need for the action.
   - Rearrange actions so that system prompts for required actions.

Example - A Fragment of a Project Management System

User is engineer working on several projects (tasks).

- Users are familiar with MS Windows on PC computers, but are not assumed to be otherwise computer literate.
- System is not supposed to require much training.
- User is supposed to update database on how complete each task is.
- Sample Task: Update status of Task 1985743 to show that it is 90% complete.

Display:



Sequence of actions:

1. Click on field showing 1985743.
2. Click on field showing 60%.
3. Enter 90.
4. Click on Commit.

Some Design Principles

Input Basics:
Keyboard vs. Mouse vs. Joystick

Basics of Display Design

Icons vs. Words as Display Objects

Method Structure

Support for Learning

Error Recovery

Documentation and On-Line Help

Input Basics:
Keyboard vs. Mouse vs. Joystick

Keyboard control is fast, but often entails memorization, slow recall.

If user hesitates to remember command, speed advantage is gone.

- Keystroke takes 0.28 sec.
- Pause for recall can take at least 1.2 sec.

Considerable practice may be needed!

The mouse is the best choice for pointing to objects in arbitrary positions.

As long as objects are big enough.

Fitts' Law equation shows that pointing time is

- Less-than-linear with increasing distance.
- Worse than linear with decreasing size.

If properly designed and set up, a mouse is as fast as pointing with the finger.

Joysticks are poor cursor control devices; use only where they are game-realistic.

Pointing time is usually worse than Fitts' Law.

- Some serious problems have been documented.

But often the best choice for movement control of vehicles.

- E.g., use in an airplane simulation.

Basics of Display Design

What not to do: Focus on appearance.

Be concerned only with what the screens look like: screen layout, colors, icons, etc.

What to do: Focus on efficient information delivery.

Figure out what decisions the user will have to make at each point in the task.

Make sure that the information for each decision is on the screen when the decision is made.

Try to have all relevant information simultaneously available.

- Eye movements are extremely fast compared to bringing up additional screens of information.
- With good layout it is possible to effectively present much more information than generally realized.

Ensure that information can be brought up easily and quickly.

- Don't force the user to struggle just to get the displays necessary for a task.

Common design error:

Open an new window or dialog for every aspect of a task.

- Incorrect intuition: Must avoid cluttered display.
- Consequence: Cluttered procedures!

Better approaches:

Key information and controls always visible.

- Constrained by other needs for display space.

All information and controls for a task available at once.

- Relatively large or dense dialogs or windows.

Icons vs. Words as Display Objects

Current GUI trend: Use nifty icons, not boring words!

Fun, but misguided!

Icons are definitely better mouse targets than words

Square shape gives more target area

Words tend to be long and thin; often provide small target

Icons are often arbitrary and meaningless compared to typical words for computer objects.

Icons can be hard to recognize compared to words

- Especially if item concept is abstract

Why make the user memorize a meaningless symbol?

- *"There is a near-universal coding scheme that users know extremely well, with many years of practice – words!"*

Icons work best when closely resemble a concrete, familiar object commonly associated with the task

- Can be recognized from appearance and context
- E.g. a pencil icon in a drawing program

Method Structure

Users should be able to accomplish tasks using a small number of simple and efficient methods.

Every high frequency critical task goal should have a simple method available.

Such goals are the critical ones for performance, so method should be quick

Every goal should have only one method for accomplishing it.

Unless there are specific reasons for the multiple methods
Unnecessary methods just add opportunities for confusion and error

If there are multiple methods:

It should be possible to state a simple and clear selection rule for using each one
If not, the method should be eliminated

The same goal should always be accomplished by the same method.

Common "subroutines" of method knowledge
E.g. text selection in a text editor

Similar goals should have similar methods.

Must be highly similar at the detailed keystroke level

- E.g. copy vs. move of text in a text editor

Rough similarity, or visual similarity is NOT adequate
Similar enough: substitute one name or concept to get the other method

Support for Learning

Common belief: System is either easy to learn or easy to use, and can't be both.

Intuitive contrast between menu and command systems.

Fact: Tradeoff between ease of learning & speed of use is not mandatory.

Shown by several experiments.

Systems with fewer, simpler methods usually require both less learning and less time to execute.

A good interface design can support both new and experienced users within a single interface.

- E.g. PC: "BLT" menus, e.g. Lotus 1-2-3
  Using single-letter typeahead menu selection teaches a command language without effort.

Single biggest contribution to ease of learning: simple and consistent methods.

Careful choice of functionality and interface design can result in simple methods for tasks.

- Example: Deleting a directory on a Mac vs. PC-DOS.

Users can *transfer* procedure knowledge to different system or application, if procedures are similar enough.

- Secret of the Macintosh - most procedures transfer.
- Very powerful determinant of learning time.

Customization functions can increase learning costs

User has to also learn customization methods.

Should be a good default interface for using the system.

- Don't just provide a "kit" for building an application.
- Do not assume that users will be able to design a better interface than the developers could.

Can result in inconsistent (and undocumented) interfaces .

Error Recovery

User errors are a difficult design issue.

At this time, no good model for predicting *when* or *what* errors will occur.

- Although some progress could be made quickly

But GOMS model can help evaluate design quality given that an error has occurred.

- User now has the goal of "recover from error."
- What methods are available for accomplishing this goal?

User errors should be prevented if possible.

Incorrect method or operator can not be executed.

- Mac gray-out of menu items.

Confirmation checks at critical places.

- Do you want to save your work? - default is *yes.*

Error recovery should be routine.

Critical because 10 - 50% of operations are erroneous.

Universal method for backing out or canceling.

Allows user to recover routinely instead of problem-solve.

- UNDO operation.
- Standard cancel operation.

Error messages must supply or identify a method for recovery.

Allows user to read and execute instead of problem-solve.

- Explicit "user action" in error message manuals.

Permit user to simply retry or reenter command.

Easiest to just rerun method if error was trivial.
Negotiated error recovery entails more methods or problem solving.

Documentation and On-Line Help

Documentation should present all of the components of a GOMS model for the task.

Most documentation presents lists of operator descriptions.

Mostly a list of menu choices, commands.

Rarely presents methods.

- What sequences of commands to use to accomplish goals.

Rarely presents selection rules.

- Which of multiple methods is most suitable for a particular situation.

Instead:

- Table of contents and menu should match user's goals.
- Complete methods for goals should be provided.
- Selection rules for multiple methods should be provided.

Hierarchical form of methods works well with hypertext presentation.

User can read top-down, looking only at unfamiliar methods.