NN/g Nielsen Norman Group
Evidence-Based User Experience Research, Training, and Consulting

Direct Manipulation: Definition

by Samyukta Sherugar and Raluca Budiu on August 21, 2016
Topics: Human Computer Interaction Application Design

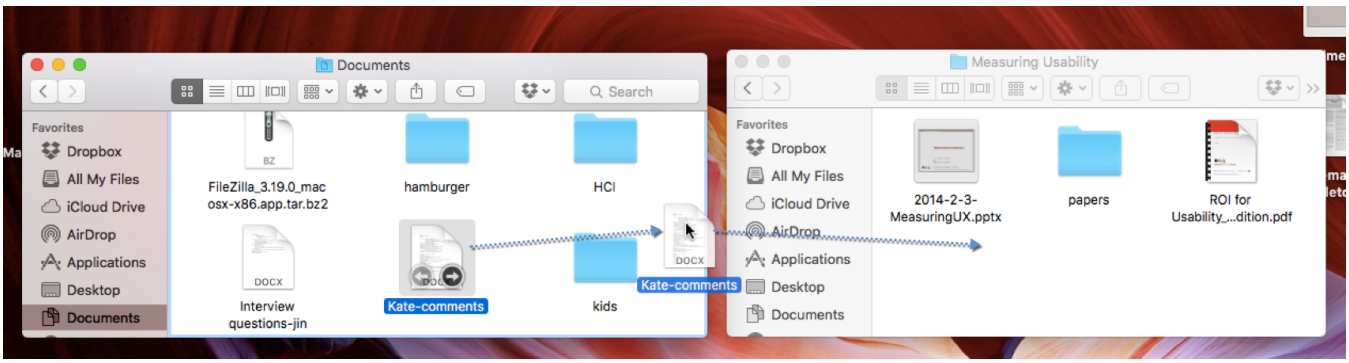
Summary: Direct manipulation is an interaction style in which the objects of interest in the UI are visible and can be acted upon via physical, reversible, incremental actions that receive immediate feedback.

Let's say that you're looking at an image of yourself on a roller coaster and want to see if your terrified expression has been caught on camera. What do you do? Something like this?



On a mobile phone you can pinch out to zoom into an image and pinch in to zoom out.

The action of using your fingertips to zoom in and out of the image is an example of a direct-manipulation interaction. Another classic example is dragging a file from a folder to another one in order to move it.



Moving a file on MacOS using direct manipulation involves dragging that file from the source folder and moving it into the destination folder.

> Definition: **Direct manipulation (DM)** is an interaction style in which users act on displayed objects of interest using physical, incremental, reversible actions whose effects are immediately visible on the screen.

Ben Shneiderman first coined the term “direct manipulation” in the early 1980s, at a time when the dominant interaction style was the command line. In **command-line interfaces,** the user must remember the system label for a desired action, and type it in together with the names for the objects of the action.



Moving a file in a command-line interface involves remembering the name of the command (“mv” in this case), the names of the source and destination folders, as well as the name of the file to be moved.

Direct manipulation is one of the central concepts of graphical user interfaces (GUIs) and is sometimes equated with “what you see is what you get” (WYSIWYG). These interfaces combine menu-based interaction with physical actions such as dragging and dropping in order to help the user use the interface with minimal learning.

The Characteristics of Direct Manipulation

In his analysis of direct manipulation, Shneiderman identified several attributes of this interaction style that make it superior to command-line interfaces:

- **Continuous representation of the object of interest**. Users can see visual representations of the objects that they can interact with. As soon as they perform an action, they can see its effects on the state of the system. For example, when moving a file using drag-and-drop, users can see the initial file displayed in the source folder, select it, and, as soon as the action was completed, they can see it disappear from the source and appear in the destination — an immediate confirmation that their action had the intended result. Thus, direct-manipulation UIs satisfy, by definition, the first usability heuristic: the visibility of the system status. In contrast, in a command-line interface, users usually must explicitly check that their actions had indeed the intended result (for example, by listing the content of the destination directory).
- **Physical actions instead of complex syntax**. Actions are invoked physically via clicks, button presses, menu selections, and touch gestures. In the move-file example, drag-and-drop has a direct analog in the real world, so this implementation for the move action has the right signifiers and can be easily learned and remembered. In contrast, the command-line interface requires users to recall not only the name of the command (“mv”), but also the names of the objects involved (files and paths to the source and destination folders). Thus, unlike DM interfaces, command-line interfaces are based on recall instead of recognition and violate an important usability heuristic.

- **Continuous feedback and reversible, incremental actions**. Because of the visibility of the system state, it's easy to validate that each action caused the right result. Thus, when users make mistakes, they can see right away the cause of the mistake and they should be able to easily undo it. In contrast, with command-line interfaces, one single user command may have multiple components that can cause the error. For instance, in the example below, the name of the destination folder contains a typo "Measuring Usablty" instead of "Measuring Usability". The system simply assumed that the file name should be changed to "Measuring Usablty". If users check the destination folder, they will discover that there was a problem, but will have no way of knowing what caused it: did they use the wrong command, the wrong source filename, or the wrong destination?



The command contains a typo in the destination name. Users have no way of identifying this error and must do detective work to understand what went wrong.

> This type of problem is familiar to everyone who has written a computer program. Finding a bug when there are variety of potential causes often takes more time than actually producing the code.

- **Rapid learning.** Because the objects of interest and the potential actions in the system are visually represented, users can use recognition instead of recall to see what they could do and select an operation most likely to fulfill their goal. They don't have to learn and remember complex syntax. Thus, although direct-manipulation interfaces may require some initial adjustment, the learning required is likely to be less substantial.

Direct Manipulation vs. Skeuomorphism

When direct manipulation first appeared, it was based on the office-desk metaphor — the computer screen was an office desk, and different documents (or files) were placed in folders, moved around, or thrown to trash. This underlying metaphor indicates the skeuomorphic origin of the concept. The DM systems described originally by Shneiderman are also **skeuomorphic** — that is, they are based on resemblance with a physical object in the real world. Thus, he talks about software interfaces that copy Rolodexes and physical checkbooks to support tasks done (at the time) with these tools.

As we all know, skeuomorphism saw a huge revival in the early iPhone days, and has now come out of fashion.



A skeuomorphic direct-manipulation interface for "playing" the piano on a phone

While skeuomorphic interfaces are indeed based on direct manipulation, not all direct-manipulation interfaces need to be skeuomorphic. In fact, today's flat interfaces are a reaction to skeuomorphism and depart from the real-world metaphors, yet they do rely on direct manipulation.

Disadvantages of Direct Manipulation

Almost each DM characteristic has a directly corresponding disadvantage:

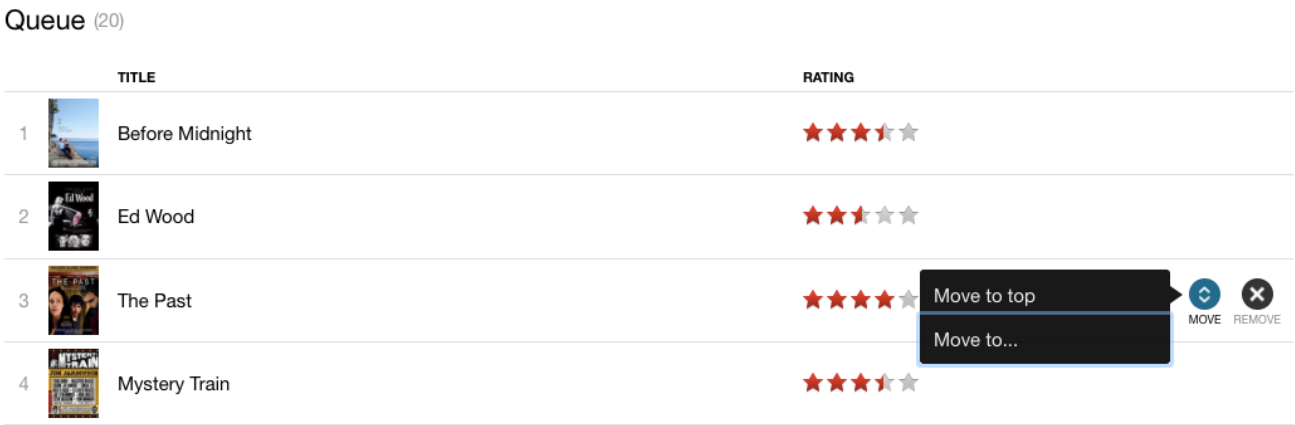
- *Continuous representation of the objects?* It means that you can only act on the small number of objects that can be seen at any given time. And objects that are out of sight, but not out of mind, can only be dealt with after the user has laboriously navigated to the place that holds those objects so that they can be made visible.
- *Physical actions?* One word: RSI (repetitive strain injury). It's a lot of work to move all those icons and sliders around the screen. Actually, two more words: accidental activation, which is particularly common on touchscreens, but can also happen on mouse-driven systems.
- *Continuous feedback?* Only if you attempt an operation that the system feels like letting you do. If you want to do something that's not available, you can push and drag buttons and icons as much as you want with no effect whatsoever. No feedback, only frustration. (A good UI will show in-context help to explain why the desired action isn't available and how to enable it. Sadly, UIs this good are not very common.)
- *Rapid learning?* Yes, if the design is good, but in practice learnability depends on how well designed the interface is. We've all seen menus with poorly chosen labels, buttons that did not look clickable, or drop-down boxes with more options than the length of the screen.

And there are even more disadvantages:

- **DM is slow**. If the user needs to perform a large number of actions, on many objects, using direct manipulation takes a lot longer than a command-line UI. Have you encountered any software engineers who use DM to write their code? Sure, they might use DM elements in their software-development interfaces, but the majority of the code will be typed in.
- **Repetitive tasks are not well supported.** DM interfaces are **great for novices** because they are easy to learn, but because they are slow, experts who have to perform the same set of tasks with high frequency, usually rely on keyboard shortcuts, macros, and other command-language interactions to speed up the process. For example, when you need to send an email attachment to one recipient, it is easy to drag the

desired file and drop it into the attachment section. However, if you needed to do this for 50 different recipients with customized subject lines, a macro or script will be faster and less tedious.

- **Some gestures can be more error-prone than typing**. Whereas in theory, because of the continuous feedback, DM minimizes the chance of certain errors, in practice, there are situations when a gesture is harder to perform than typing equivalent information. For example, good luck trying to move the 50th column of a spreadsheet into the 2nd position using drag and drop. For this exact reason, Netflix offers 3 interaction techniques for reordering subscribers' DVD queues: dragging the movie to the desired position (easy for short moves), a one-button shortcut for moving into the #1 position (handy when you *must* watch a particular movie ASAP), and the indirect option of typing the number of the desired new position (useful in most other cases).



*Netflix allows 3 interactions for rearranging a queue: dragging a movie to the desired position (not shown), moving it directly to top (*Move to top *option), or typing in the position where it needs to be moved (*Move to *option).*

- **Accessibility may suffer.** DM UIs may fail visually impaired users or users with motor skill impairments, especially if they are heavily based on physical actions, as opposed to button presses and menu selections. (Workarounds exist, but it can be difficult to implement them.)

Conclusion

It's hard to imagine modern interfaces without direct manipulation. Almost any interface that is aimed at a broad audience and has a graphical component is based on DM. With the explosion of touchscreen devices, we've seen DM UIs depart from the original office metaphors and innovate in a variety of domains. And augmented-reality and virtual-reality systems will push DM to even newer limits.

Despite the many downsides, we still recommend a heavy dose of direct manipulation for most UIs. Direct manipulation often enhances users' sense of empowerment over the computer by letting them feel that they are in control and are the ones making things happen. The upsides of DM usually enhance usability more than the downsides degrade it. Any interaction style has its minuses and can be ruined by lack of attention to the details: there is no magic bullet for UX, but there are definitely design ideas that can advance usability if employed correctly, and direct manipulation has proven to be one of these good ideas for more than 30 years.

References

Shneiderman, B. 1983. *Direct Manipulation: A Step Beyond Programming Languages*. Computer **16** (8), pp. 57–69. (Access-contolled archival copy available in ACM Digital Library.)

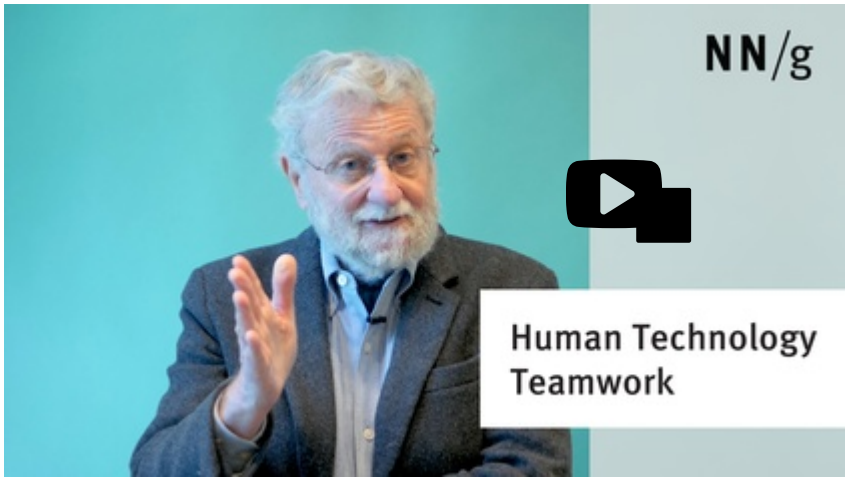
Videos



Are You a Cognitive Designer? (Don Norman)
2 minute video



Design for How People Think (Don Norman)
3 minute video



Human Technology Teamwork: The Role of Machines and Humans in Good UX Design (Don Norman)

2 minute video

Copyright © 1998-2018 Nielsen Norman Group, All Rights Reserved.