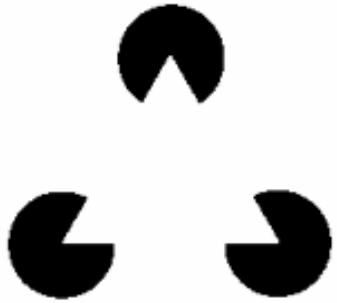


Gestalt psykologi



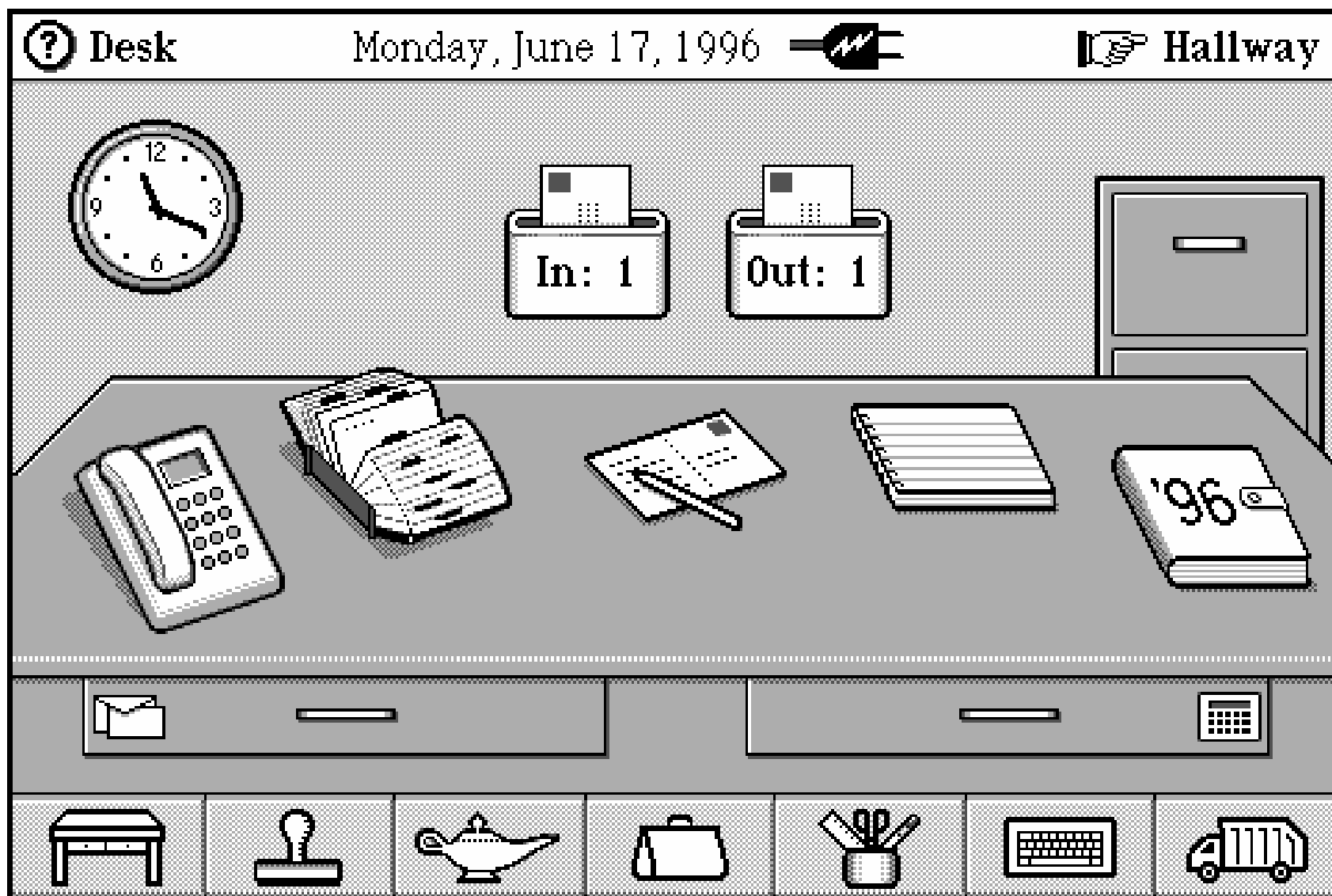
Tekst interface kunst

**Paul Vanouse:
"Persistent Data Confidante"
Online-projekt (ca. 1998)**

<http://pdc.walkerart.org/>

Interface til et interface

Familiar physical settings vs denial of the computer world



Metafor? Simulacrum?



Microsoft Bob

Jef Raskin's "Interface laws"

§ 1.

A computer shall not harm your work or, through inaction, allow your work to come to harm

§2.

A computer shall not waste your time or require you to do more work than is strictly necessary.

Jakob Nielsen: "10 Usability Heuristics"

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.

Recognition rather than recall

Minimize the user's memory load 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 use of the system should be visible or easily retrievable whenever appropriate.

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.

Ben Shneiderman: "8 Golden Rules"

1 Strive for consistency.

Consistent sequences of actions should be required in similar situations; identical terminology should be used in prompts, menus, and help screens; and consistent commands should be employed throughout.

2 Enable frequent users to use shortcuts.

As the frequency of use increases, so do the user's desires to reduce the number of interactions and to increase the pace of interaction. Abbreviations, function keys, hidden commands, and macro facilities are very helpful to an expert user.

3 Offer informative feedback.

For every operator action, there should be some system feedback. For frequent and minor actions, the response can be modest, while for infrequent and major actions, the response should be more substantial.

4 Design dialog to yield closure.

Sequences of actions should be organized into groups with a beginning, middle, and end. The informative feedback at the completion of a group of actions gives the operators the satisfaction of accomplishment, a sense of relief, the signal to drop contingency plans and options from their minds, and an indication that the way is clear to prepare for the next group of actions.

5 Offer simple error handling.

As much as possible, design the system so the user cannot make a serious error. If an error is made, the system should be able to detect the error and offer simple, comprehensible mechanisms for handling the error.

6 Permit easy reversal of actions.

This feature relieves anxiety, since the user knows that errors can be undone; it thus encourages exploration of unfamiliar options. The units of reversibility may be a single action, a data entry, or a complete group of actions.

7 Support internal locus of control.

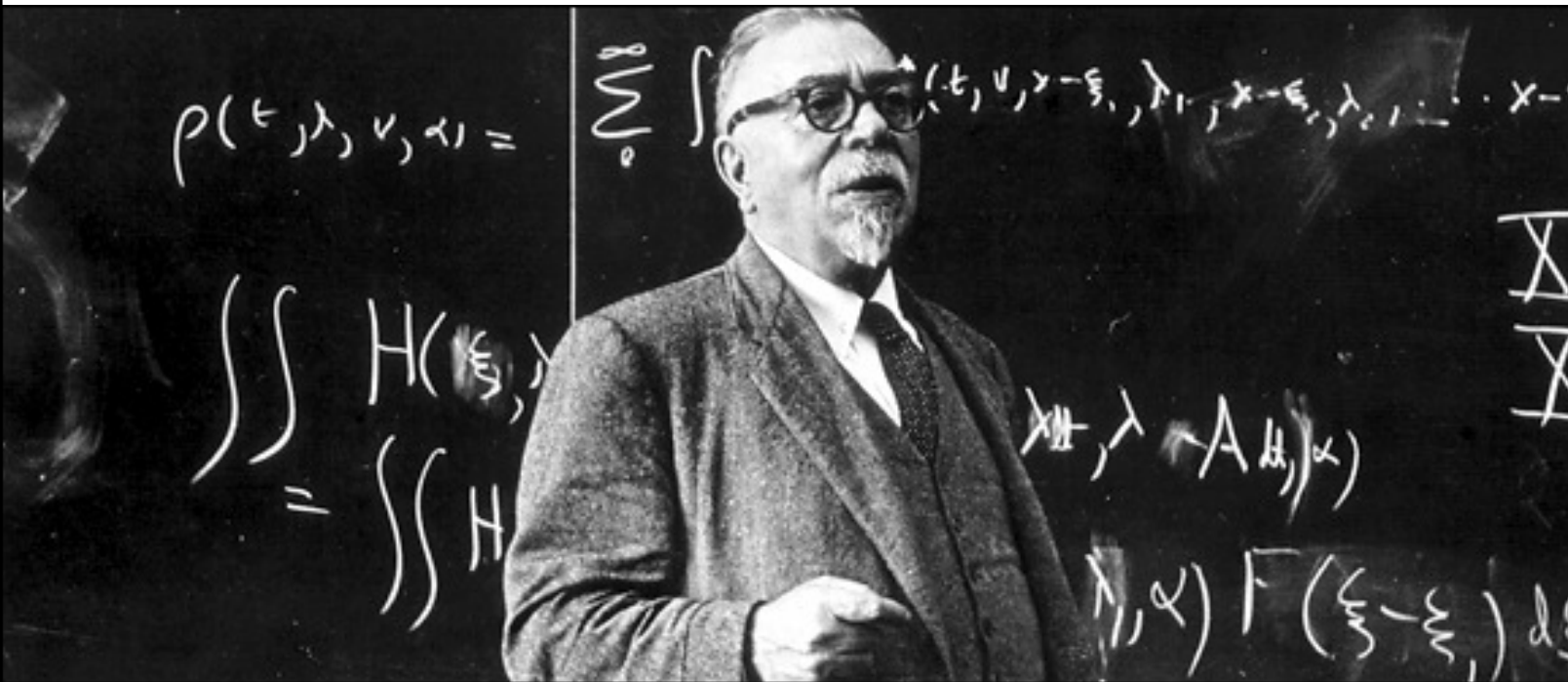
Experienced operators strongly desire the sense that they are in charge of the system and that the system responds to their actions. Design the system to make users the initiators of actions rather than the responders.

8 Reduce short-term memory load.

The limitation of human information processing in short-term memory requires that displays be kept simple, multiple page displays be consolidated, window-motion frequency be reduced, and sufficient training time be allotted for codes, mnemonics, and sequences of actions.

Kybernetik

Cybernetics is the study of communication and control in living beings and machines, and in combinations of the two.



Norbert Wiener: "Cybernetics or Control and Communication in the Animal and the Machine" (1948)

Hva' så med spil?



Processing Kode

Processing: Løkker/gentagelser

```
for (int i=0; i<10; i=i+1)  
{  
  kodeblok;  
}
```

Variabelnavn
og startværdi

Kontrolbetingelse

Ændring
pr. gennemløb

Koden kan læses sådan:

Gentag kodeblokken og tæl med variabelen "i".

Start med værdien 0 og gentag så længe i er mindre end 10.

Læg 1 til variabelen for hvert gennemløb.

Løkker

```
void draw()  
{  
    background(100,100,0);  
  
    for (int abc = 0; abc < width; abc = abc +30)  
        {  
            line (abc ,0, mouseX, mouseY);  
        }  
}
```

Processing: Betingelser

```
if(x < 100)  
{  
  kodeblok;  
}
```

Koden kan læses sådan:
Udfør kodeblokken hvis variablen x er mindre end 100.

Processing: Betingelser

- x == 10 Hvis x er lig 10
- x != 10 Hvis x er forskellig fra 10
- x > 10 Større end 10
- x >= 10 Større eller lig 10
- x < 10 Mindre end 10
- x <= 10 Mindre eller lig 10

Løkker + betingelse

```
void draw()
{
    background(100,100,0);

    for (int abc = 0; abc < width; abc = abc +30)
        {
            if(mouseY>50)
                {
                    line (abc ,0, mouseX, mouseY);
                }
        }
}
```

Betingelser/Booleske operatører

```
if((i > 35) && (i < 60))
```

Hvis i er større end 35 **og** mindre end 60

```
if((i > 35) && (x = 5))
```

Hvis i er større end 35 **og** x er 5

```
if((i < 35) || (i > 60))
```

Hvis i er mindre end 35 **eller** større end 60

Løkker + betingelse + booleske op.

```
void draw()
{
    background(100,100,0);

    for (int abc = 0; abc < width; abc = abc +30)
        {
            if((mouseY>50) && (?????))
                {
                    line (abc ,0, mouseX, mouseY);
                }
        }
}
```