# EBU6305

# Interactive Media Design and Production

## Heuristic Analysis

Dr. Xianhui Cherry Che

With special thanks to Dr. Ethan Lau & Dr. Changjae Oh

Queen Mary
University of London

# Learning Objectives

- Understand basic requirements of user testing
- Able to apply the ten principles of heuristic analysis

# User Testing

- Never blame the user.
  - If you do, you probably don't understand your users well enough.
- If users do not perform as you have expected, it is your fault, not user's.
  - Understand how and why they got lost in your design, and find solutions to solve these problems.
- Be silent, and observe.
  - If you lead users by giving hints, the test is ruined.
  - Normal users will not receive these hints.
- Ethical approval is required.

# Sample Size for User Testing

- As a general rule, more participants makes for more reliable information, even if it is subjective.
  - Another example of empirical research:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

- There is a debate on whether large quantity of subjective opinions (wisdoms of the crowd) can become almost objective.
  - Debatable examples:

# Heuristic Evaluation

- Developed by Jakob Nielsen
- Helps find usability problems in a UI design
- Small set of evaluators used to examine UI (from 3 to 5)
  - Each checks for compliance with usability principles - *heuristics*
  - Use multiple evaluators as each will identify different problems
- At the end of session, problems are compiled and used to inform re-design
- “Discount” usability engineering method
  - One case study found factor of 48 in cost/benefit:
    - Cost of inspection: $10,500. Benefit: $500,000 (Nielsen, 1994)

# 10 Basic Principles

*From Nielsen's web page*

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and Documentation

**Essential Reading - 10 Usability Heuristics for User Interface Design**

# More Guidelines…

1) Things that look different should act different.
2) If it is not needed, it's not needed.
3) The information for the decision needs to be there when the decision is needed.
4) The user should control the system. The system shouldn't control the user. The user is the boss, and the system should show it.
5) The idea is to empower the user, not speed up the system.
6) Don't overload the user's buffers.
7) Keep it simple.
8) Things that look the same should act the same.
9) The user should be able to do what the user wants to do.
10) Every action should have a reaction.
11) Everything in its place, and a place for everything.
12) Let people shape the system to themselves, and paint it with their own personality
13) Error messages should actually mean something to the user, and tell the user how to fix the problem.
14) The best journey is the one with the fewest steps. Shorten the distance between the user and their goal.
15) Everyone makes mistakes, so every mistake should be fixable.
16) The more you do something, the easier it should be to do.

# More Guidelines…

17) Cute is not a good adjective for systems.
18) Keep it neat. Keep it organized.
19) Consistency, consistency, consistency.
20) The user should always know what is happening.
21) Minimize the need for a mighty memory.
22) Know they user, and YOU are not thy user.
23) If I made an error, at least let me finish my thought before I have to fix it.
24) Design for regular people and the real world.
25) Eliminate unnecessary decisions, and illuminate the rest.
26) You should always know how to find out what to do next.
27) If I made an error, let me know about it before I get into REAL trouble.
28) Even experts are novices at some point. Provide help.
29) Provide a way to bail out and start over.
30) Don't let people accidentally shoot themselves.
31) Color is information.
32) The user should be in a good mood when done.
33) The fault is not in thyself, but in thy system.
34) To know the system is to love it.
35) Deliver a model and stick to it.

# More Guidelines…

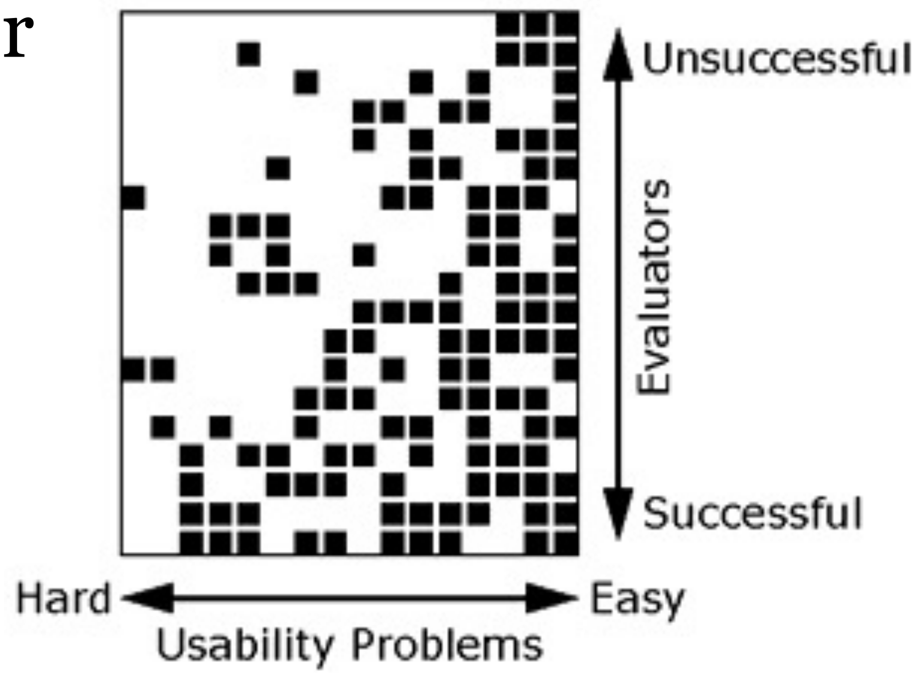
36) Follow platform conventions.
37) Make it hard for people to make errors.
38) The system status (i.e., what's going on should always be visible.
39) Accommodate individual differences among users through automatic adaptation or user tailoring of the interface.
40) Make it easy for a beginner to become an expert.
41) No you can't just explain it in the manual.
42) Provide user documentation that is easy to search, focused on the user's task, lists concrete steps to be carried out, and is not too large.
43) The system should speak the users' language, following real-world conventions.
44) Instructions for use of a system should be visible or easily retrievable.
45) What does marketing want? Ok, now how do we show them they're wrong?
46) What does management think it wants? Ok, now how do we show them they're wrong?
47) Allow users to tailor frequent actions.
48) Users don't know what they want, and users can't always say what they know.
49) Roll the videotape.
50) Common sense is an uncommon commodity
51) Make objects, actions, and options visible.

# More Guidelines…

52) Data drives good design.
53) Help users develop a conceptual representation of the structure of the system.
54) Minimize the amount of information a user must maintain in short-term memory.
55) It's a jungle. Be careful out there.
56) People should not have to remember information across a dialogue.
57) Make it impossible to make errors that will get the user into REAL trouble.
58) Dialogues should not contain information that is irrelevant or rarely needed.
59) Testing, testing, testing.
60) Keep the user mental workload within acceptable limits.
61) Minimize the amount of information recoding that is necessary.
62) Minimize the difference in dialogue both within and across interfaces.
63) An ounce of good design is worth a pound of technical support.
64) Provide the user with feedback and error-correction capabilities.
65) So how is this better than what the competition is doing?
66) Provide good error messages that are expressed in plain language, precisely indicate the problem, and constructively suggest a solution.
67) Whadya mean, they're not all computer scientists?
68) Support undo and redo.
69) Different words, situations, or actions should result in different things happening.
70) The best user interface is one the user doesn't notice.

# Evaluator Staffs

- Systematic inspection of system
- Multiple evaluators are better
- Trained evaluators are better
  - 22% vs. 41% vs. 60% of errors found
- Go through whole interface
- Result: list of problems, guidelines violated, and proposed fixes



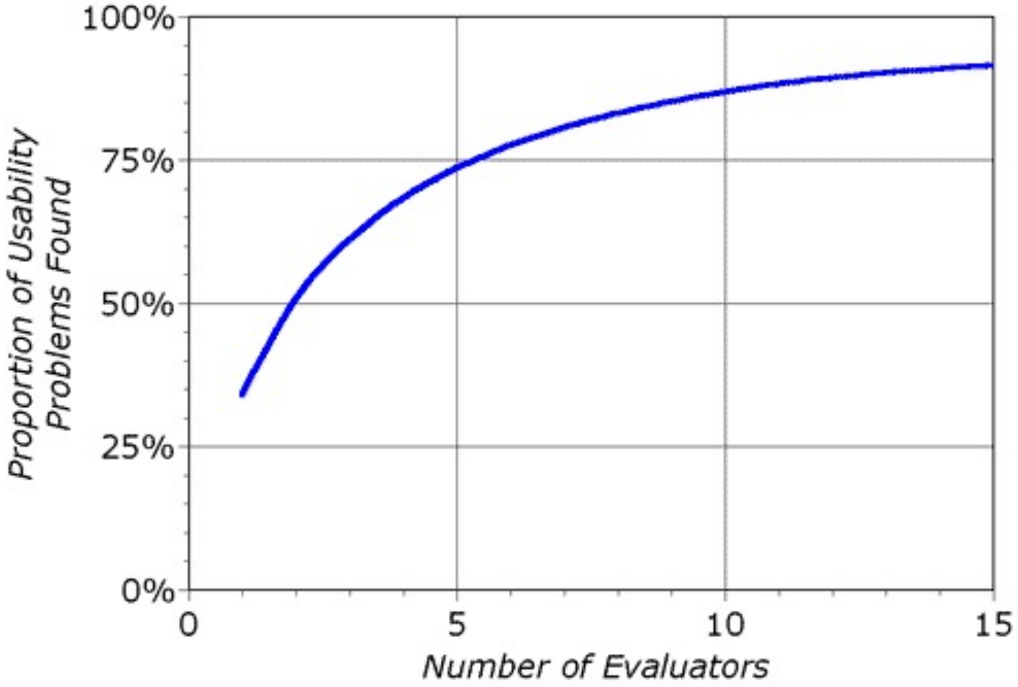
# Methodology

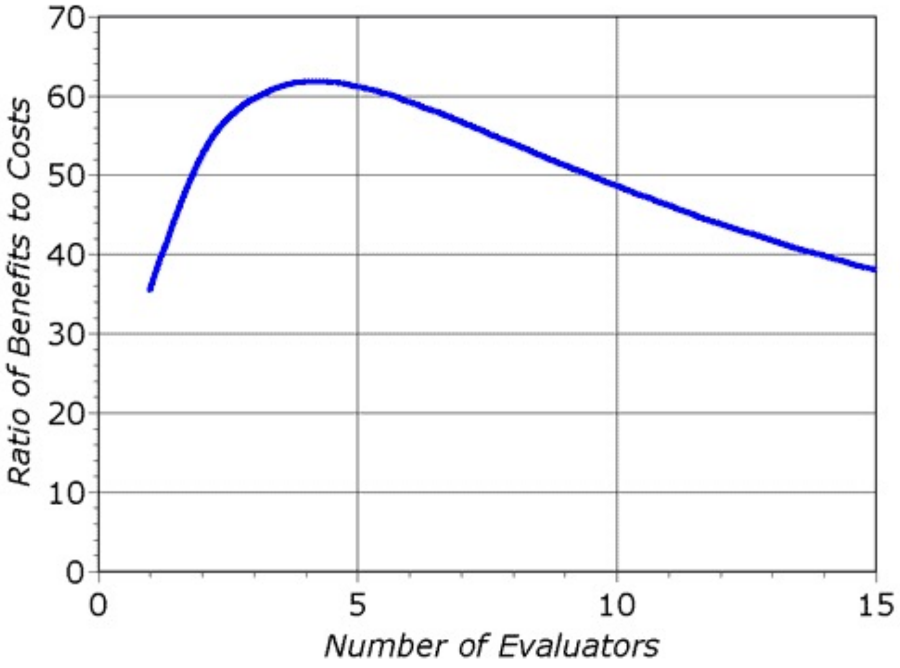
- Reference: Neilsen's "How to Conduct a Heuristic Evaluation"
  - Each evaluator inspects interface separately
  - OK for designer to answer evaluator's questions
  - Go through interface several times using heuristics
  - Can supply evaluators with scenarios of user tasks

**Background Reading - How to Conduct a Heuristic Evaluation**

# How Many Evaluators?

- Nielsen suggests optimal might be 4



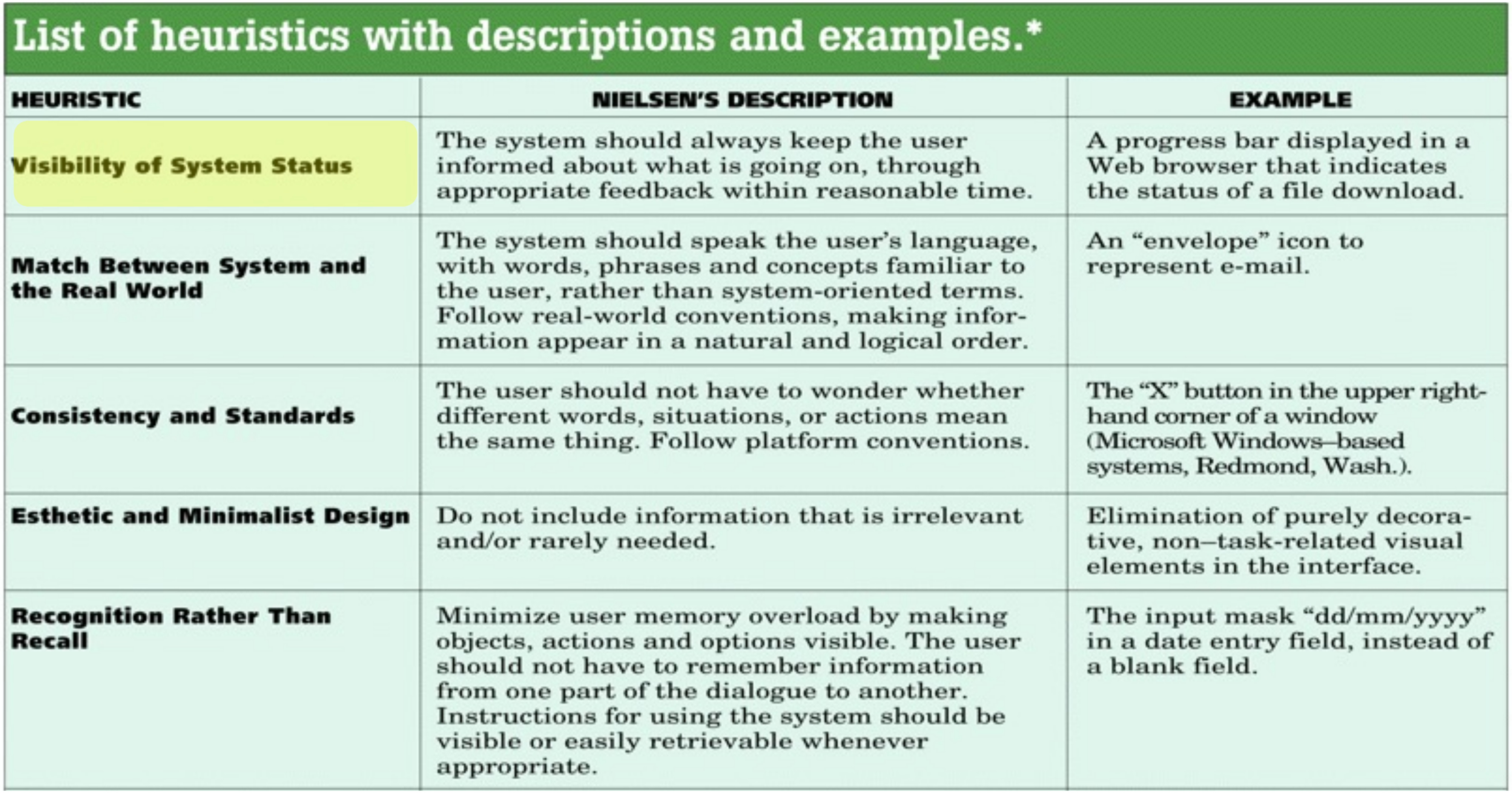


# Set of Usability Heuristics

- H1: visibility of system status
- H2: match between system and real world
- H3: user control and freedom
- H4: consistency & standards
- H5: error prevention
- H6: recognition rather than recall
- H7: flexibility and efficiency of use
- H8: aesthetic and minimalist design
- H9: help users recognize and recover from errors
- H10: help and documentation

# Usability Heuristics

- **Meet expectations**
  - (H2) Match the real world
  - (H4) Consistency & standards
  - (H10) Help & documentation
- **User focused**
  - (H1) Visibility of system status
  - (H3) User control & freedom
  - (H7) Flexibility & efficiency
- **Errors**
  - (H5) Error prevention
  - (H6) Recognition, not recall
  - (H9) Error reporting, diagnosis, and recovery
- **Keep it simple**
  - (H8) Aesthetic & minimalist design

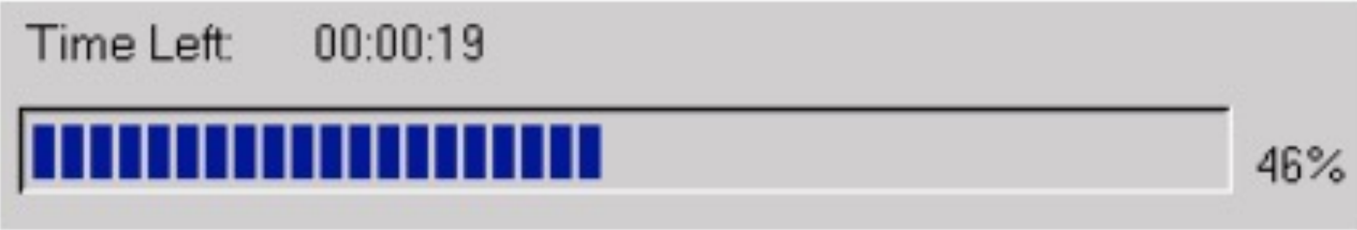


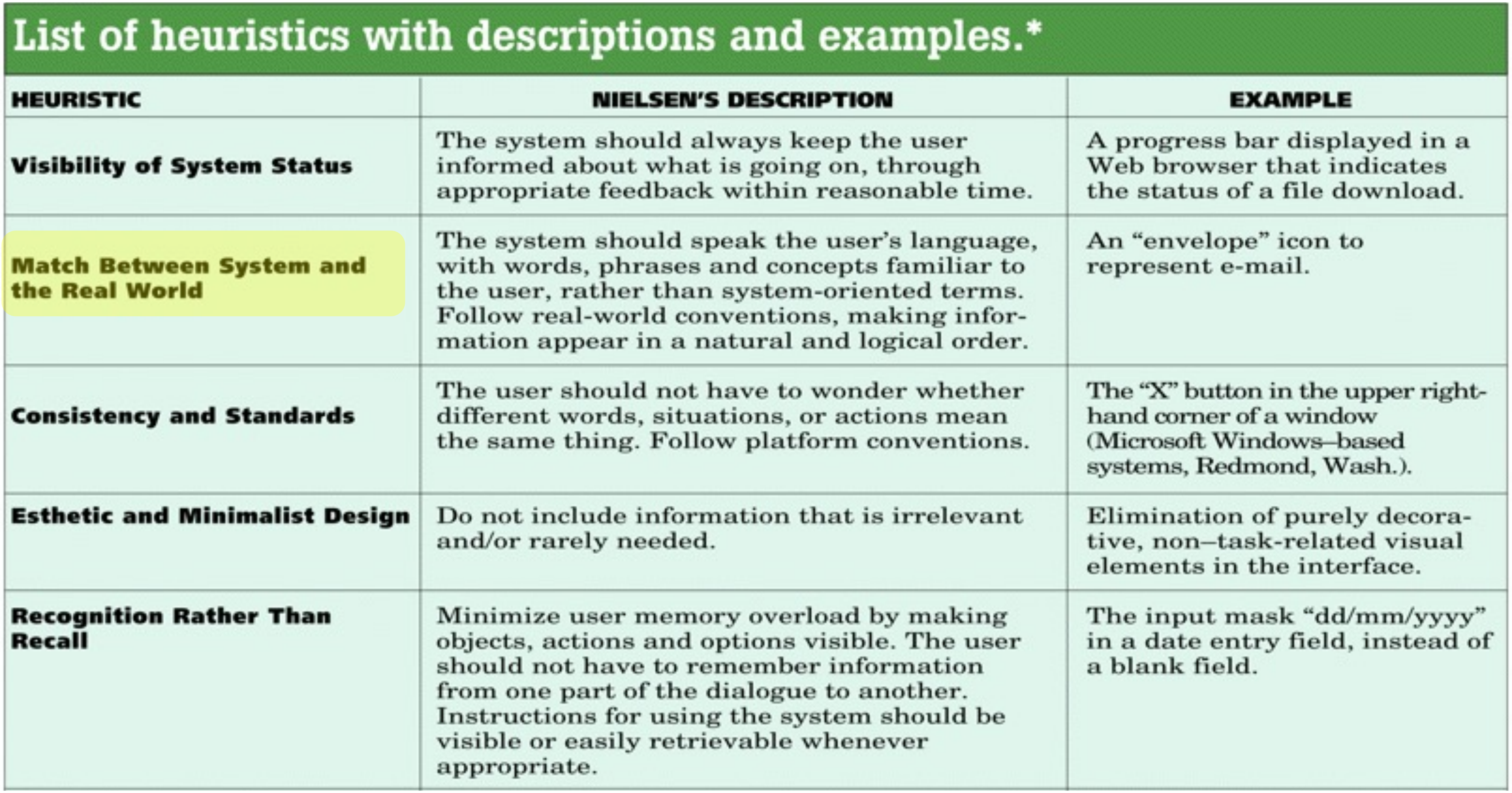
## List of heuristics with descriptions and examples.*

| HEURISTIC | NIELSEN'S DESCRIPTION | EXAMPLE |
|---|---|---|
| Visibility of System Status | The system should always keep the user informed about what is going on, through appropriate feedback within reasonable time. | A progress bar displayed in a Web browser that indicates the status of a file download. |
| Match Between System and the Real World | The system should speak the user's 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. | An "envelope" icon to represent e-mail. |
| Consistency and Standards | The user should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions. | The "X" button in the upper right-hand corner of a window (Microsoft Windows–based systems, Redmond, Wash.). |
| Esthetic and Minimalist Design | Do not include information that is irrelevant and/or rarely needed. | Elimination of purely decorative, non–task-related visual elements in the interface. |
| Recognition Rather Than Recall | Minimize user memory overload by making objects, actions and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for using the system should be visible or easily retrievable whenever appropriate. | The input mask "dd/mm/yyyy" in a date entry field, instead of a blank field. |

# 1. Visibility of System Status

- Keep users informed about what is going on
- Example: pay attention to response time
  - – 0.1 sec: no special indicators needed
  - – 1.0 sec: user tends to lose track of data
  - – 10 sec: max. duration if user to stay focused on 1 action
  - – for longer delays, use percent-done progress bars





List of heuristics with descriptions and examples.*

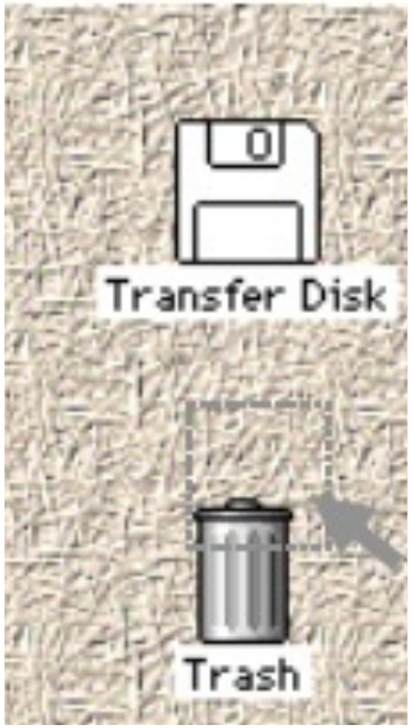
| HEURISTIC | NIELSEN'S DESCRIPTION | EXAMPLE |
|---|---|---|
| **Visibility of System Status** | The system should always keep the user informed about what is going on, through appropriate feedback within reasonable time. | A progress bar displayed in a Web browser that indicates the status of a file download. |
| **Match Between System and the Real World** | The system should speak the user's 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. | An "envelope" icon to represent e-mail. |
| **Consistency and Standards** | The user should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions. | The "X" button in the upper right-hand corner of a window (Microsoft Windows–based systems, Redmond, Wash.). |
| **Esthetic and Minimalist Design** | Do not include information that is irrelevant and/or rarely needed. | Elimination of purely decorative, non–task-related visual elements in the interface. |
| **Recognition Rather Than Recall** | Minimize user memory overload by making objects, actions and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for using the system should be visible or easily retrievable whenever appropriate. | The input mask "dd/mm/yyyy" in a date entry field, instead of a blank field. |

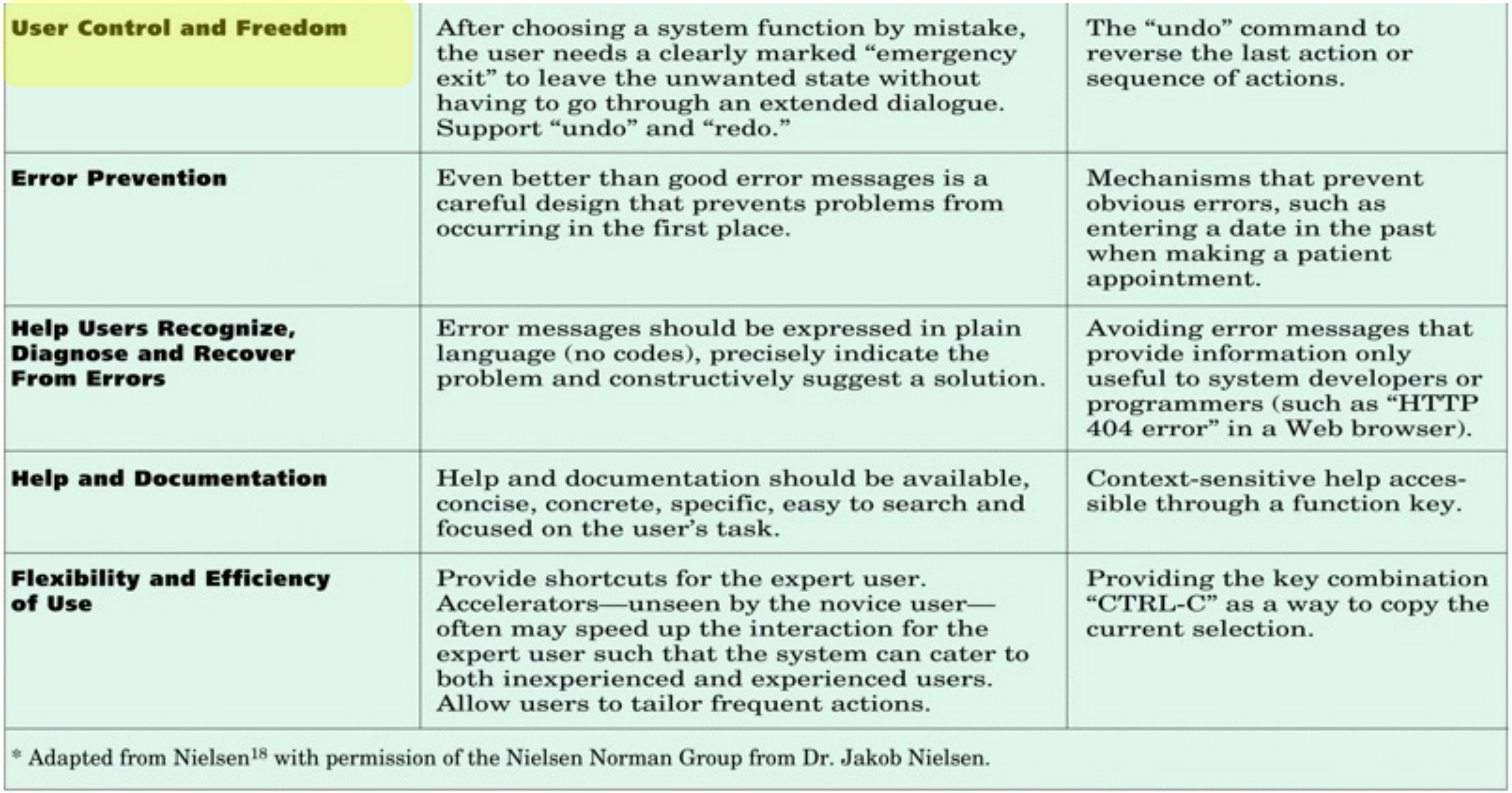
# 2. Match to Real World

- Speak the users' language
- Follow real world conventions

Mac desktop

– Dragging disk to trash should delete it, **not eject it**





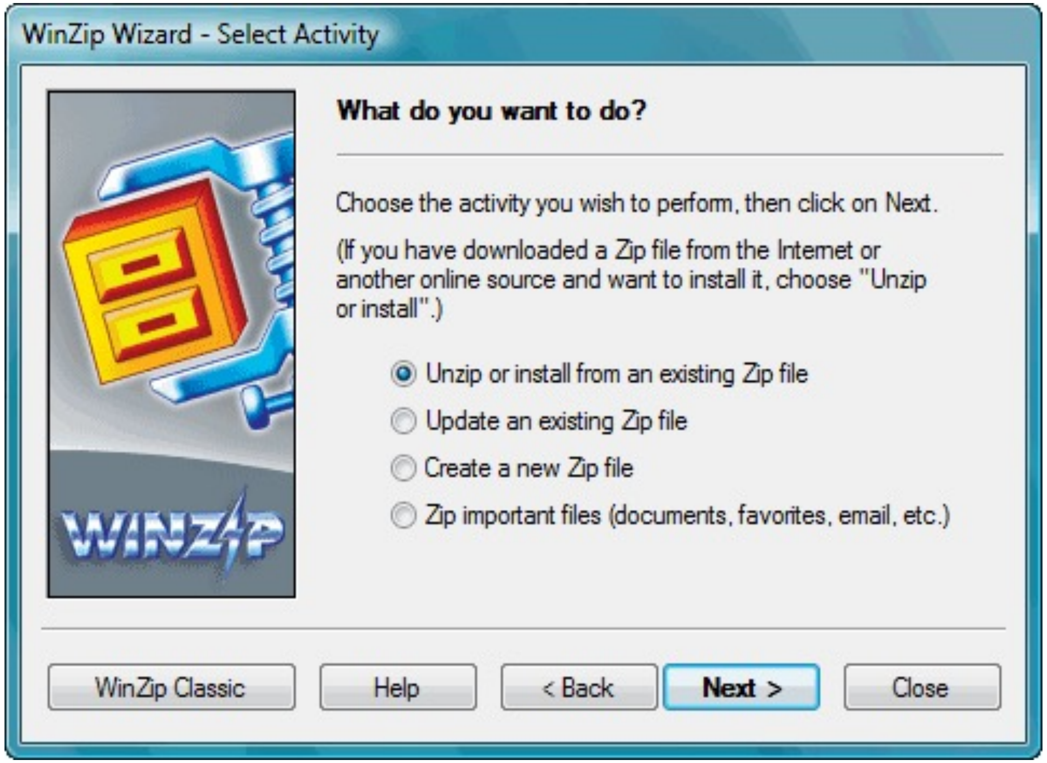
| | | |
|---|---|---|
| **User Control and Freedom** | After choosing a system function by mistake, the user needs a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support "undo" and "redo." | The "undo" command to reverse the last action or sequence of actions. |
| **Error Prevention** | Even better than good error messages is a careful design that prevents problems from occurring in the first place. | Mechanisms that prevent obvious errors, such as entering a date in the past when making a patient appointment. |
| **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. | Avoiding error messages that provide information only useful to system developers or programmers (such as "HTTP 404 error" in a Web browser). |
| **Help and Documentation** | Help and documentation should be available, concise, concrete, specific, easy to search and focused on the user's task. | Context-sensitive help accessible through a function key. |
| **Flexibility and Efficiency of Use** | Provide shortcuts for the expert user. Accelerators—unseen by the novice user—often may 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. | Providing the key combination "CTRL-C" as a way to copy the current selection. |

* Adapted from Nielsen[18] with permission of the Nielsen Norman Group from Dr. Jakob Nielsen.

# 3. User Control and Freedom

- Allow for mistaken choices, undo, redo
- Do not force down fixed paths

- *e.g. Wizards*
  - must respond to question before going to the next (forced down fixed path)
  - OK for infrequent tasks
    - *e.g. modem configuration*
  - not so good for common tasks
  - Good for beginners



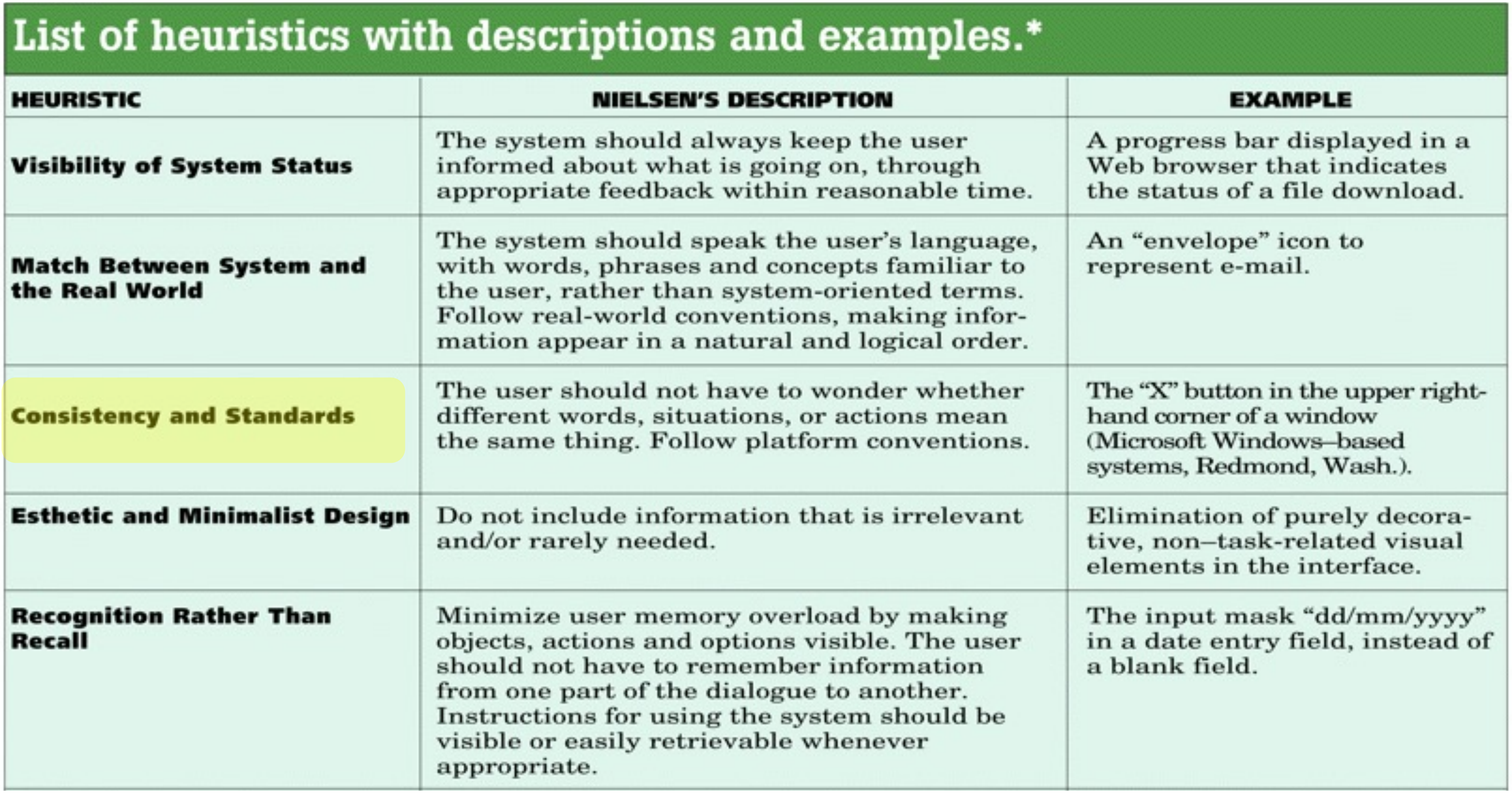
WinZip Wizard - Select Activity

**What do you want to do?**

Choose the activity you wish to perform, then click on Next.

(If you have downloaded a Zip file from the Internet or another online source and want to install it, choose "Unzip or install".)

● Unzip or install from an existing Zip file
○ Update an existing Zip file
○ Create a new Zip file
○ Zip important files (documents, favorites, email, etc.)

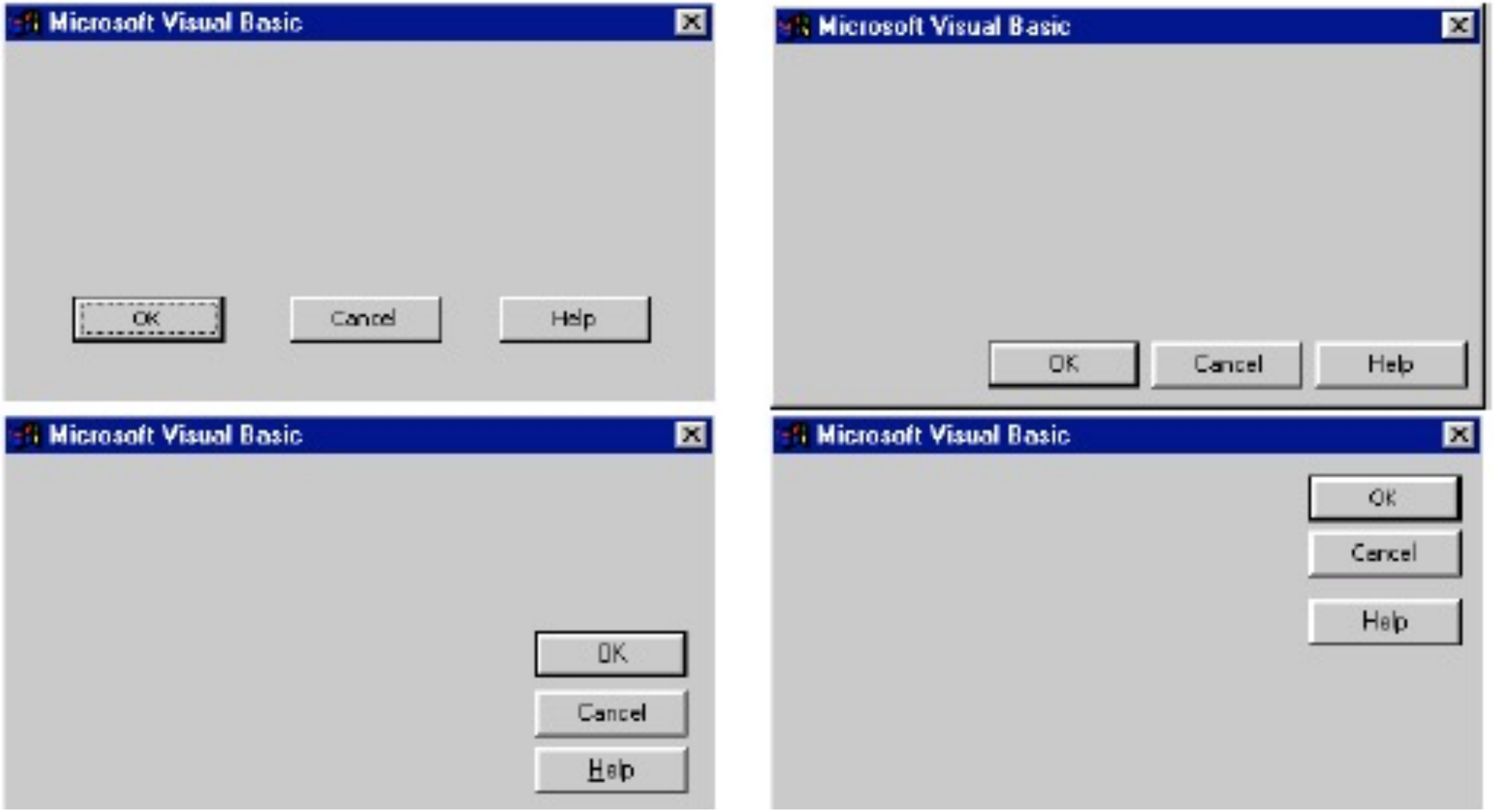


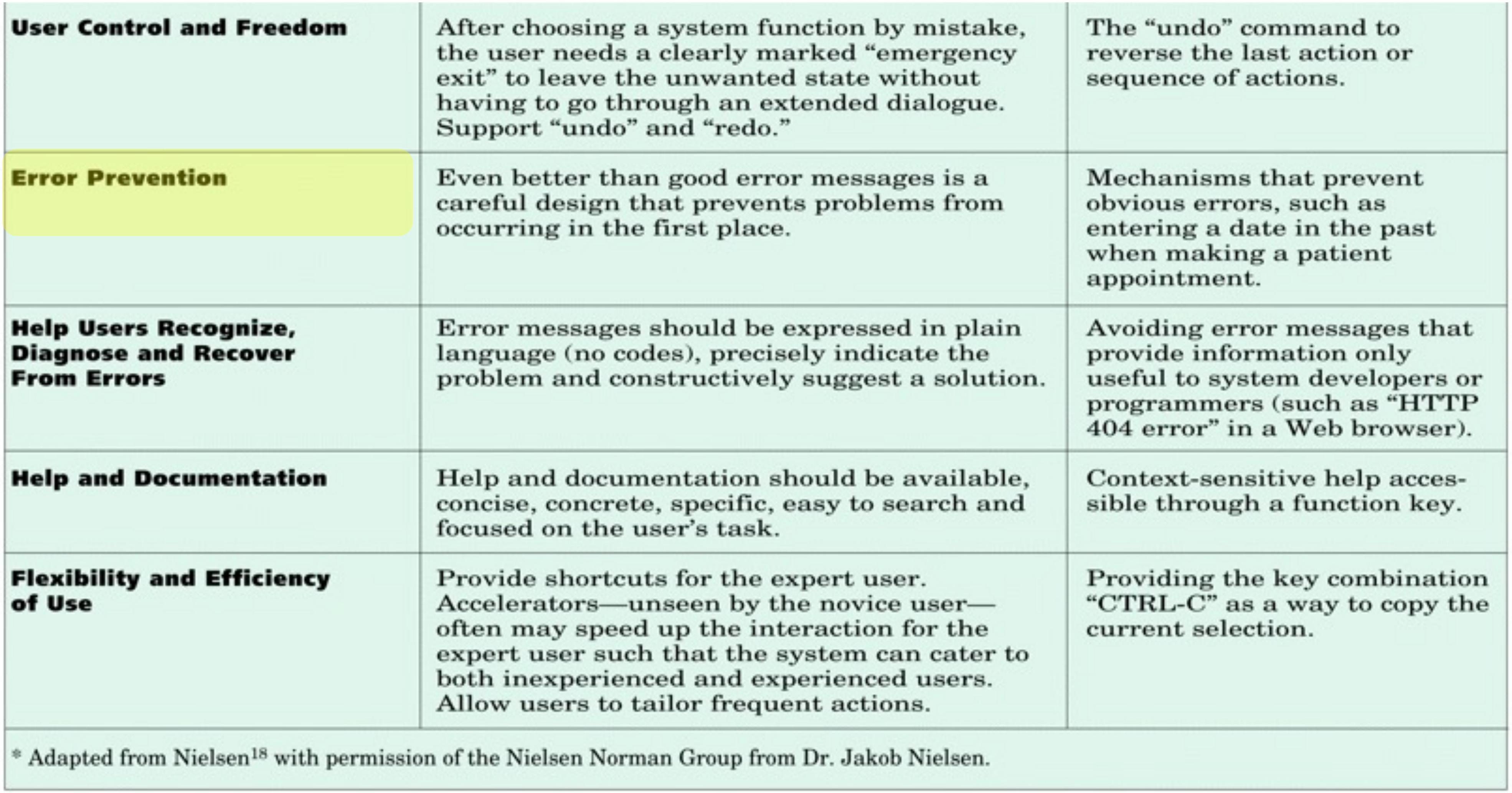
## List of heuristics with descriptions and examples.*

| HEURISTIC | NIELSEN'S DESCRIPTION | EXAMPLE |
|---|---|---|
| **Visibility of System Status** | The system should always keep the user informed about what is going on, through appropriate feedback within reasonable time. | A progress bar displayed in a Web browser that indicates the status of a file download. |
| **Match Between System and the Real World** | The system should speak the user's 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. | An "envelope" icon to represent e-mail. |
| **Consistency and Standards** | The user should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions. | The "X" button in the upper right-hand corner of a window (Microsoft Windows–based systems, Redmond, Wash.). |
| **Esthetic and Minimalist Design** | Do not include information that is irrelevant and/or rarely needed. | Elimination of purely decorative, non–task-related visual elements in the interface. |
| **Recognition Rather Than Recall** | Minimize user memory overload by making objects, actions and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for using the system should be visible or easily retrievable whenever appropriate. | The input mask "dd/mm/yyyy" in a date entry field, instead of a blank field. |

# 4. Consistency and Standards

- consistency within **and between applications**



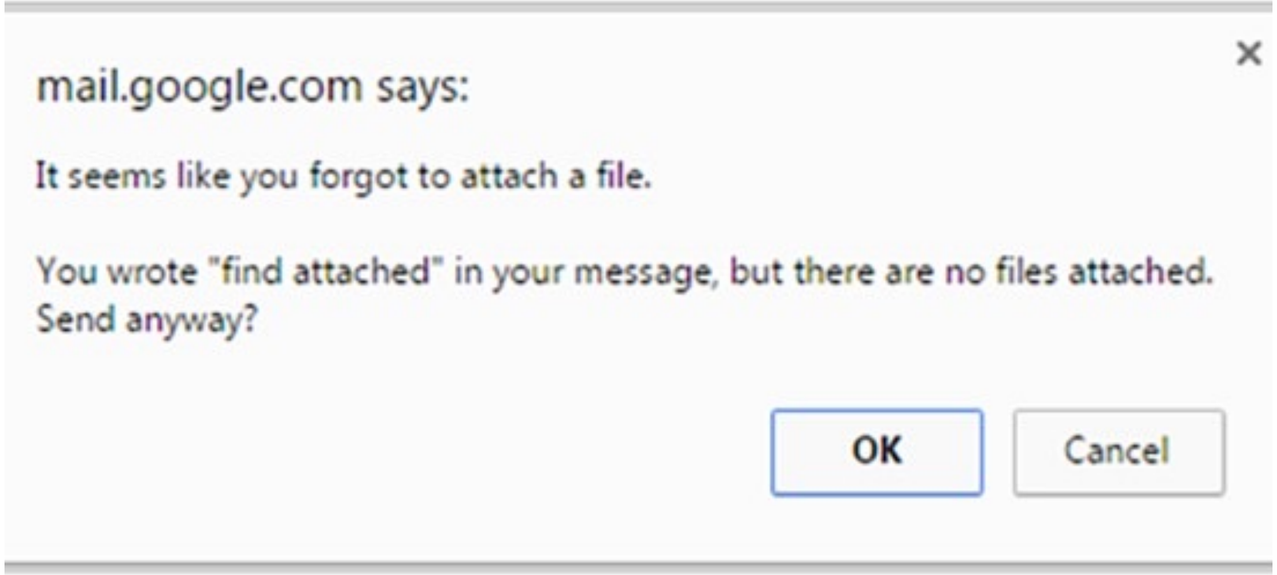


| | | |
|---|---|---|
| **User Control and Freedom** | After choosing a system function by mistake, the user needs a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support "undo" and "redo." | The "undo" command to reverse the last action or sequence of actions. |
| **Error Prevention** | Even better than good error messages is a careful design that prevents problems from occurring in the first place. | Mechanisms that prevent obvious errors, such as entering a date in the past when making a patient appointment. |
| **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. | Avoiding error messages that provide information only useful to system developers or programmers (such as "HTTP 404 error" in a Web browser). |
| **Help and Documentation** | Help and documentation should be available, concise, concrete, specific, easy to search and focused on the user's task. | Context-sensitive help accessible through a function key. |
| **Flexibility and Efficiency of Use** | Provide shortcuts for the expert user. Accelerators—unseen by the novice user—often may 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. | Providing the key combination "CTRL-C" as a way to copy the current selection. |

* Adapted from Nielsen[18] with permission of the Nielsen Norman Group from Dr. Jakob Nielsen.

# 5. Error Prevention

- *e.g. if PIN is 4 digits, only allow 4 numeric* characters to be entered,
- *e.g.* email attachment



mail.google.com says:

It seems like you forgot to attach a file.

You wrote "find attached" in your message, but there are no files attached. Send anyway?

# Consider the Following Usability Problem

***Occasionally, the web site does not allow you to go back to the previous page*****.**

Which of the five heuristics below characterises best this problem? (choose one only)

1. Visibility of system status
2. Match between system and real world
3. User control and freedom
4. Consistency and standards
5. Error prevention

# Consider the following usability problem

***Buttons have different labels on different pages, but they always perform the same sort of actions*.**

Which of the five heuristics below characterises best this problem? (Choose one only)

1. Visibility of system status
2. Match between system and real world
3. User control and freedom
4. Consistency and standards
5. Error prevention

# Consider the following usability problem

***It is not clear how to modify one’s address in “MyAccount” .***

Which of the five heuristics below characterises best this problem? (Choose one only)

1. Visibility of system status
2. Match between system and real world
3. User control and freedom
4. Consistency and standards
5. Error prevention

# Consider the following usability problem

***There is no reminder before the time is out.***

Which of the five heuristics below characterises best this problem? (Choose one only)

1. Visibility of system status
2. Match between system and real world
3. User control and freedom
4. Consistency and standards
5. Error prevention

# Consider the following statement

***Users shouldn’t need to wait more than 2 seconds for a system’s response.***

Which of the five heuristics below is related to this statement? (Choose one only)

1. Visibility of system status
2. Match between system and real world
3. User control and freedom
4. Consistency and standards
5. Error prevention

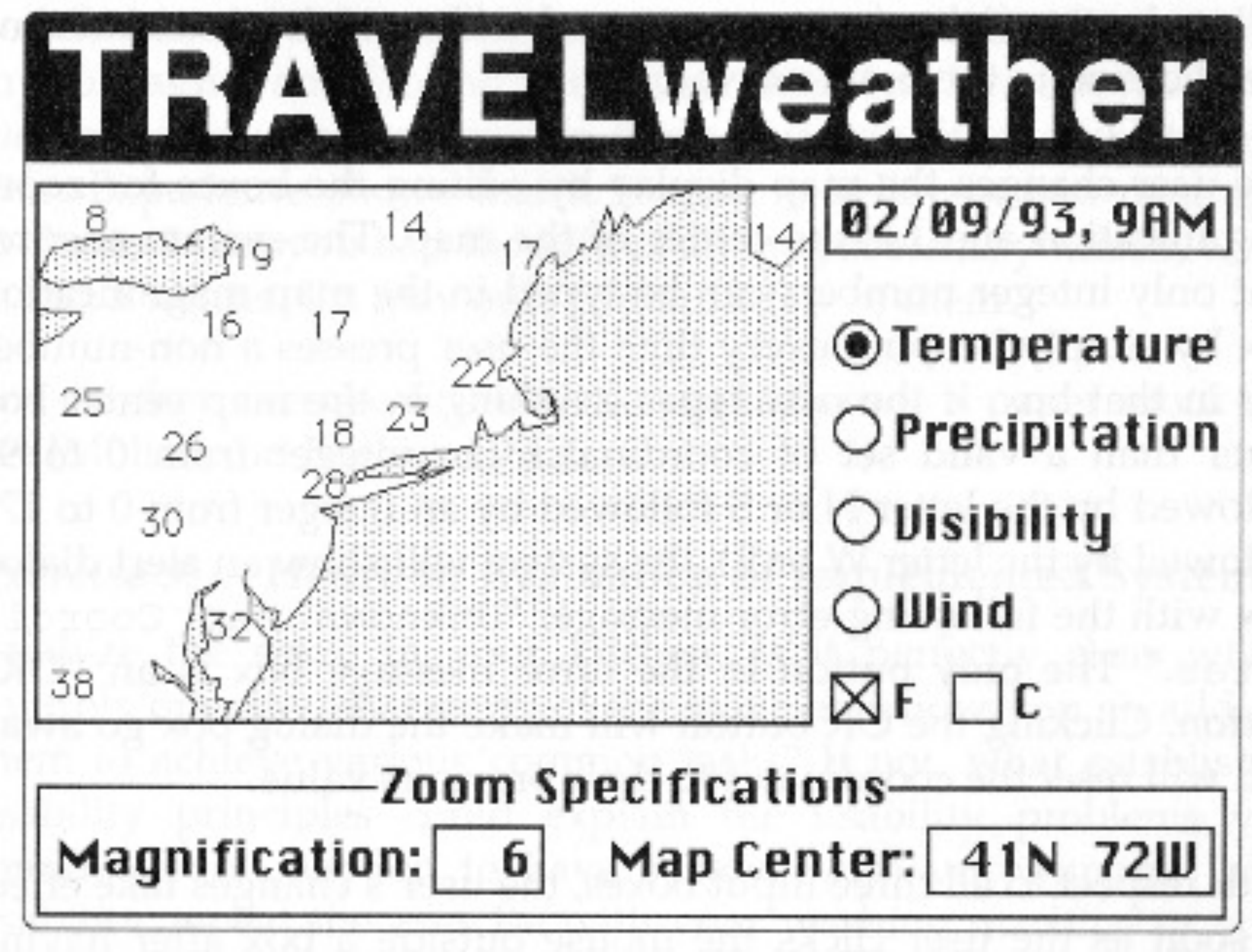
# Consider the following statement

***Users generally prefer systems similar to those they learned already and dislike unfamiliar systems.***

Which of the five heuristics below is related to this statement? (Choose one only)

1. Visibility of system status
2. Match between system and real world
3. User control and freedom
4. Consistency and standards
5. Error prevention

# Exercise



02/09/93, 9AM

● Temperature

○ Precipitation

○ Visibility

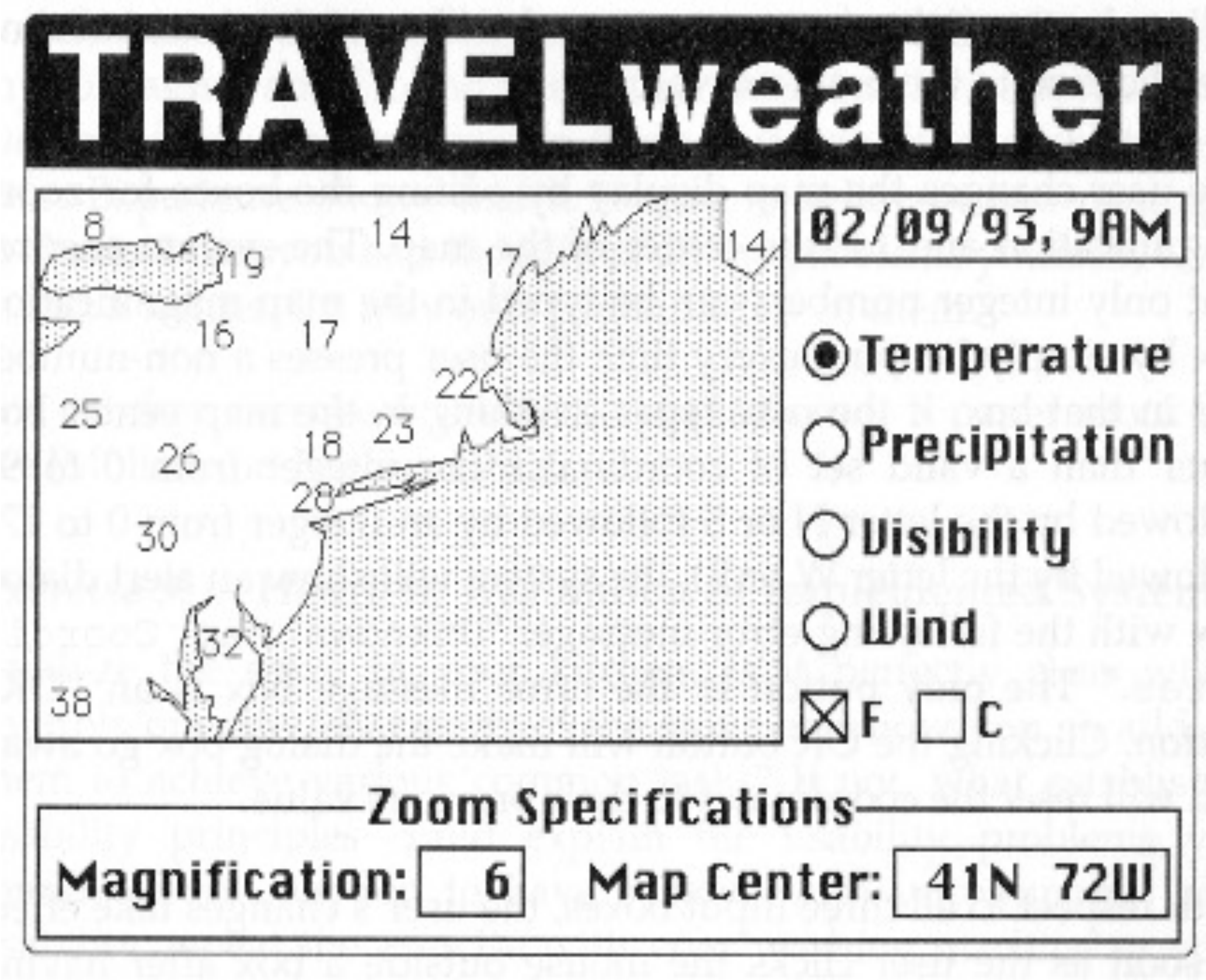
○ Wind

☒ F □ C

Zoom Specifications

Magnification: 6 Map Center: 41N 72W

# H1: Visibility of System Status?



TRAVELweather

02/09/93, 9AM

● Temperature

○ Precipitation

○ Visibility

○ Wind

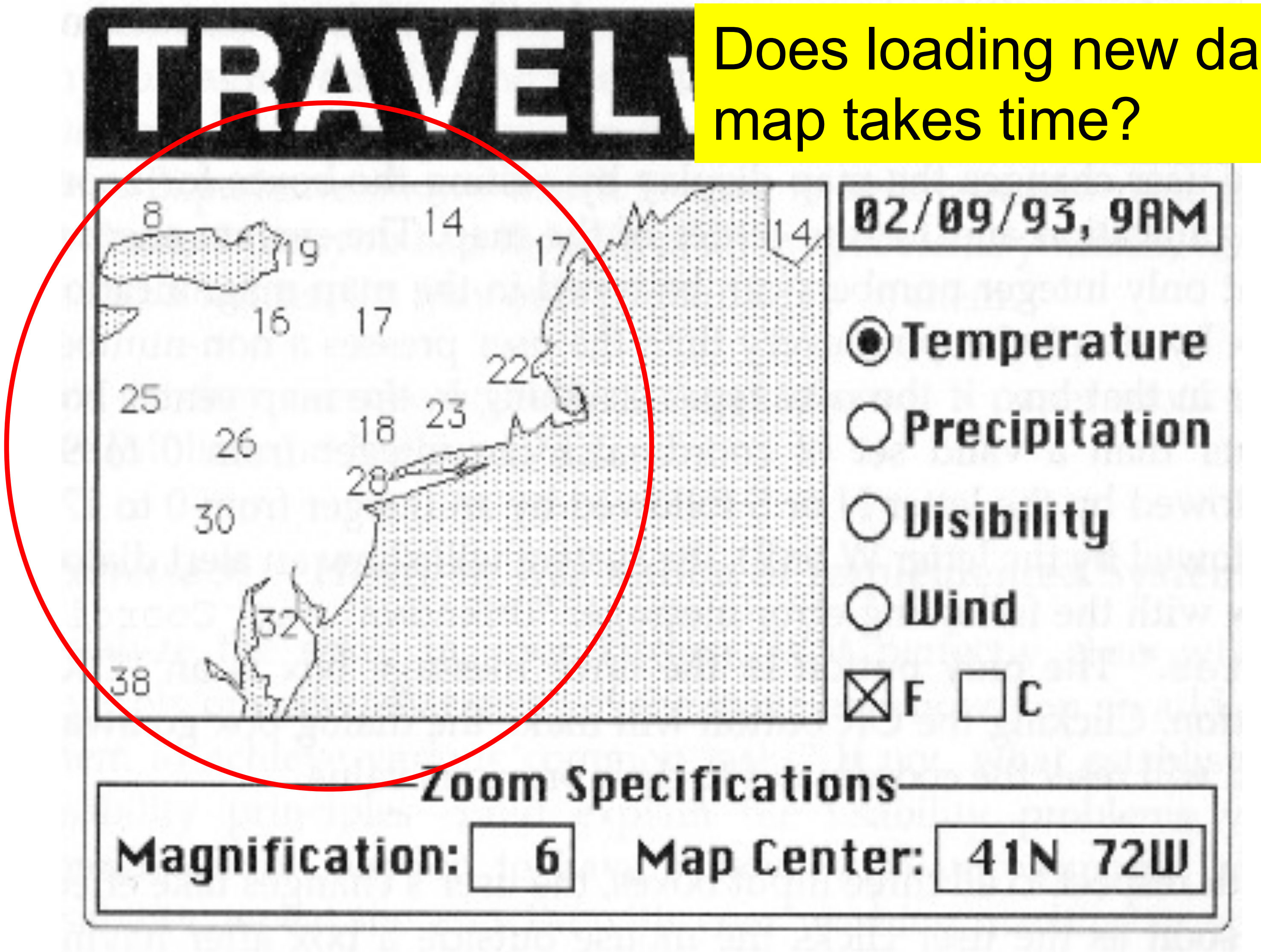
☒ F □ C

Zoom Specifications

Magnification: 6

Map Center: 41N 72W

# H1: Visibility of System Status?



Does loading new data onto the map takes time?

TRAVEL

02/09/93, 9AM

● Temperature

○ Precipitation

○ Visibility

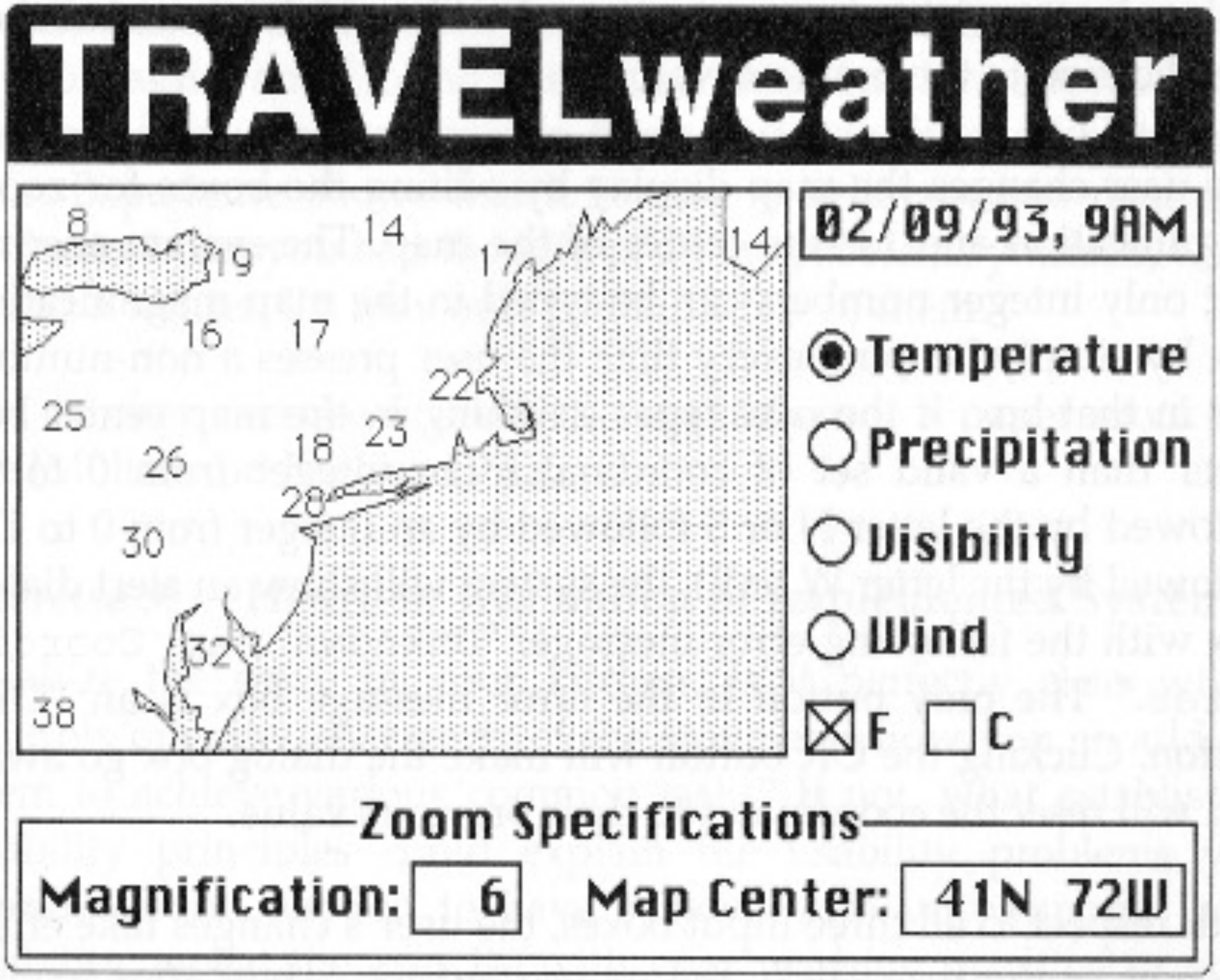
○ Wind

☒ F □ C

Zoom Specifications

Magnification: 6 Map Center: 41N 72W

# H2: Match Between System and Real World



TRAVELweather

02/09/93, 9AM

● Temperature

○ Precipitation

○ Visibility

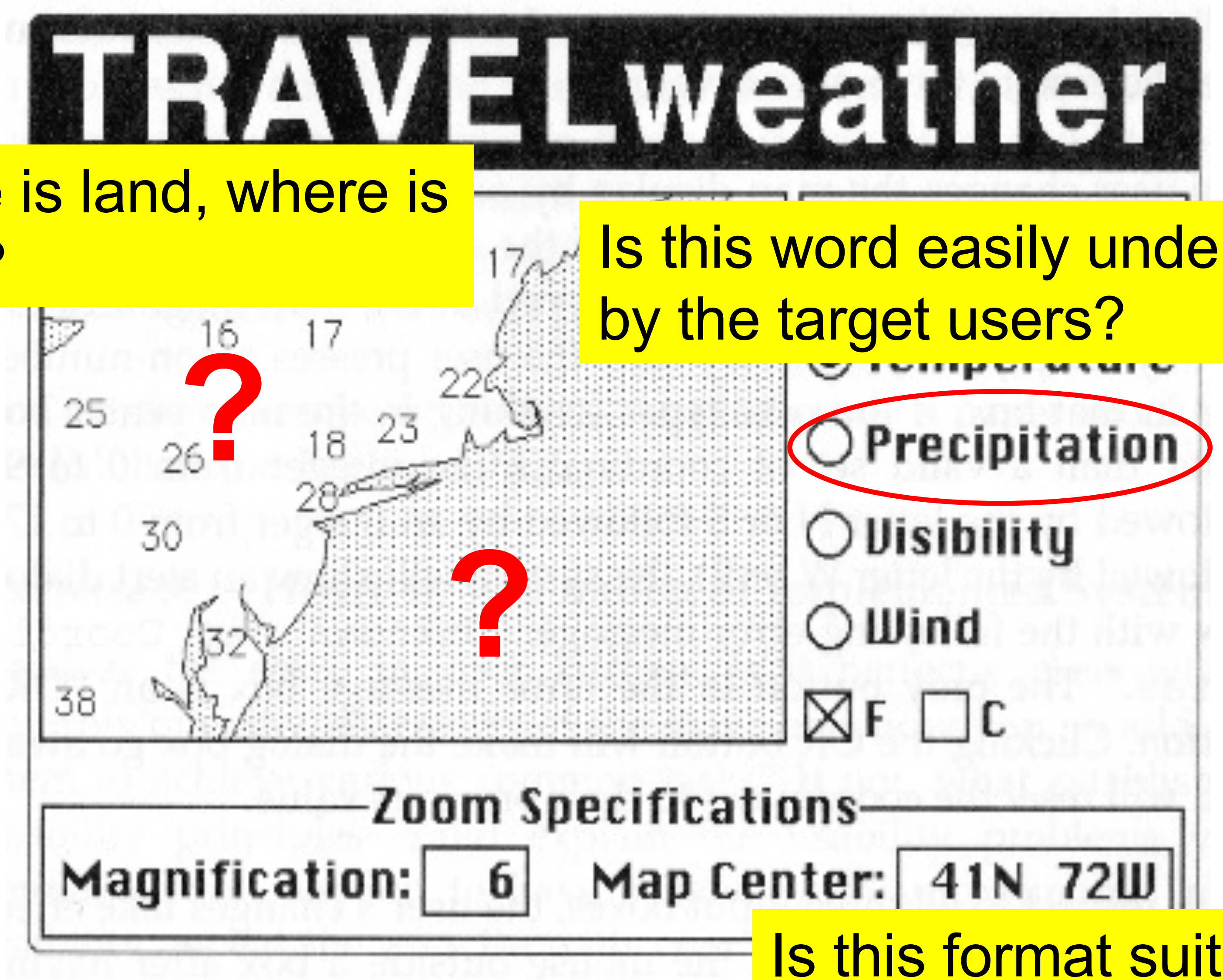
○ Wind

☒ F □ C

Zoom Specifications

Magnification: 6 Map Center: 41N 72W

# H2: Match Between System and Real World



TRAVELweather

○ Precipitation

○ Visibility

○ Wind

☒ F □ C

Zoom Specifications

Magnification: 6

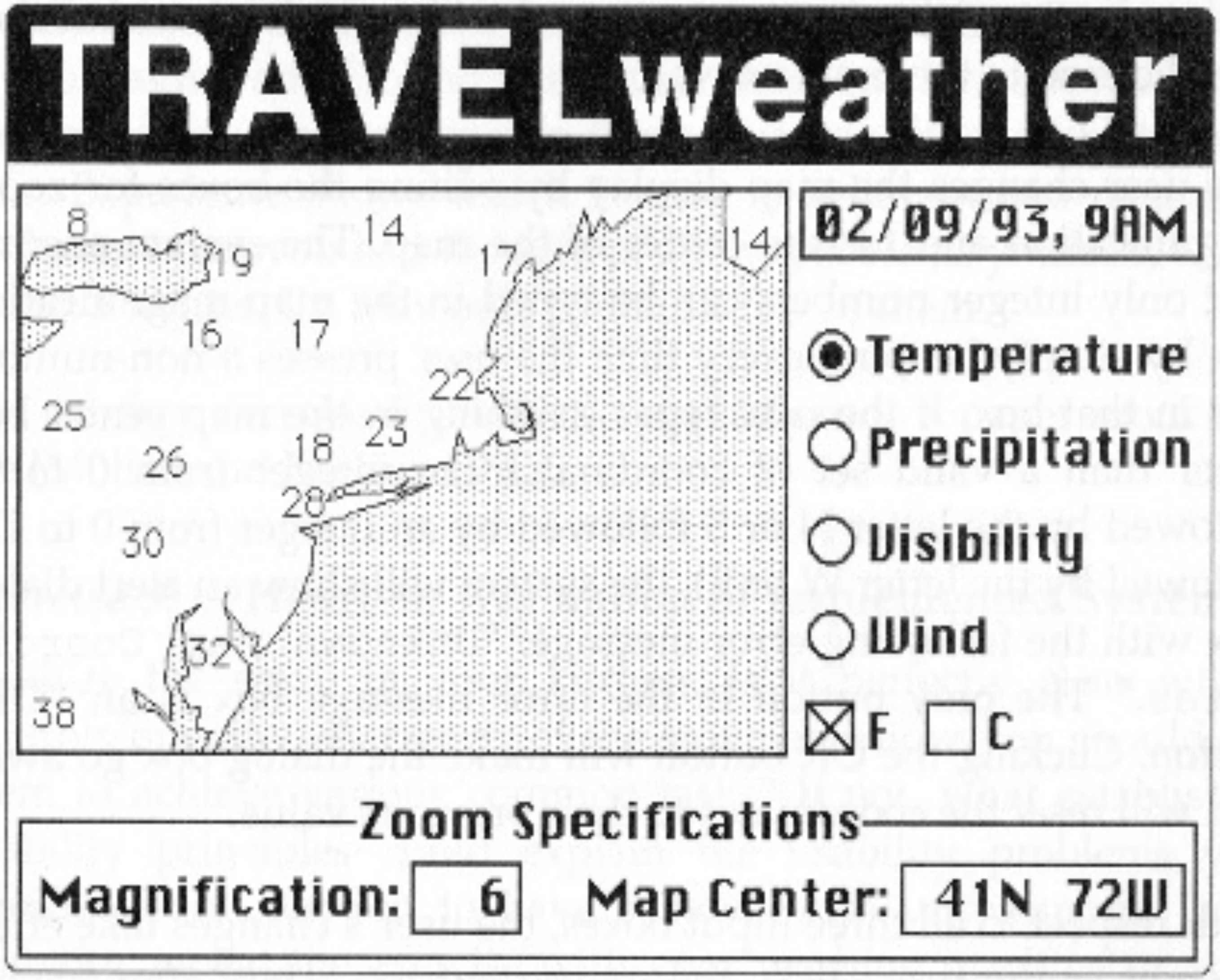
Map Center: 41N 72W

Where is land, where is water?

Is this word easily understandable by the target users?

Is this format suitable?

# H3: User Control and Freedom



TRAVELweather

02/09/93, 9AM

● Temperature

○ Precipitation

○ Visibility

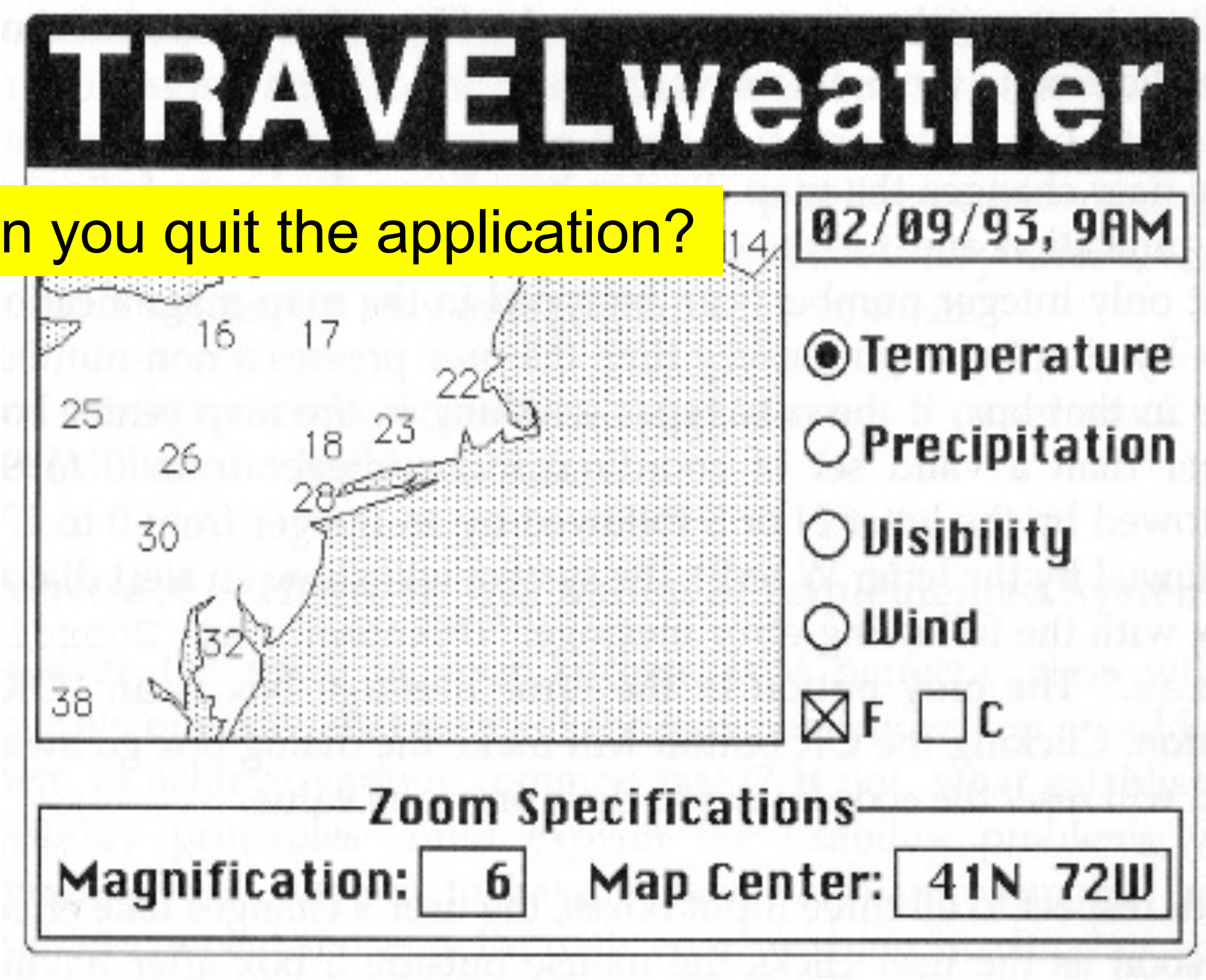
○ Wind

☒ F □ C

Zoom Specifications

Magnification: 6 Map Center: 41N 72W

# H3: User Control and Freedom



How can you quit the application?

TRAVELweather

02/09/93, 9AM

● Temperature
○ Precipitation
○ Visibility
○ Wind

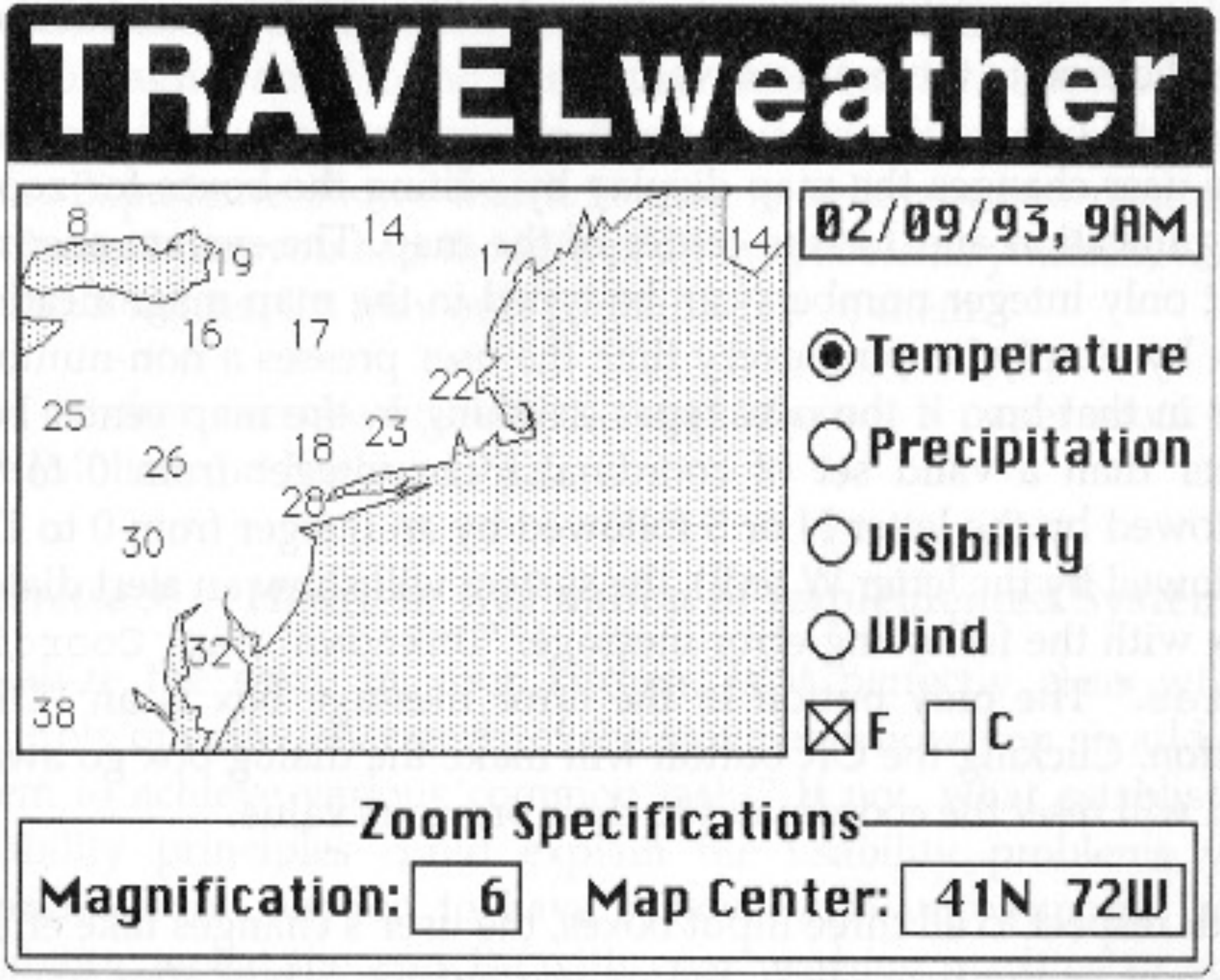
☒ F □ C

Zoom Specifications

Magnification: 6

Map Center: 41N 72W

# H4: Consistency & Standards



TRAVELweather

02/09/93, 9AM

● Temperature

○ Precipitation

○ Visibility

○ Wind

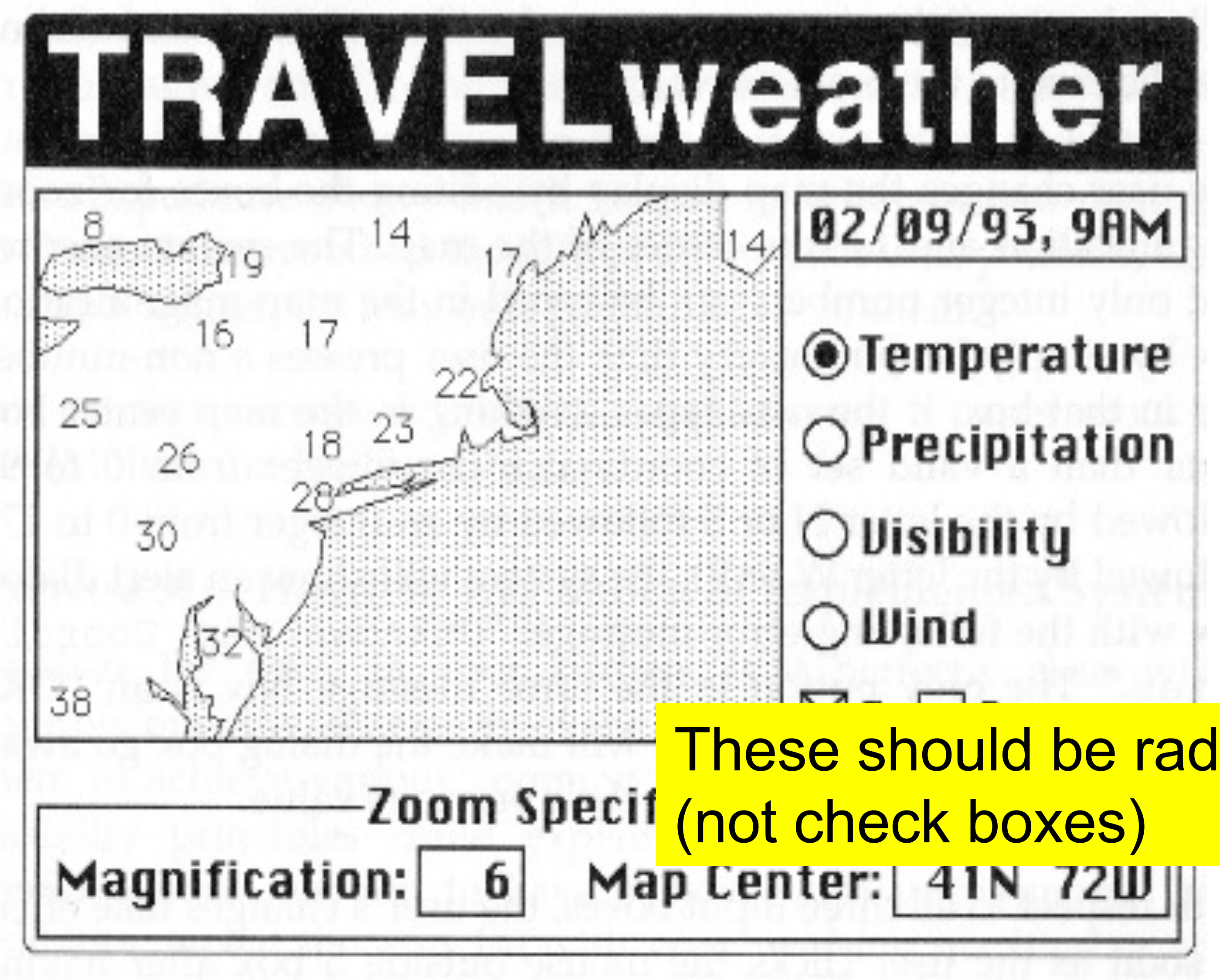
☒ F □ C

Zoom Specifications

Magnification: 6

Map Center: 41N 72W

# H4: Consistency & Standards



TRAVELweather

02/09/93, 9AM

● Temperature

○ Precipitation

○ Visibility

○ Wind

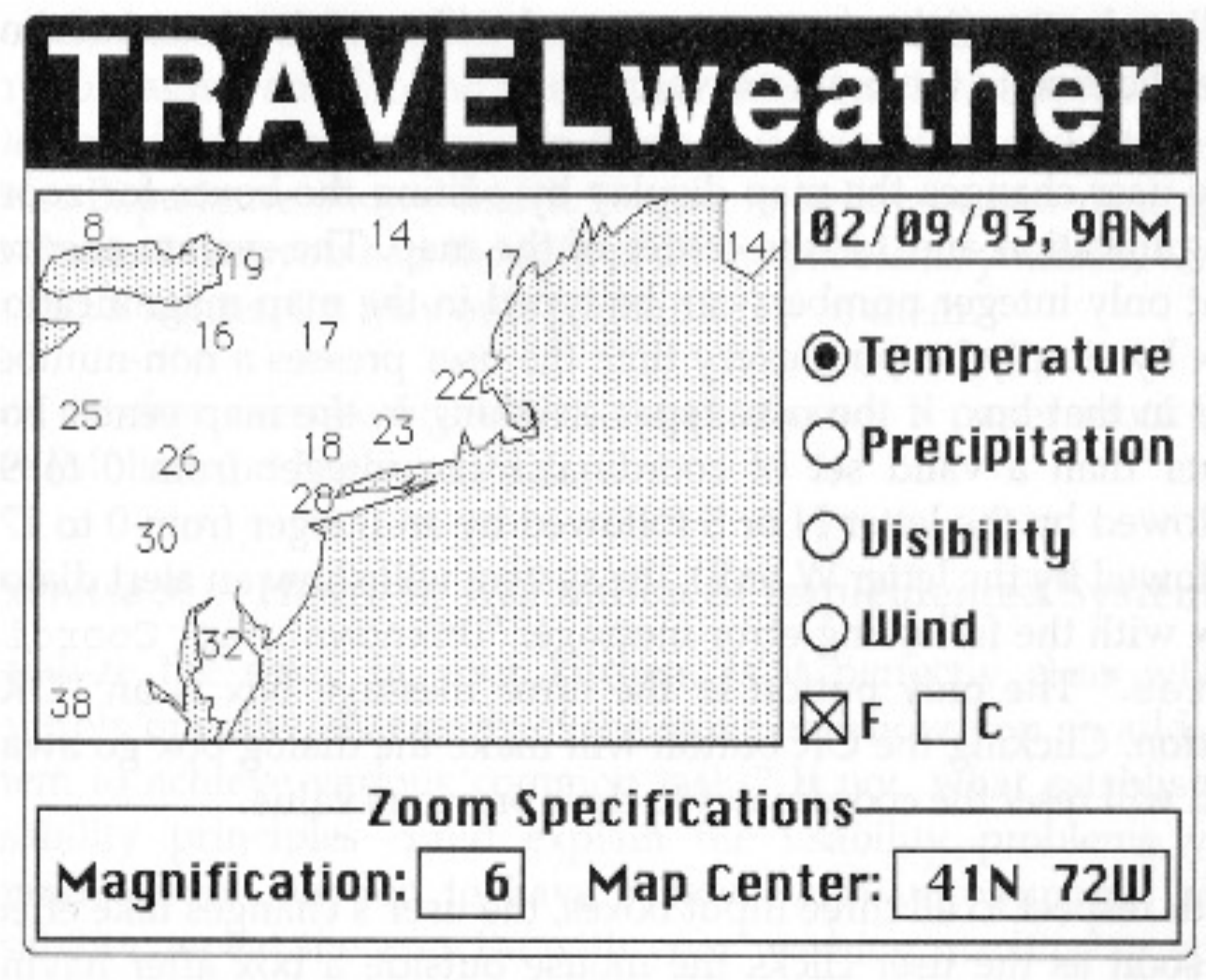
Zoom Specif

Magnification: 6

Map Center: 41N 72W

These should be radio buttons (not check boxes)

# H5: Error Prevention



TRAVELweather

02/09/93, 9AM

● Temperature

○ Precipitation

○ Visibility

○ Wind

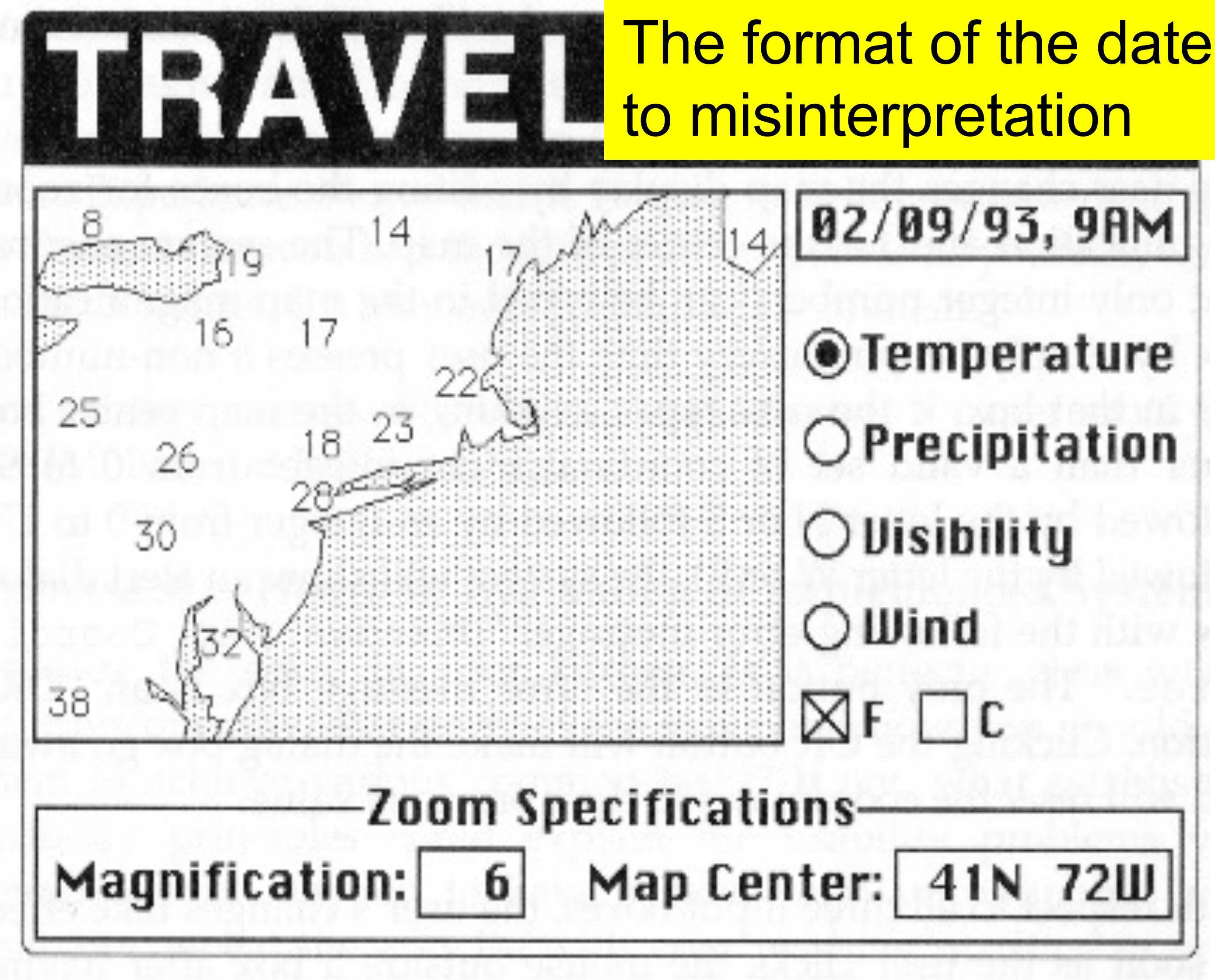
☒ F □ C

Zoom Specifications

Magnification: 6

Map Center: 41N 72W

# H5: Error Prevention



TRAVEL

02/09/93, 9AM

● Temperature

○ Precipitation

○ Visibility

○ Wind

☒ F □ C

Zoom Specifications

Magnification: 6

Map Center: 41N 72W

The format of the date is prone to misinterpretation

# Questions?

x.che@qmul.ac.uk