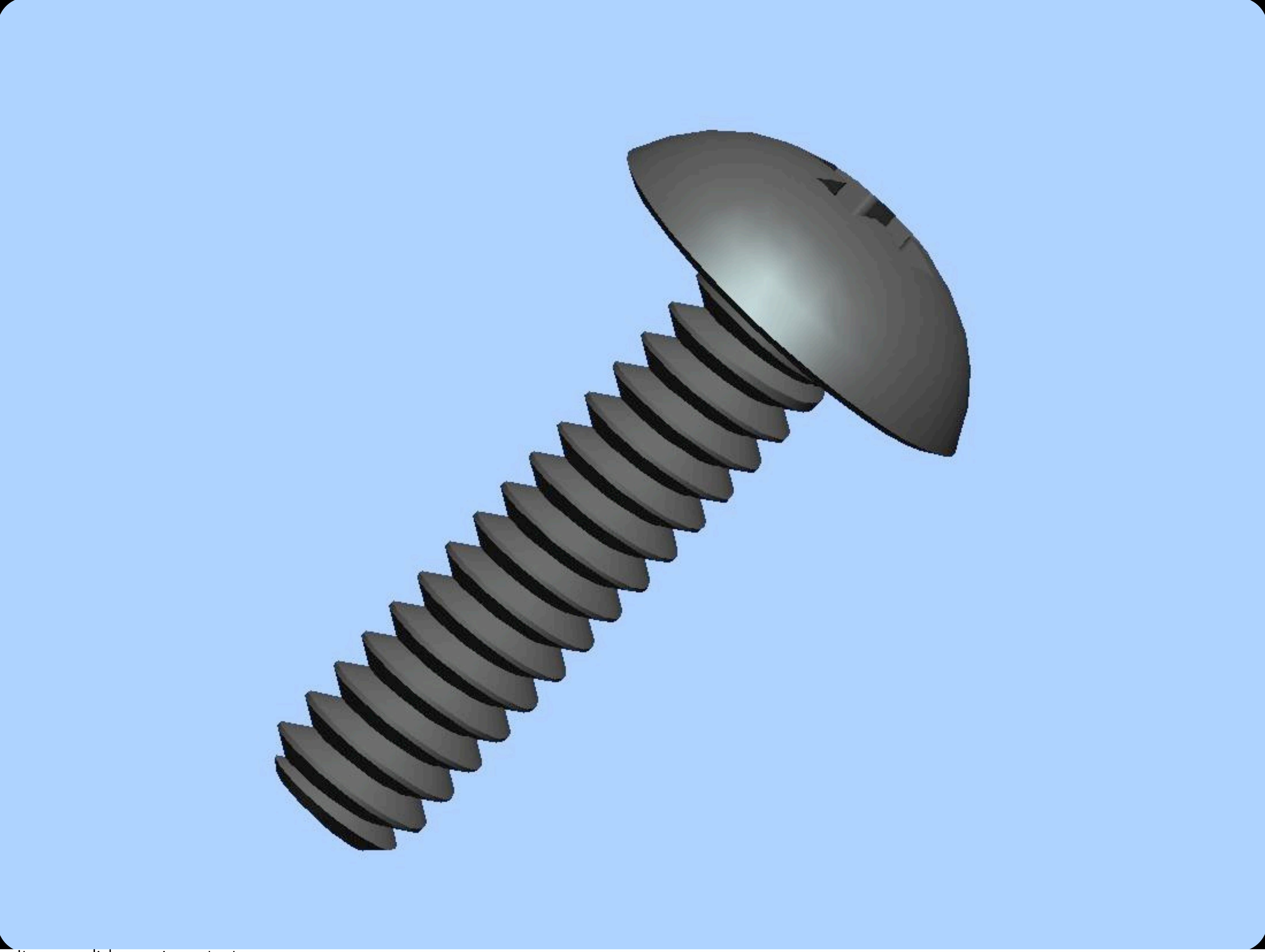
or other geometric designs
flat
closed
open
Alan's idea: attach the wires to the middle tubing
Bill: copyright visualization of licens
some string mechanism
w/ 2 servos / 2 sets of strings
pulling on string set 1 opens
2 closes
BOTTOM
SERVO 1
close
SERVO 2
open
PACHINKO
pinball machine
passive gate array
Scott: a gate that shows who walked through it last
Bill: a gate that measures
ceremonial gates
interlocking turnstiles
turnstiles
TOP

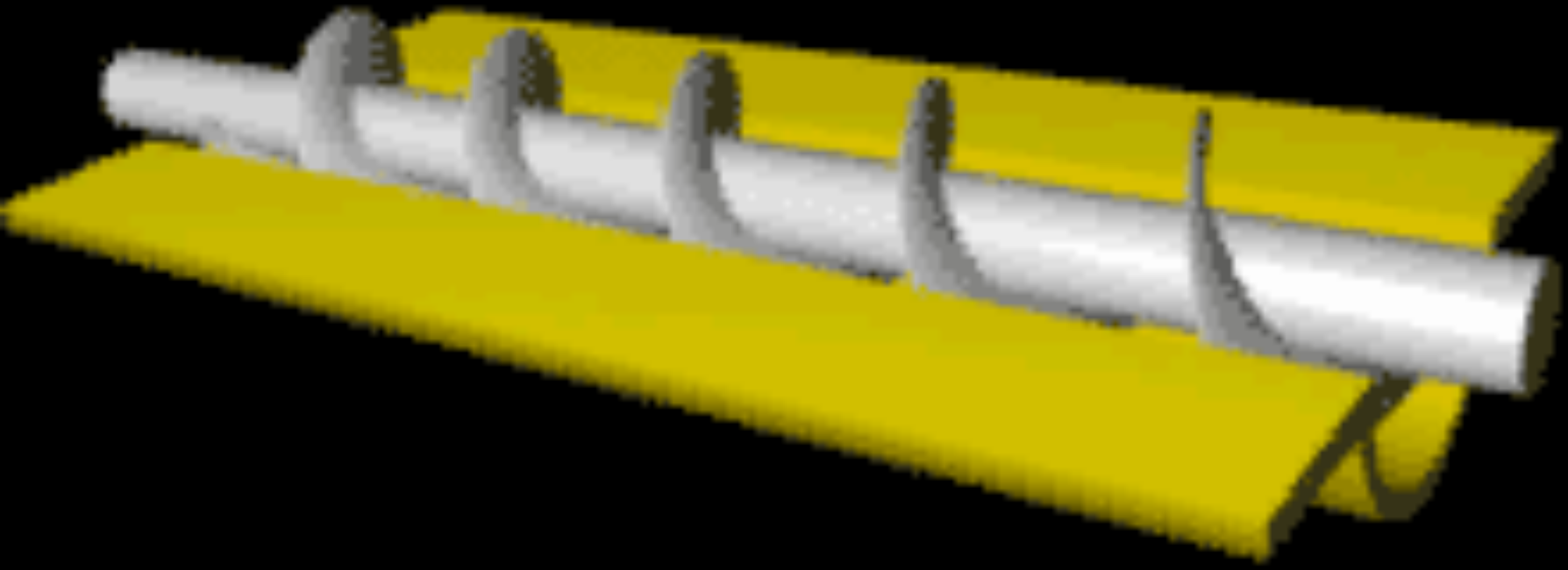
Software Tools

Scott Klemmer

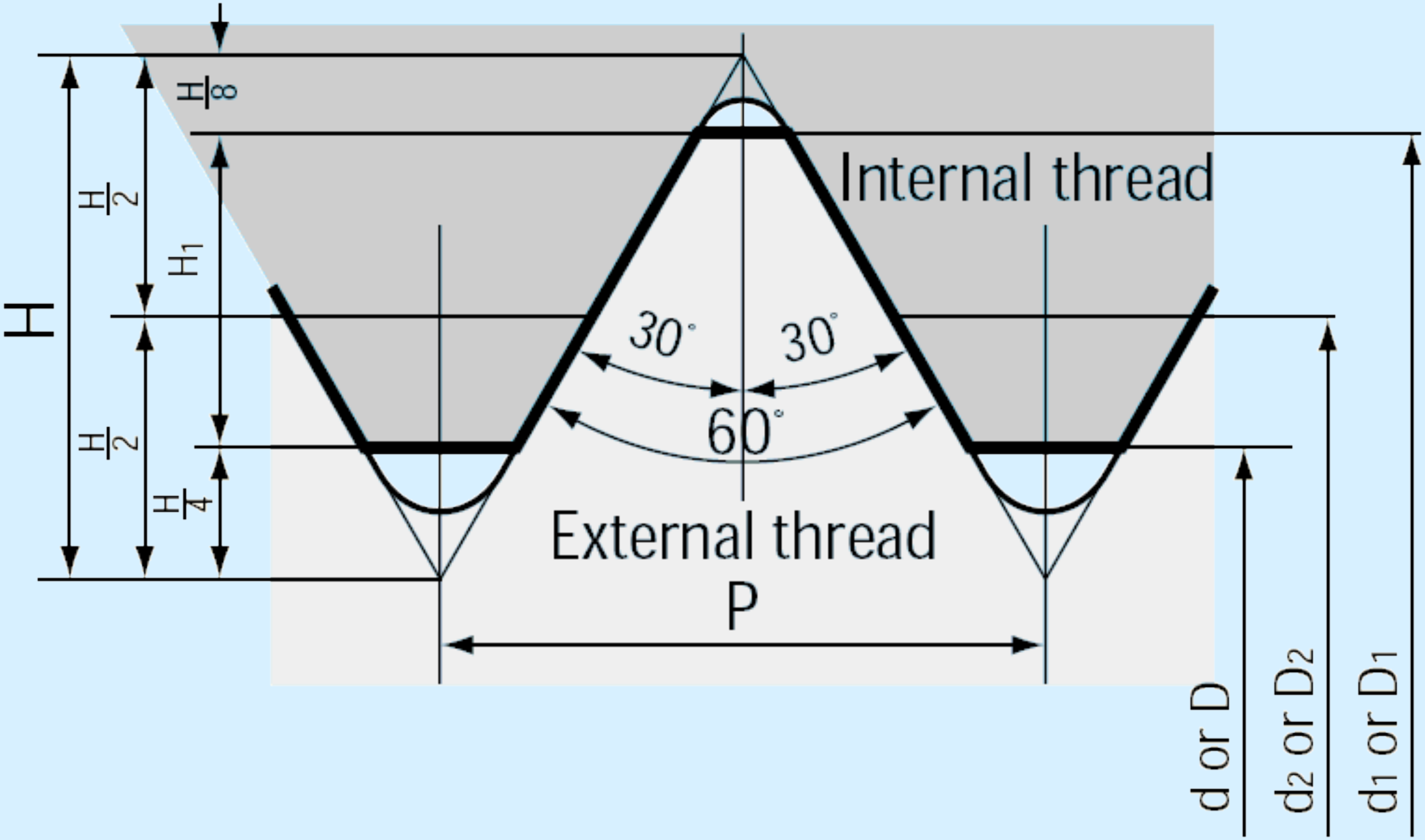
Autumn 2009



It accomplishes an important task



(for better and for worse) You don't have to make it yourself, and it abstracts a set of knowledge (in this case, how the threads of a screw act as a wedge)



Lastly, (for better and for worse) it provides a platform for standardization

On April 21, 1864, a man named William Sellers

Standards

Change where innovation happens. It inhibits innovation in the area of the standard (b/c ___), but it enables innovation in areas that build on the standard.

Developers are People Too…
…and tools are interfaces too

Toolkits

- A collection of widgets
  - Menus, scroll bars, text entry fields, buttons, etc.
- Toolkits help with programming
- Help maintain consistency among UIs
  - Key insight of Macintosh toolbox

➡ Path of least resistance translates into getting programmers to do the right thing

- Address common, low-level features for all UIs
  - ➡ Address the useful & important aspects of UIs

Why use toolkits?

- Code reuse saves programmer time
    - 50% of code is for the GUI [Myers & Rosson, CHI '92]
- Consistent look & feel across apps
- Easier to modify and iterate the UI
- Make UI development accessible to more people

What should tools do?

- Help **design** the interface given a specification of the tasks.
- Help **implement** the interface given a design.
- Help **evaluate** the interface after it is designed and propose improvements, or at least provide information to allow the designer to evaluate the interface.
- Create easy-to-use interfaces.
- Allow the designer to rapidly investigate different designs.
- Allow non-programmers to design and implement user interfaces.
- Provide portability across different machines and devices.
- Be easy to use themselves.

Tools Can Yield Better Interfaces

- Designs can be rapidly prototyped and implemented, possibly even before the application code is written.
- It is easier to incorporate changes discovered through user testing.
- More effort can be expended on the tool than may be practical on any single user interface since the tool will be used with many different applications.
- Different applications are more likely to have consistent user interfaces if they are created using the same user interface tool.
- Tools can enable a variety of specialists to participate in interface design

Tools Can Lower Maintenance Costs

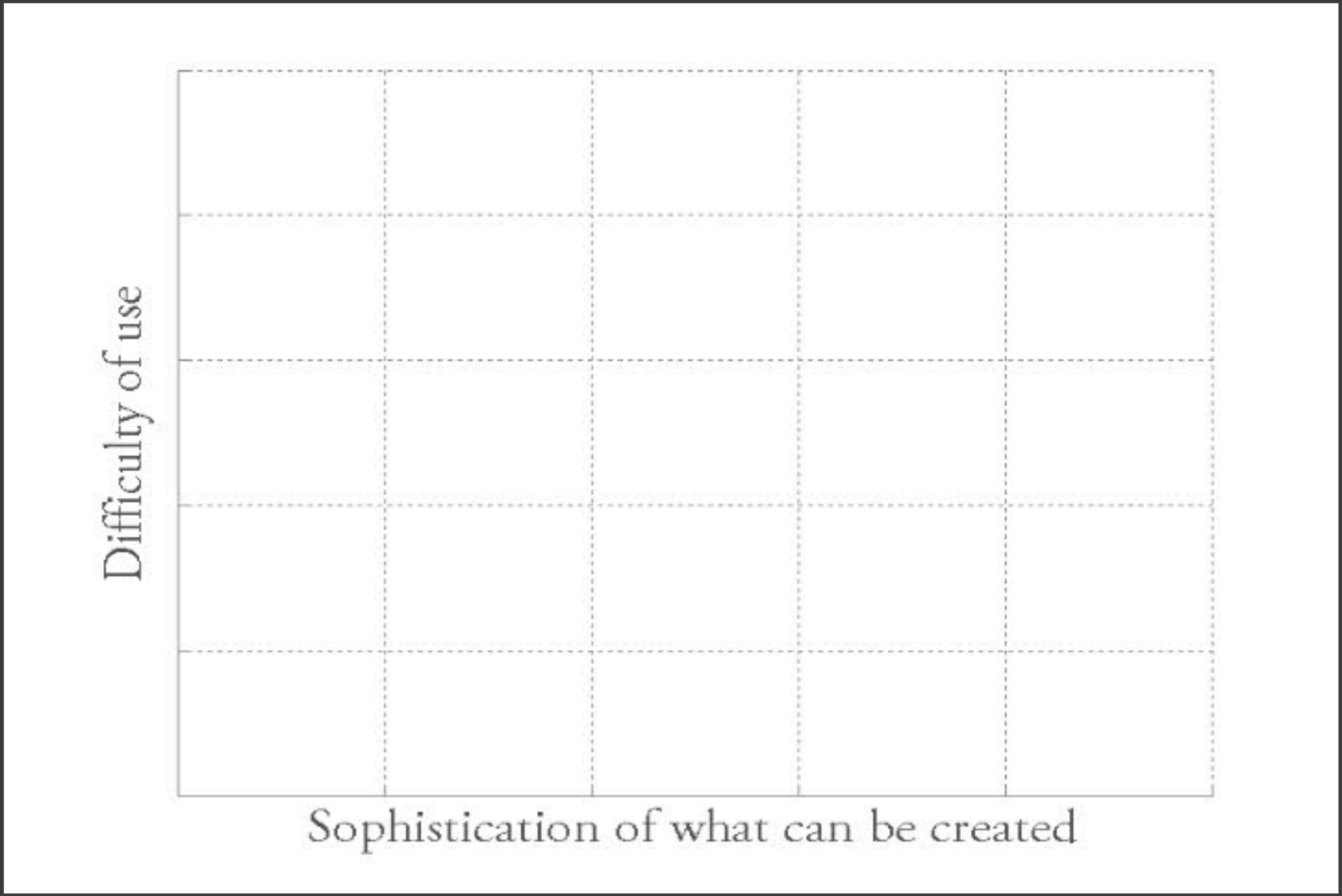
- There will be less code to write
- There will be better modularization due to the separation of the user interface component from the application.
- The level of expertise of the interface designers and implementers might be able to be lower, because the tools hide much of the complexities of the underlying system.
- Reliability may improve, since the code for the user interface is created automatically from a higher level specification.
- It may be easier to port an application to different hardware and software environments since the device dependencies are isolated in the user interface tool.

Success of Tools

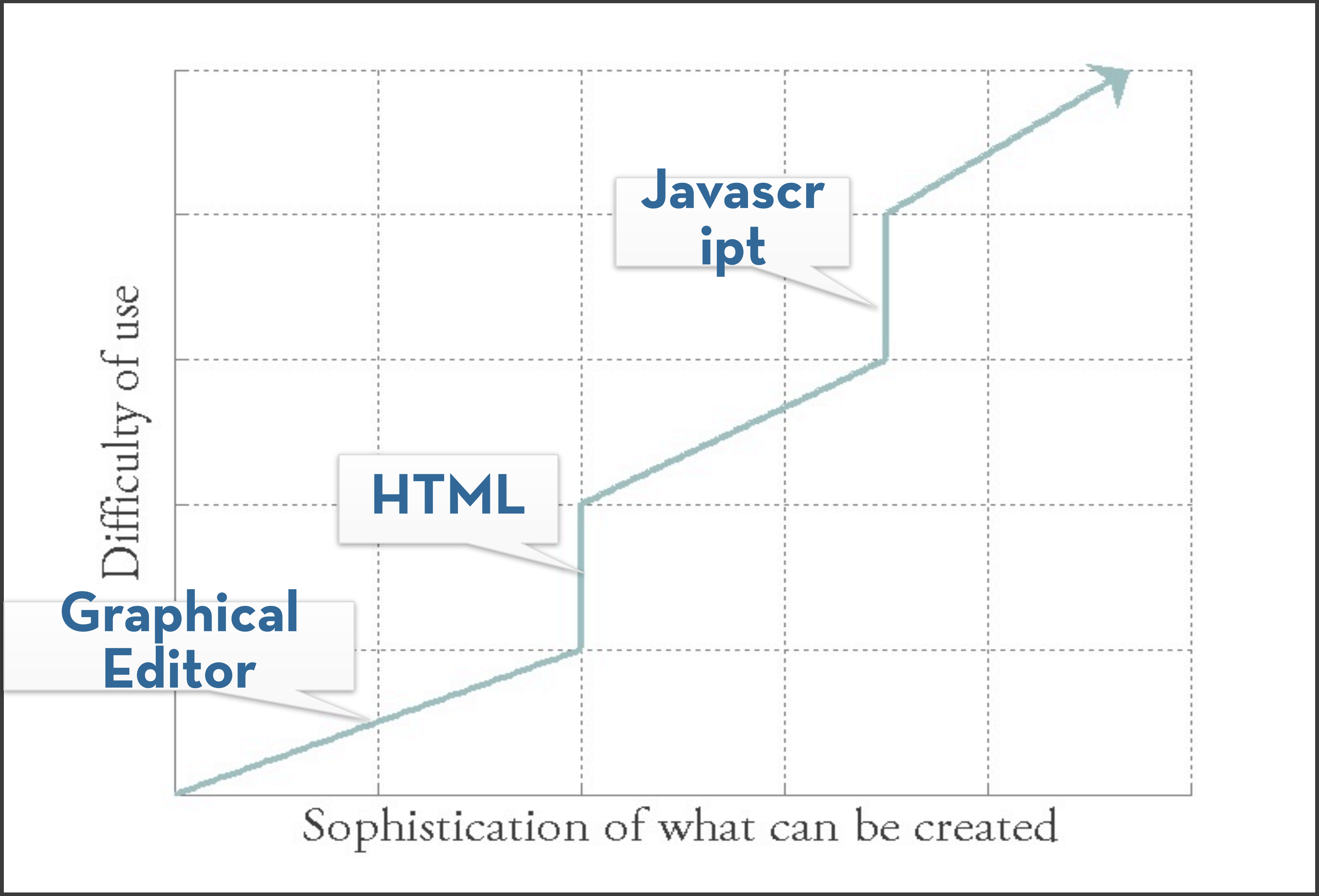
- Today's tools are highly successful
  - Window Managers, Toolkits, Interface Builders ubiquitous
  - Most software built using them
  - Are based on many years of HCI research

    Brad A. Myers. "A Brief History of Human Computer Interaction Technology." *ACM interactions*. Vol. 5, no. 2, March, 1998. pp. 44-54.

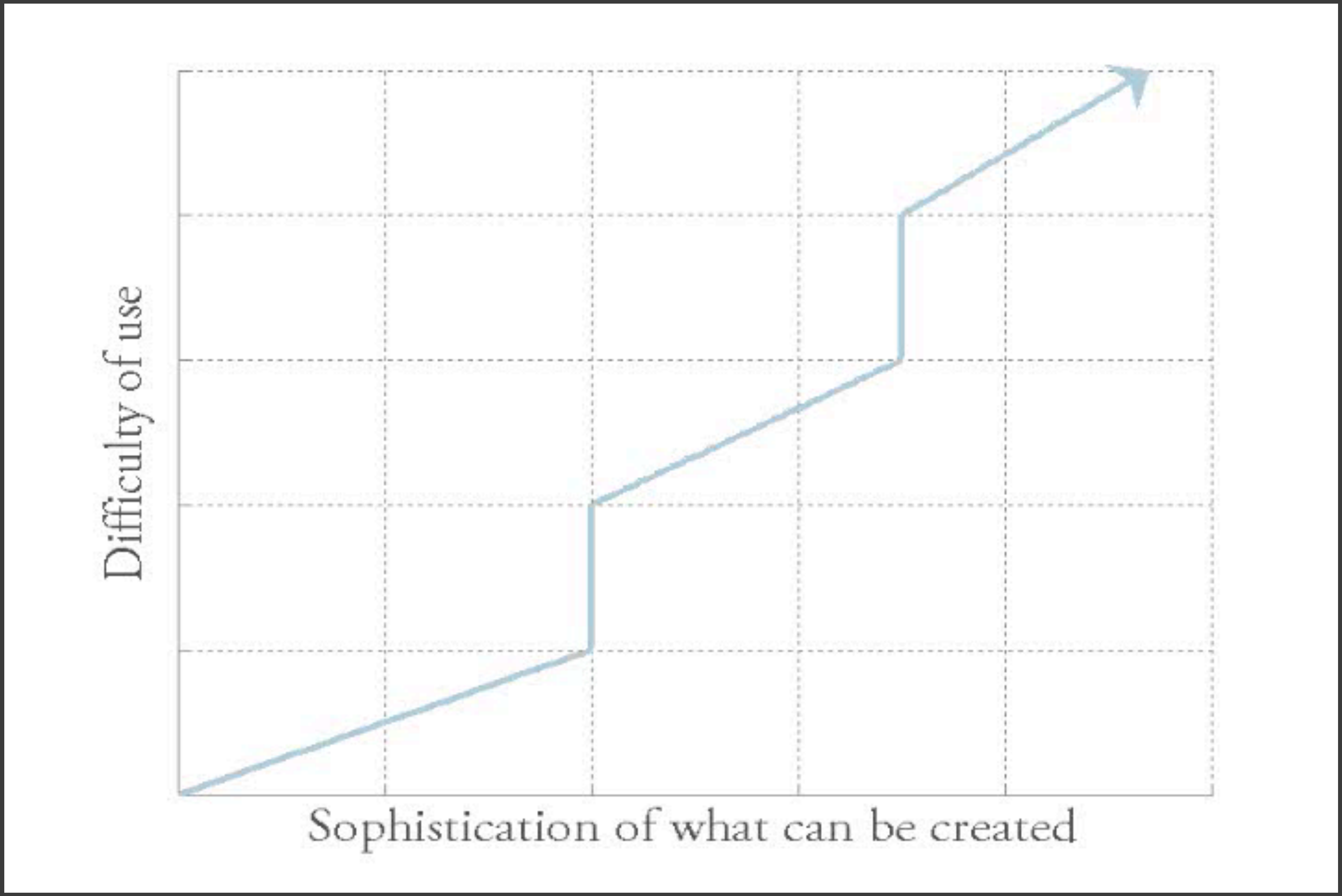
Plotting the Learning Curve



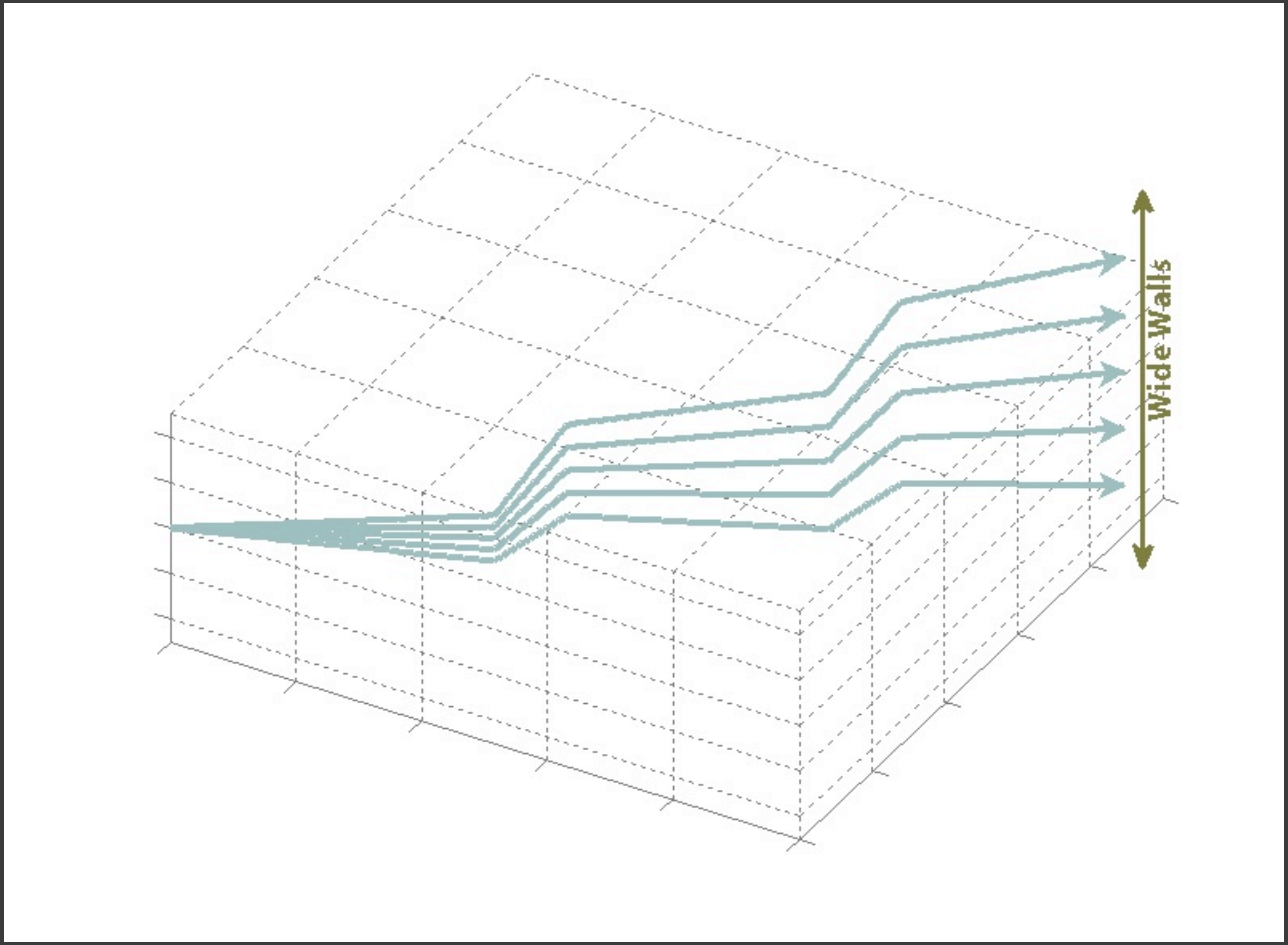
Plotting the Learning Curve



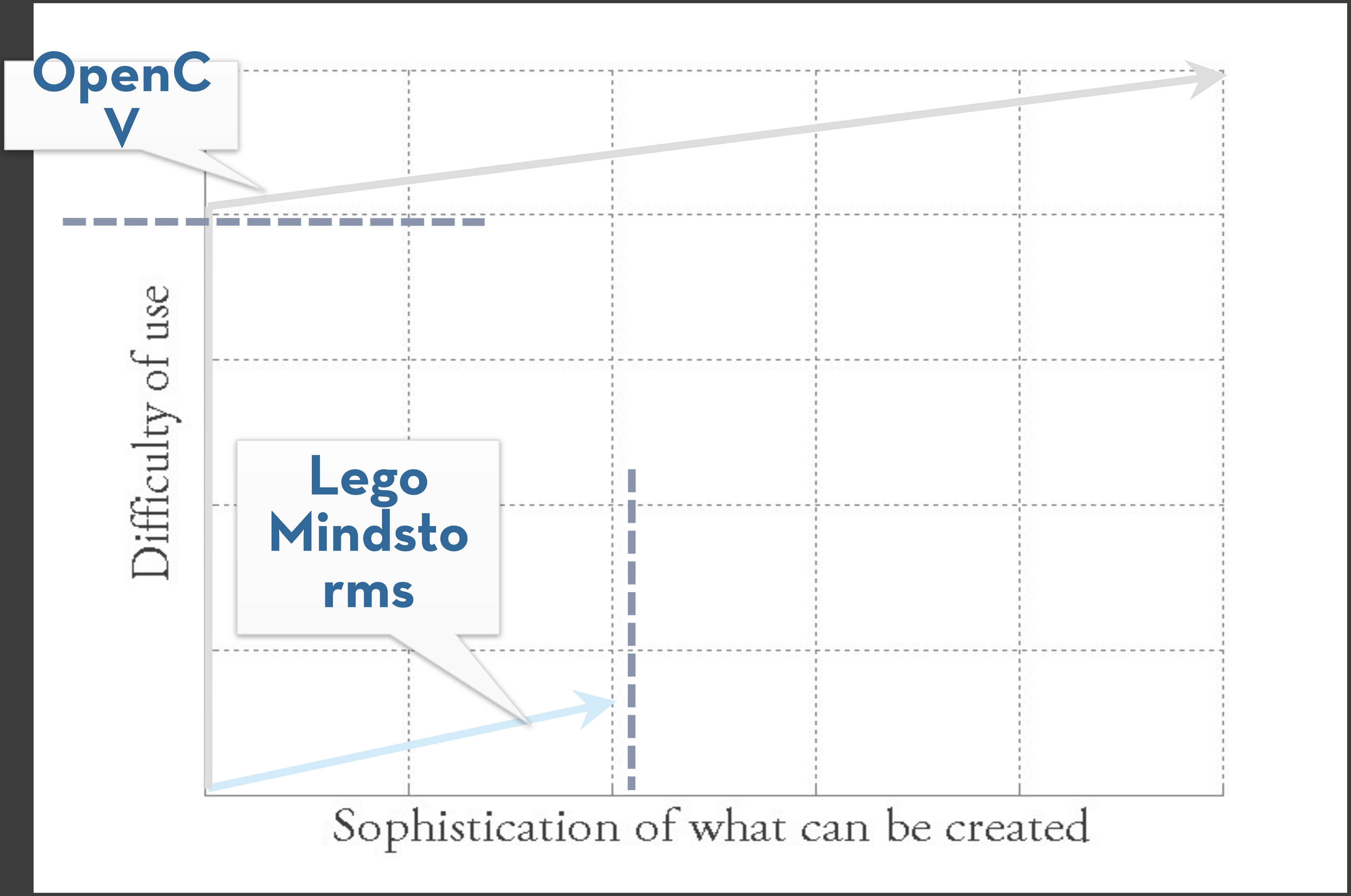
Threshold and Ceiling



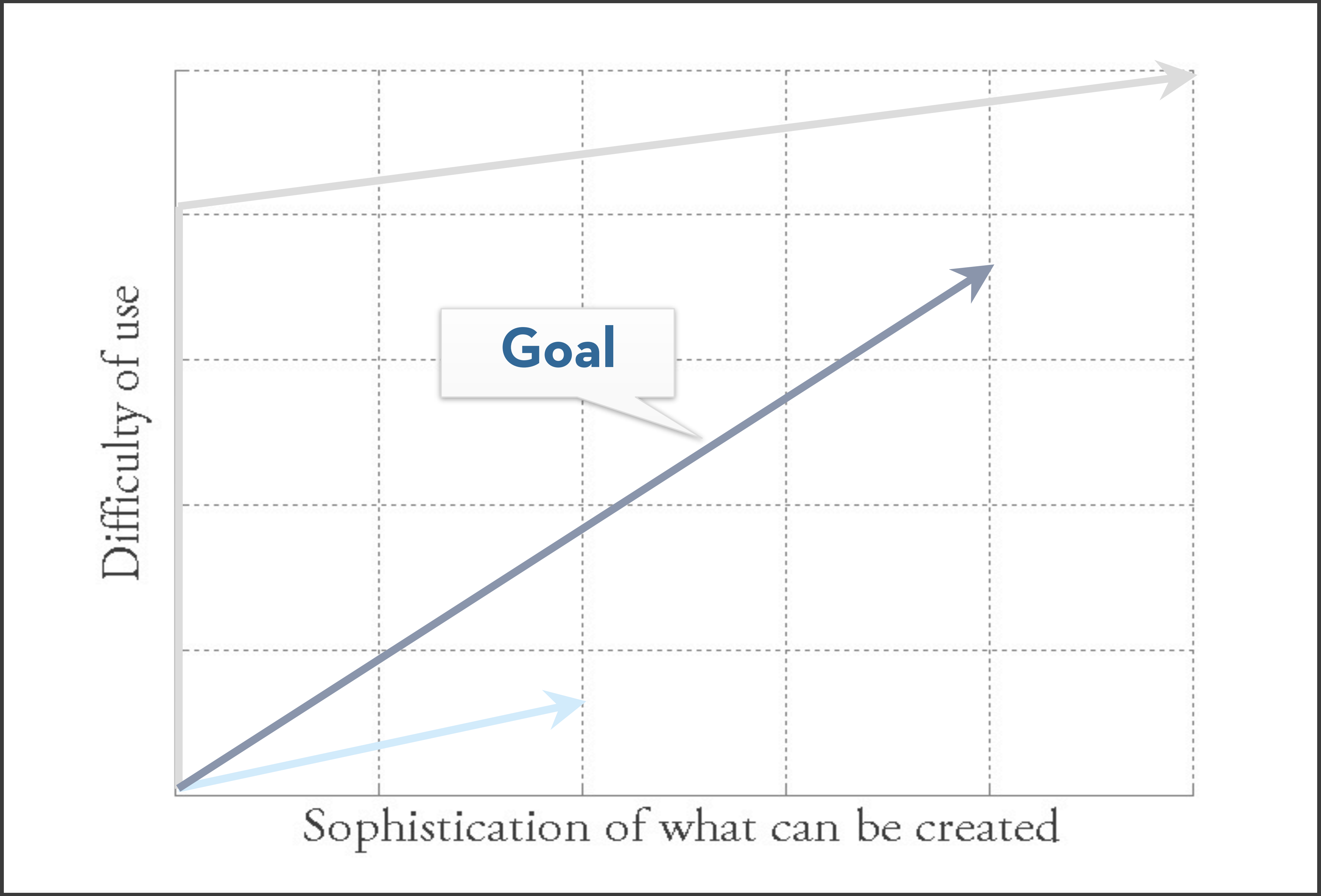
Threshold, Ceiling, and Walls



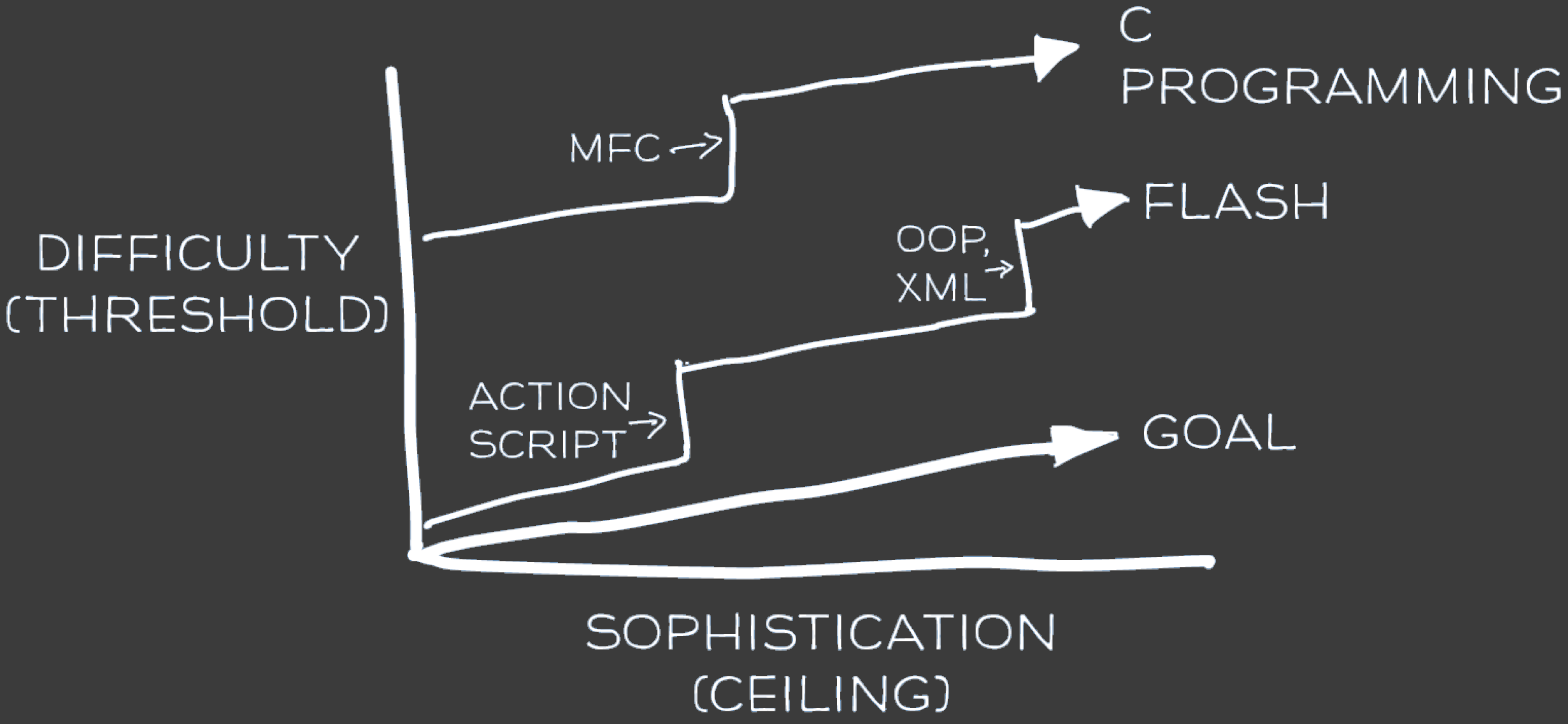
Threshold, Ceiling, and Walls



Threshold, Ceiling, and Walls



Threshold and Ceiling



(after Myers)

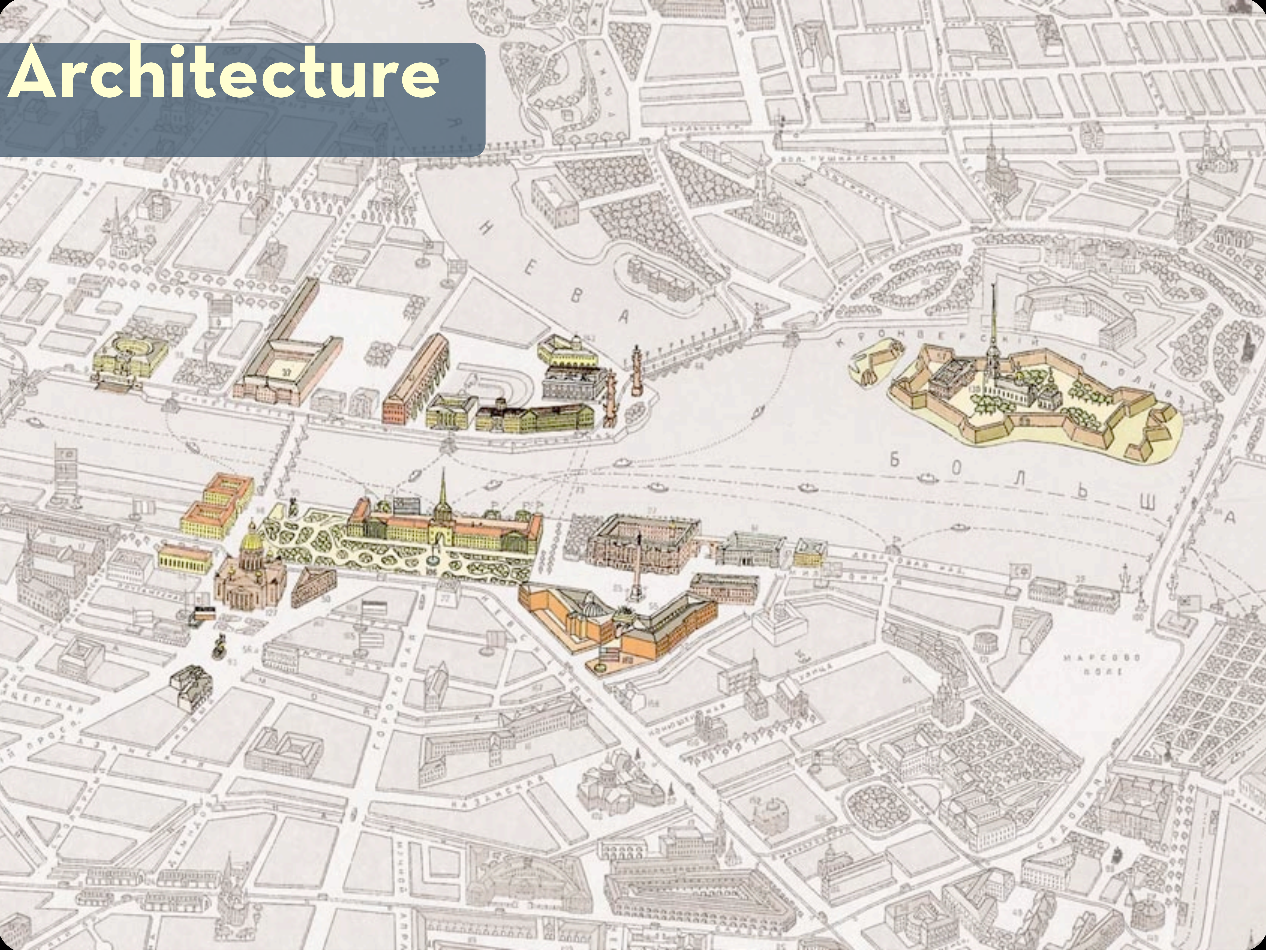
A success story: view source on the Web

Huge user interface innovation.

Library

Chinese Materia Medica
HERBAL
TREES
FLORA BRITANNICA
RICHARD MABEY
THE HEALING FOREST
MEDICINAL PLANTS
HERBS
WILDFLOWERS
People's plants
PHARMACOGNOSY
HERBAL MEDICINE

Architecture



Discussion of Themes, cont.

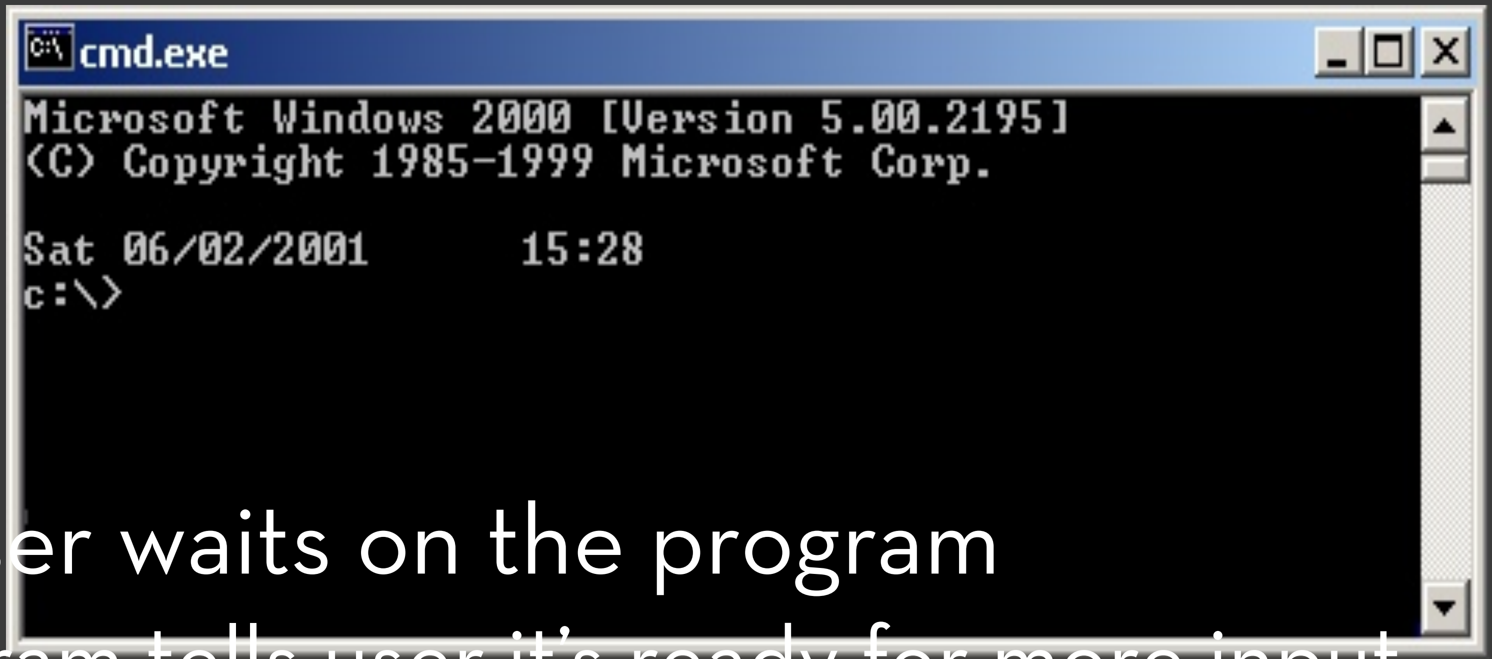
- Path of Least Resistance
  - Tools *should* guide implementers into better user interfaces
  - Goal for the future: do this more?
- Predictability
  - Programmers do not seem willing to release control
  - Especially when system may do sub-optimal things
- Moving Targets
  - Long stability of Macintosh Desktop paradigm has enabled maturing of tools

Window Managers

- Multiple (tiled) windows in research systems of 1960's: NLS, etc.
- Overlapping introduced in Alan Kay's thesis (1969)
- Smalltalk, 1974 at Xerox PARC
- Successful because multiple windows help users manage scarce resources:
  - Screen space and input devices
  - Attention of users
  - Affordances for reminding and finding other work

Sequential Programs

- Program takes control, prompts for input
  - command-line prompts (DOS, UNIX)



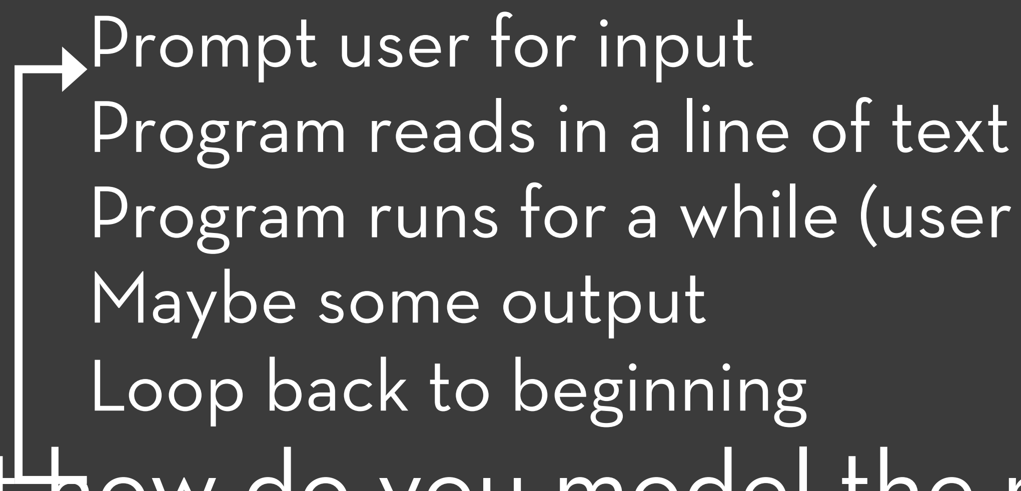
```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-1999 Microsoft Corp.

Sat 06/02/2001      15:28
c:\>
```

- The user waits on the program
  - program tells user it's ready for more input
  - user enters more input

Sequential Programs (cont.)

- General Flow

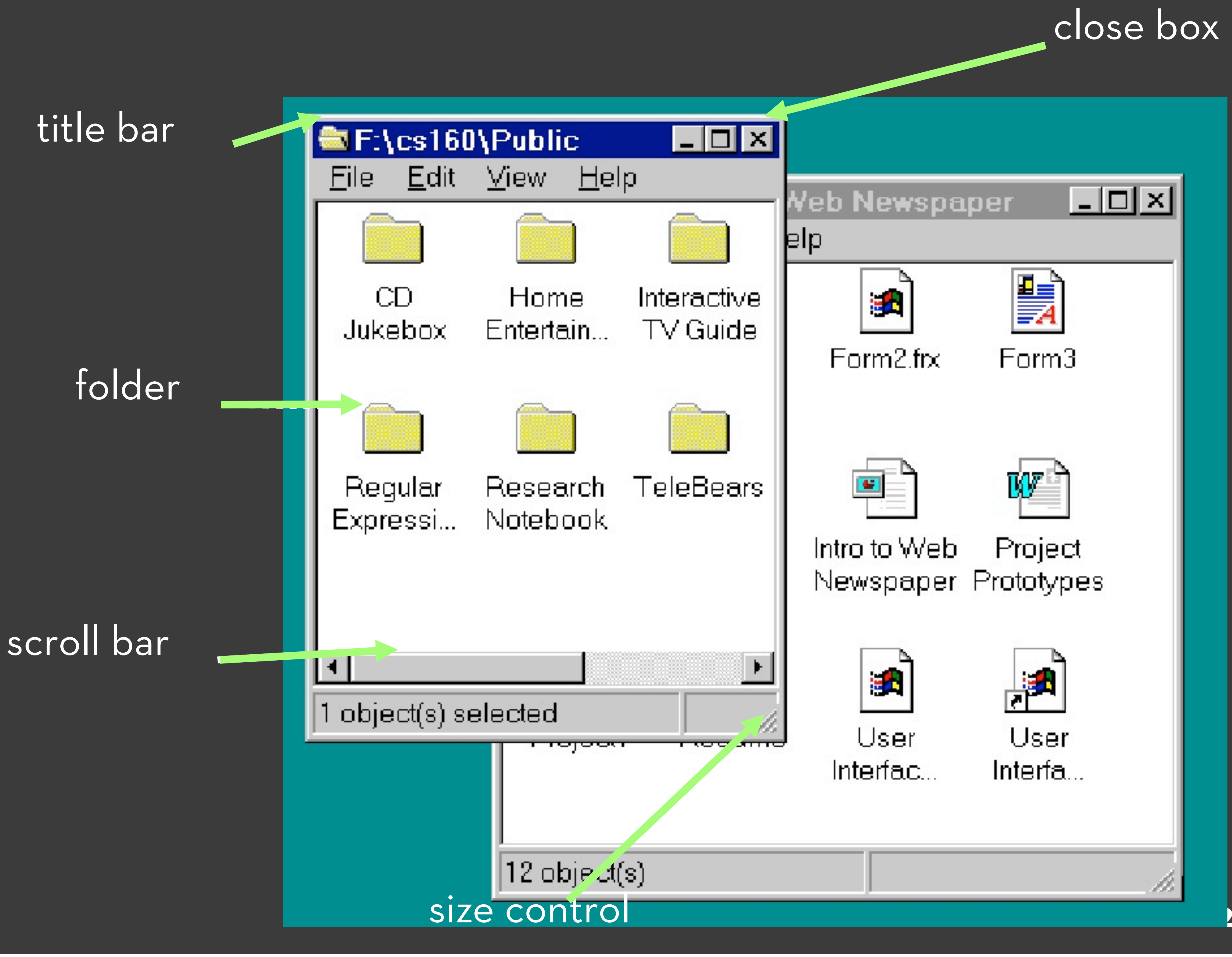


Prompt user for input
Program reads in a line of text
Program runs for a while (user waits)
Maybe some output
Loop back to beginning

- But how do you model the many actions a user can take?
  - for example, a word processor?
  - printing, editing, inserting, whenever user wants
  - sequential doesn't work as well for graphical and for highly-interactive apps

•Sequential works really well for text-based and low-interaction apps
•Doesn't work as well for graphical and high-interaction apps
•For example, how do you model the mouse input?

Example Interactions



1 object(s) selected

12 object(s)

title bar

close box

folder

scroll bar

size control

Modern GUI Systems

- Three concepts:
  - Event-driven programming
  - Widgets
  - Interactor Tree
- Describes how most GUIs work
  - Closest to Java
  - But similar to Windows, Mac, Palm Pilot

- Three concepts that are part of windowing systems today:
  - Event-driven programming
  - Widgets (Model / View / Controller, but we won't talk about that here)
  - Interactor Tree

- Describes how *most* windowing systems work
- Closest to how Java works, but if you understand this then you'll also understand how Windows, Macintosh, and Palm Pilot work too!

What this means for design

- Harder to use non-standard widgets
  - have to buy or create your own, ex. pie menus
- Easy to re-arrange widgets and layout of app, but hard to change behavior (i.e. the code)
  - provides some support, not a lot
  - stresses importance of getting features right first
- Harder to do things beyond mouse and keyboard
  - speech and sketching harder
- Harder to do multi-user multi-device apps