南京大学
NANJING UNIVERSITY

人机交互的软件工程方法
—— 交互设计模型与理论

主讲教师：冯桂焕
fgh@software.nju.edu.cn
2012年春季

背景

- 汽车上的刹车踏板和油门踏板相距很近，且刹车踏板要比油门踏板大很多
  - 经验告诉我们，可达到以最快的速度准确制动的目的
  - 但是，依据的原理是什么呢？
- 设计学科通常借助模型生成新的想法并对其测试
  - 如建筑学领域，有重量分布模型、空气环流模型、流体力学模型和光学模型等
- 交互设计领域
  - 计算用户完成任务的时间：KLM
  - 描述交互过程中系统状态的变化：动态转移网
  - 探讨任务的执行方法等：GOMS

预测模型

预测模型

- 能够预测用户的执行情况，但不需要对用户做实际测试
- 特别适合于无法进行用户测试的情形
  - 举例：为改进对员工使用计算机的支持，设计了许多可行方案。如何判断那一种方法更有效？
- 不同模型关注用户执行的不同方面
  - GOMS
    - 击键层次模型
  - Fitts定律

GOMS模型

- 最著名的预测模型
  - 1983年由Card, Morgan和Newell提出
  - 基于人类处理机模型
  - 泛指整个GOMS模型体系
- 是关于人类如何执行认知—动作型任务以及如何与系统交互的理论模型
  - 采用“分而治之”的思想，将一个任务进行多层次的细化
  - 把每个操作的时间相加就可以得到一项任务的时间
    - 操作指用户的目光从屏幕的一处移到另一处、识别出某个图标、手移到鼠标上

GOMS全称

- Goal-目标
  - 用户要达到什么目的
- Operator-操作
  - 任务执行的底层行为，不能分解
    - 为达到目标而使用的认知过程和物理行为
  - 如点击鼠标
- Method-方法
  - 如何完成目标的过程，即对应目标的子目标序列和所需操作
  - 如移动鼠标，输入关键字，点击Go按钮
- Selection-选择规则
  - 确定当有多种方法时选择和方法
  - GOMS认为方法的选择不是随机的

Goals

- 用户要达到的目的
  - 可分不同层次
  - 如“编辑一篇文章” 和“删除一个字符”
- 通常是层次化的
  - 高层次目标可分解为若干个低层次目标
- 通常表示为动作-对象序列
  - 如复制文件，创建目录等

Operators

- 用户为了完成任务必须执行的基本动作
  - 如双击鼠标左键、按回车键等
- 操作种类
  - 外部操作
    - 用户与系统之间可观测的物理操作
  - 心理操作
    - 用户内部行为
    - 不可观测
    - 如假想、猜测等
- 操作时间是上下文无关的
  - 花费的时间与用户正在完成什么样的任务或当前的操作环境没有关系

Methods

- 为实现目标所需要的操作序列
  - 外部 + 心理
  - 如使用鼠标单击输入域，输入关键字，再单击“查找”按钮
- 若用户能够执行一个任务，则表示其拥有该任务的一种方法
  - 目标对应的方法不唯一

```
GOAL: CLOSE-WINDOW
.    [select GOAL: USE-MENU-METHOD
                .    MOVE-MOUSE-TO-FILE-MENU
                .    PULL-DOWN-FILE-MENU
                .    CLICK-OVER-CLOSE-OPTION
               GOAL: USE-ALT-F4-METHOD
                .    PRESS-ALT-F4-KEYS]
```

Selection Rules

- 选择规则是用户要遵守的判定规则
  - 以确定在特定环境下所使用的方法
  - GOMS中并不认为这是一个随机的选择
  - 而是根据特定用户、系统的状态、目标的细节来预测要选择哪种方法。

- 举例，对用户Sam:

Rule 1: Use the CLOSE-METHOD unless another rule applies

Rule 2: If the application is GAME, use ALT-F4-METHOD

GOMS实例

任务: 编辑文档

GOAL:EDIT-MANUSCRIPT
  GOAL:EDIT-UNIT-Task … repeat until no more unit tasks
    GOAL:ACQUIRE-UNIT-TASK
      GET-NEXT-PAGE … if at end of manuscript
      GET-NEXT-TASK
    GOAL:EXECUTE-UNIT-TASK
      GOAL:MODIFY-TEXT
        [select: GOAL:MOVE-TEXT… if text is to be moved
          GOAL:DELETE-PHRASE… if phrase is to be deleted
          GOAL:INSERT-WORD] … if a word is to be inserted
        VERIFY-EDIT

举例

- 使用GOMS模型描述在Word中删除文本的过程
  - 目标：删除Word中的文本
  - 方法1：使用菜单删除文本
    - 步骤1：思考，需要选定待删除的文本
    - 步骤2：思考，应使用“剪裁”命令
    - 步骤3：思考，“剪裁”命令在“编辑”菜单中
    - 步骤4：选定待删除文本，执行“剪裁”命令
    - 步骤5：达到目标，返回

- 方法2：使用“删除键”删除文本
  - 步骤1：思考，应把光标定位在待删除的第一个字符处
  - 步骤2：思考，需要使用“删除Del”键
  - 步骤3：定位光标，按“删除”键逐个删除字符
  - 步骤4：达到目标，返回

- 选择规则如下：
  - 1：若需要删除大量文本，则使用鼠标，通过菜单进行删除；
  - 2：若只是删除个别词，则使用“删除键”进行删除

GOMS方法步骤

- 选出最高层的用户目标
- 写出具体的完成目标的方法
  - 即激活子目标
- 写出子目标的方法
  - 递归过程，一直分解到最底层操作时停止

- 子目标的关系
  - 顺序关系
  - 选择关系
    - 以select：引导

GOMS模型分析

- 优点
  - 能够容易地对不同的界面或系统进行比较分析
  - 如Ernestine项目说明，GOMS有助于确定新产品的有效性
  - 美国电话公司NYNEX
    - 利用GOMS分析一套即将被采用的新的计算机系统的应用效果不理想，放弃了使用新系统，为公司节约了数百万的资金。
- 局限性
  - 假设用户完全按一种正确的方式进行人机交互，没有清楚地描述错误处理的过程
  - 只针对那些不犯任何错误的专家用户
  - 任务之间的关系描述过于简单
  - 忽略了用户间的个体差异

四种GOMS模型

- 击键层次模型
  - 简化的假设
- CMN GOMS (Card Moran Newell)
  - 伪码描述，结构严格
- NGOMSL (Natural GOMS Language)
  - 程序形式，结构泛化
  - 仅能够预测性能和学习次数
- CPM GOMS (Cognitive Perceptual Motor GOMS)
  - 基于Model-Human Processor
  - 允许操作符的并行操作

击键层次模型

- Card等1983
- 对用户执行情况进行量化预测
  - 仅涉及任务性能的一个方面：时间
- 用途
  - 预测无错误情况下专家用户在下列输入前提下完成任务的时间
  - 便于比较不同系统
  - 确定何种方案能最有效地支持特定任务

操作符

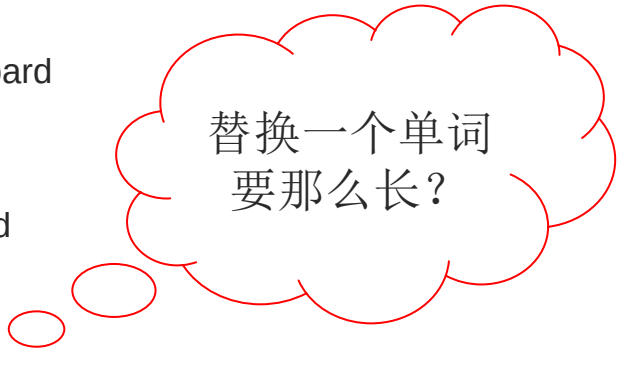
Operator name	Description	Time (see)
按键 K	Pressing a single key or button	0.35 (average)
	Skilled typist (55 wpm)	0.22
	Average typist (40 wpm)	0.28
	User unfamiliar with the keyboard	1.20
	Pressing shift or control key	*0.08*
定位 P	Pointing with a mouse or other device to a target on a display	1.10
P_1	Clicking the mouse or similar device	0.20
复位 H	Homing hands on the keyboard or other device	0.40
绘图 D	Draw a line using a mouse	Variable depending on the length of line
思维 M	Mentally prepare to do something (**e.g.,** make a decision)	1.35
R(t)	System response time--counted only if it causes the user to wait when carrying out their task	t

使用

- 执行时间预测方法
  - 列出操作次序，累加每一项操作的预计时间
  - $T_{execute}=T_k+T_P+T_h+T_d+T_m+T_r$
- 举例
  - DOS环境下执行“ipconfig”命令
    - MK[i] K[p] K[c] K[o] K[n] K[f] K[i] K[g] K[回车]
    - 简略表达版本：M9K[ipconfig回车]
    - $T_{execute}=1.35+9*0.20=3.15s$
  - 菜单选择
    - H[鼠标]MP【网络连接图标】K[右键]P[修复]K[左键]
    - $T_{execute}=0.40+1.35+2P+2K=4.35$秒

编码方法

- 举例：替换文字编辑器中长度为5个字符的单词
  - 任务准备 M
  - 将手放在鼠标上 H_{mouse}
  - 将鼠标移到单词 P_{word}
  - 选择单词 K
  - 回到键盘 $H_{keyboard}$
  - 准备键入 M
  - 键入新的5字符单词 $5K_{word}$



- $T_{execute}=2T_M+T_P+2T_H+6T_K = 6.28s$

放置M操作符的启发规则

以编码所有的物理操作和响应操作为开端。接着使用规则 0 放置所有的候选 M 操作符，然后循环执行规则 1 到 4，并对每一个 M 操作判断是否应该删除	
规则 0	在所有 K 操作符前插入 M 操作符，要求 K 操作的值不能是参数字符串（如数字或文本）的一部分。在所有的对应于选择命令（非参数）的 P 操作符前放置 M 操作
规则 1	如果某个操作符前的 M 操作完全可以由 M 之前的操作符预测，则删除 M（如 PMK—PK）
规则 2	如果一串 MK 组成的字符串是一个认知单元（如一个命令的名字），则删除第一个 M 以外的所有 M
规则 3	如果 K 是一个冗余的终结符（如紧跟在命令参数终结符后面的命令终结符），则删除 K 之前的 M
规则 4	如果 K 是常量字符串（如一个命令名）的终结符，则删除 K 之前的 M。如果 K 是变量字符串（如参数字符串）的终结符，则保留 K 之前的 M

KLM分析

- 建模可以给出执行标准任务的时间
- 但没有考虑下面的问题
  - 错误
  - 学习性
  - 功能性
  - 回忆
  - 专注程度
  - 疲劳
  - 可接受性

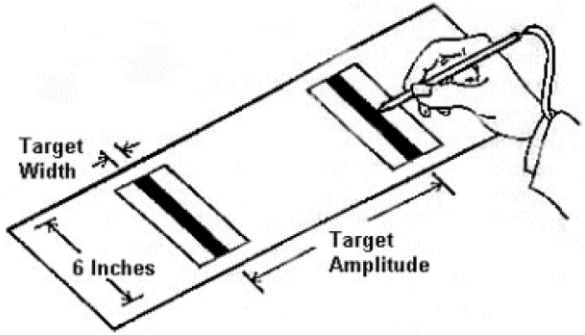
KLM应用

- 在交互设计早期阶段为用户性能提供有效、准确的模型
- 案例1：鼠标驱动的文本编辑
  - Xerox Star研发过程，解决了不存在专家用户的困难
- 案例2：查号工作站
  - 贝尔实验室，探讨操作员流程的效率
  - 设想：使用尽可能少的输入从数据库中得到尽可能多的结果(“少量输入-大量结果”)
  - 使用KLM计算标准查询时间，证明查询过程中的输入数与返回结果数之间存在折中

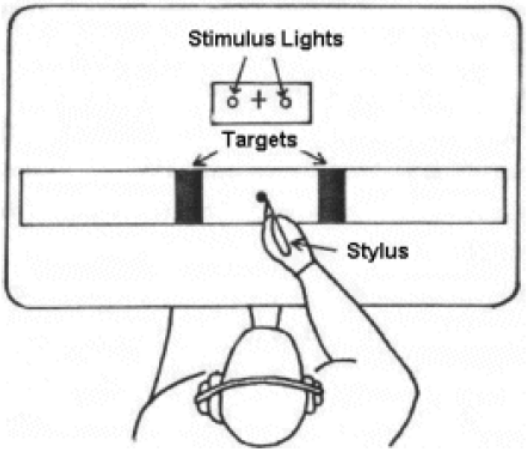
Fitts定律

- 用户访问屏幕组件的时间对于系统的使用效率是至关重要的
  - 哪些特性会影响访问效率呢？
  - Fitts，1954
- 能够预测使用某种定位设备指向某个目标的时间
- 人机交互中，根据目标大小及至目标的距离，计算指向该目标的时间
  - 可指导设计人员设计按钮的位置、大小和密集程度
- 对图形用户界面设计有明显的意义
- “最健壮并被广泛采用的人类运动模型之一”

“轮流轻拍”实验



- 记录拍中和失误的情况
- 指令
  - 尽可能准确而不是快速的轮流轻拍两个薄板
- 以实验数据为依据，得到困难指数如下



$$ID = \log_2(2A/W)$$

- Paul M. Fitts (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, volume 47, number 6, June 1954, pp. 381-391. (Reprinted in *Journal of Experimental Psychology: General*, 121(3):262–269, 1992).
- Paul M. Fitts and James R. Peterson (1964). Information capacity of discrete motor responses. *Journal of Experimental Psychology*, 67(2):103–112, February 1964.

- Fitts定律描述了人类运动系统的信息量
- 信息论中的Shannon定理

$$C = B \log_2(S/N+1)$$

  - C是有效信息量(比特)，B是通道带宽，S是信号能量，N是噪声

- Fitts定律
  - S映射为运动距离或振幅(A)，N映射为目标的宽度(W)

三个部分

- 困难指数ID (Index of Difficulty) = $\log_2(A/W+1)$ (bits)
  - 对任务困难程度的量化
  - 与宽度和距离有关
- 运动时间MT (Movement Time) = a + b*ID (secs)
  - 在ID基础上将完成任务的时间量化
- 性能指数IP (Index of Performance) = ID/MT (bits/sec)
  - 基于MT和ID的关系
  - 也称吞吐量

- MacKenzie改写为

$$ID = \log_2(A/W+1)$$

  - 更好地符合观察数据
  - 精确地模拟了支撑Fitts定律的信息论
  - 计算出的任务困难指数总是整数

- 平均时间*MT*

$$MT = a + b\log_2(A/W+1)$$

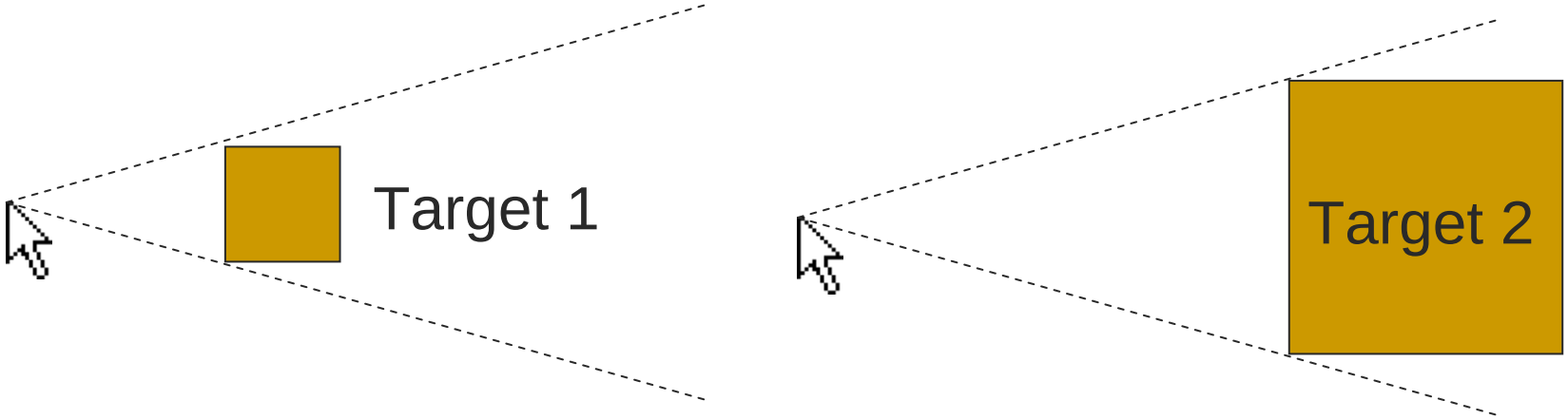
  - 常数a和b来自如图所示的线性回归

a,b的确定

- 设计一系列任务，A和W分别取不同的值
- 对每一种条件下的任务
  - 尝试多次
  - 记录每次执行时间
  - 进行统计分析
- 记录准确性
  - 记录选择的x,y坐标，或
  - 错误率，即鼠标落在目标区域外的百分比

Fitts' Law

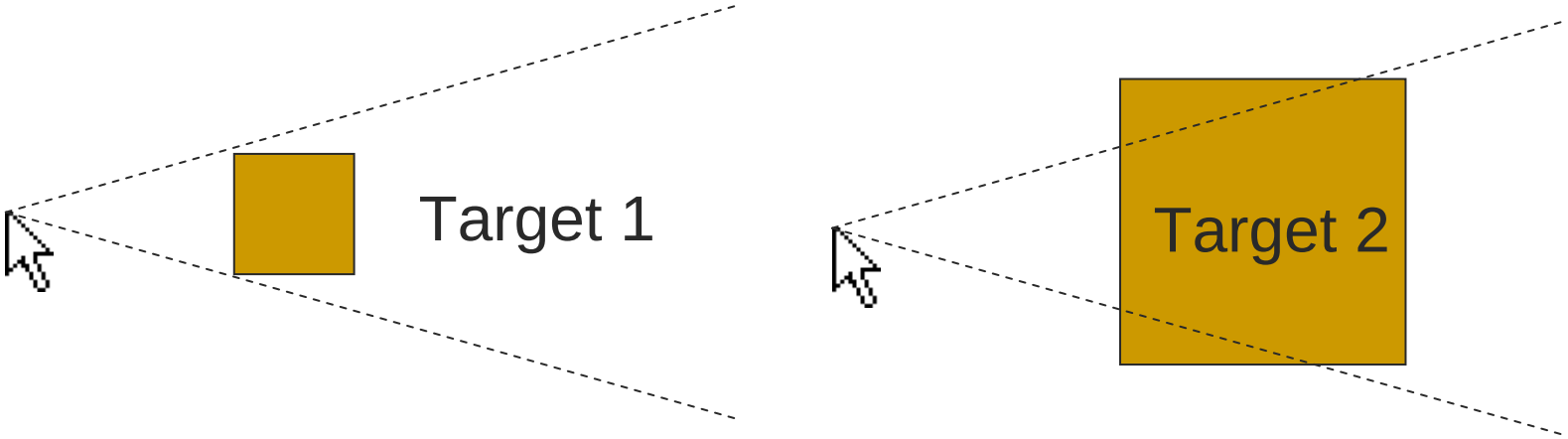
$$MT = a + b * ID,\ ID = \log_2(A/W+1)$$



Same ID → Same Difficulty

Fitts' Law

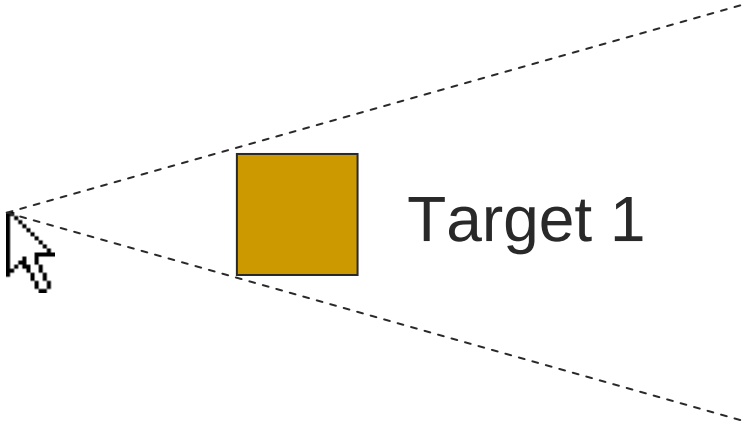
$$MT = a + b * ID,\ ID = \log_2(A/W+1)$$

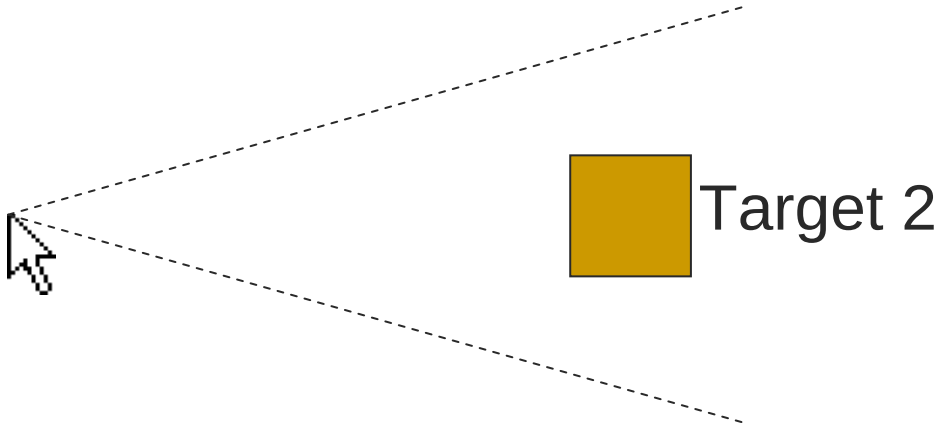


Smaller ID → Easier

Fitts' Law

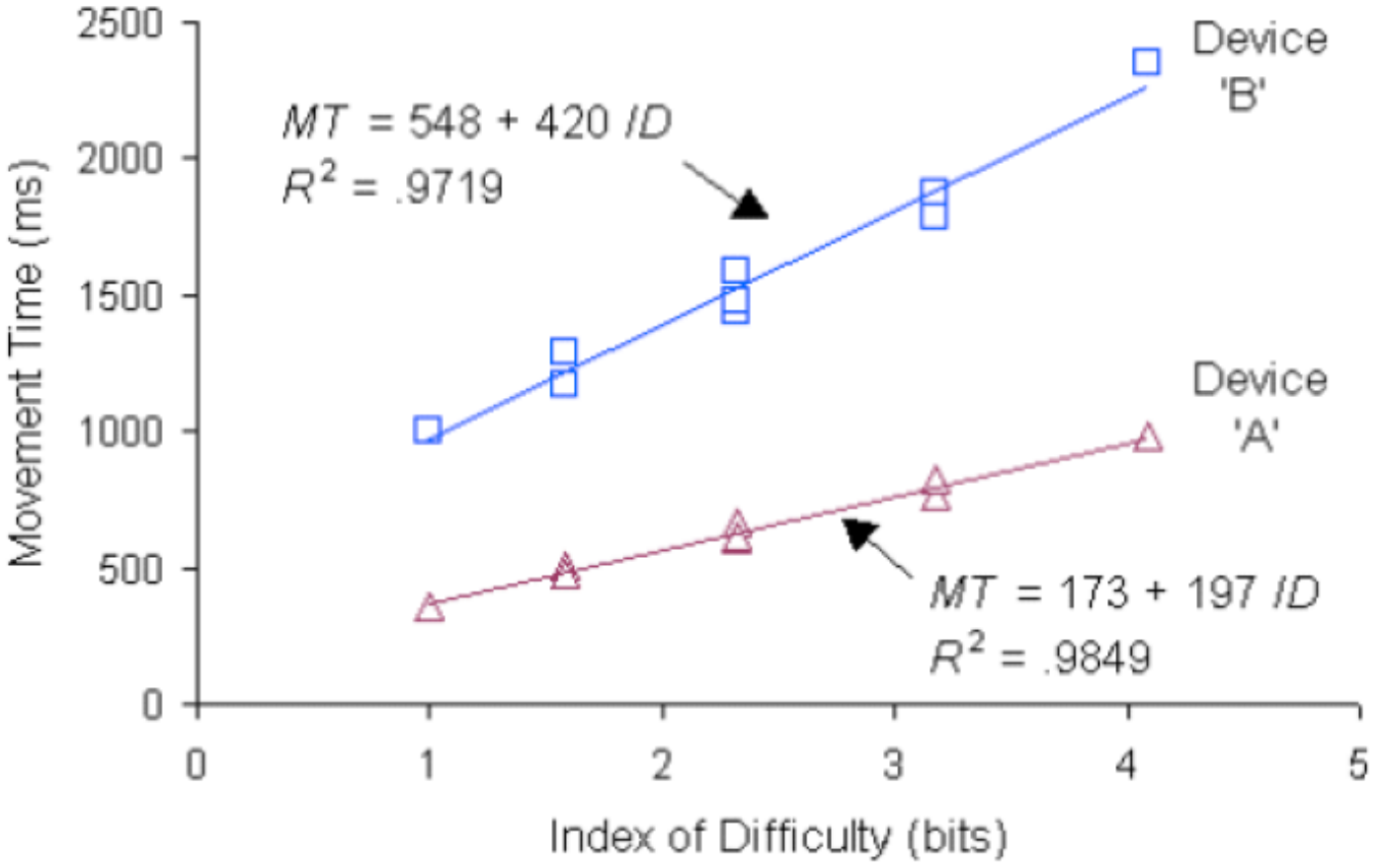
$$MT = a + b * ID,\ ID = \log_2(A/W+1)$$





Larger ID → Harder

A (pixels)	*W* (pixels)	*ID* (bits)	Device 'A'		Device 'B'	
			ER (%)	*MT* (ms)	*ER* (%)	*MT* (ms)
40	10	2.32	2.08	665	1.25	1587
40	20	1.58	3.33	501	2.08	1293
40	40	1.00	1.25	361	0.42	1001
80	10	3.17	2.92	762	2.08	1874
80	20	2.32	1.67	604	2.08	1442
80	40	1.58	1.67	481	0.83	1175
160	10	4.09	3.75	979	2.08	2353
160	20	3.17	5.42	823	1.67	1788
160	40	2.32	4.17	615	0.83	1480
Mean:		*2.40*	*2.92*	*644*	*1.48*	*1555*


2500
2000
1500
1000
500
0
Movement Time (ms)
0
1
2
3
4
5
Index of Difficulty (bits)
MT = 548 + 420 ID
R² = .9719
Device 'B'
Device 'A'
MT = 173 + 197 ID
R² = .9849

说明

- 如果MT的计算单位是秒，则a的测量单位是秒，b的测量单位是秒/比特(ID的测量单位是比特)
- 系数a(截距)和b(斜率)由经验数据确定，且与设备相关
- 对于一般性计算，可使用a=50,b=150(单位是毫秒)
- A和W在距离测量单位上必须一致，但是不需要说明使用的具体单位

Fitts定律建议

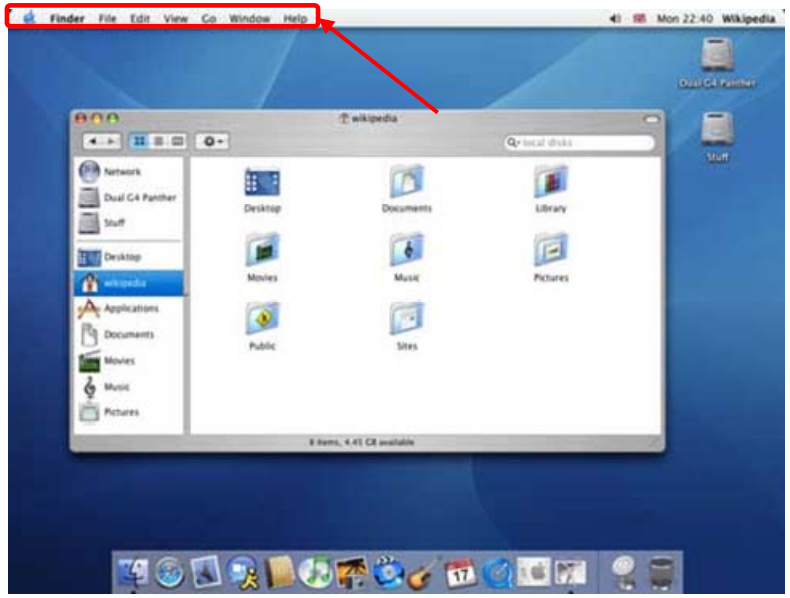
- 大目标、小距离具有优势
  - 对选择任务而言，其移动时间随到目标距离的增加而增加，随目标的大小减小而增加
- 屏幕元素应该尽可能多的占据屏幕空间
- 最好的像素是光标所处的像素
- 屏幕元素应尽可能利用屏幕边缘的优势
- 大菜单，如饼型菜单，比其他类型的菜单使用简单

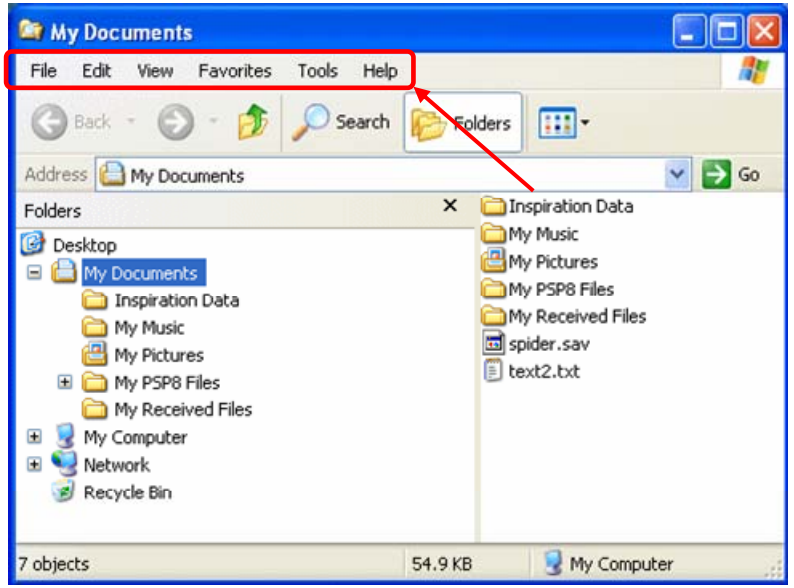
Fitts定律应用

- 首先被Card等人应用在HCI领域
  - 鼠标的定位时间和错误率都优于其他设备
  - 鼠标速率接近最快速率
  - 使用鼠标完成运动任务比使用其他设备更加协调，这在交互设计中非常重要
- 策略一：缩短当前位置到目标区域的距离
  - 如右键菜单技术
- 策略二：增大目标大小以缩短定位时间
  - Windows操作系统和Macintosh操作系统中的应用程序菜单区域位置的设计

应用实例

- Mac OS和Windows XP的比较（苹果专利）
  - Mac OS的菜单是沿着屏幕边缘排列的
  - Windows OS的菜单位于标题栏下面





Jeff Raskin

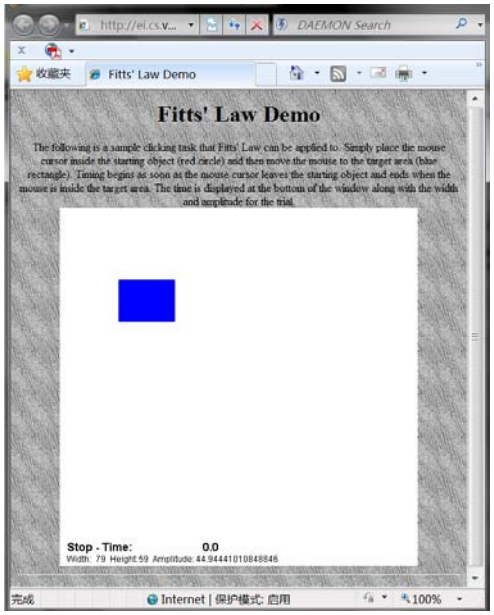
- 用户往往在距离屏幕边缘50毫米处停下来
  - 50毫米作为Mac OS的菜单宽度
- 对于Mac OS
  - ID = 50 + 150 $\log_2$(80/50+1) = 256微妙
- 对于Windows OS
  - ID = 50 + 150 log2(80/5+1) = 663微妙

Mac OS “dock”

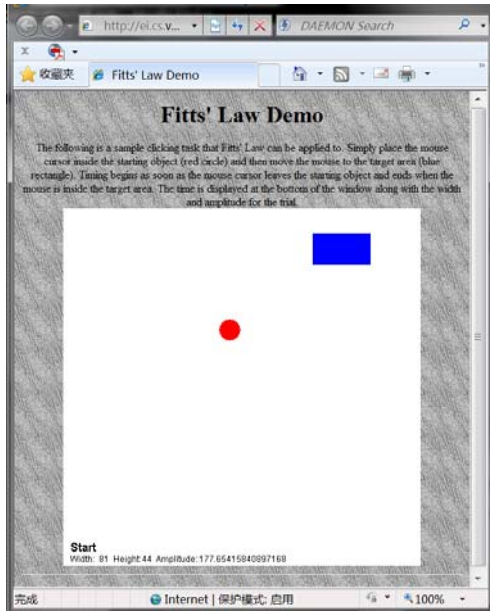
- 工具栏组件大小可以动态改变
  - 为用户提供了一个放大的目标区域
  - 可显示更多图标
  - 新版Mac操作系统中都实现了扩展工具栏
- 思考：
  - 该工具栏存在何种优缺点？

Fitts' Law, example 1

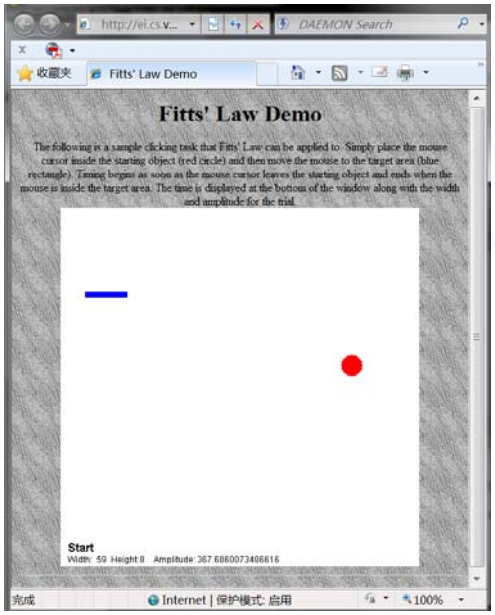
- **http://ei.cs.vt.edu/~cs5724/g1/tap.html#**



Width:79, height:59
amplitude:44.94441…
Time = 0.0s



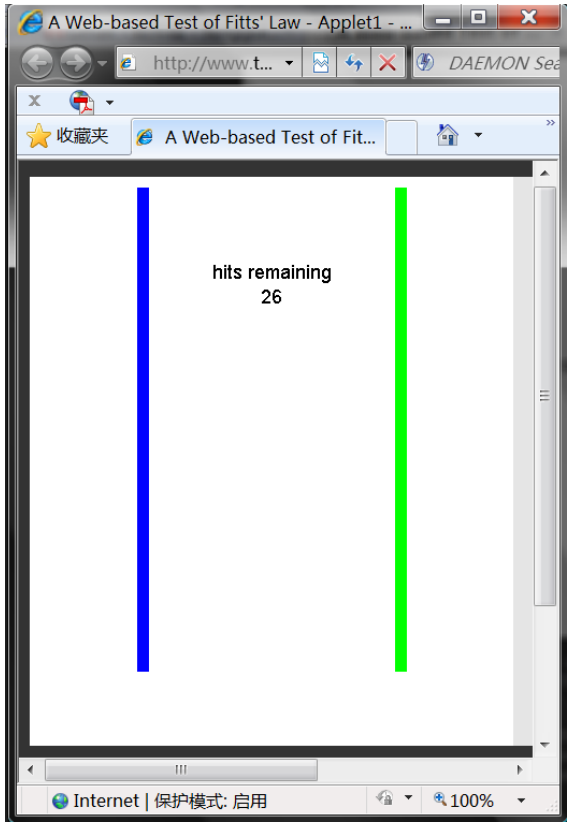
Width:81, height:44
amplitude:177.65415…
Time = ?

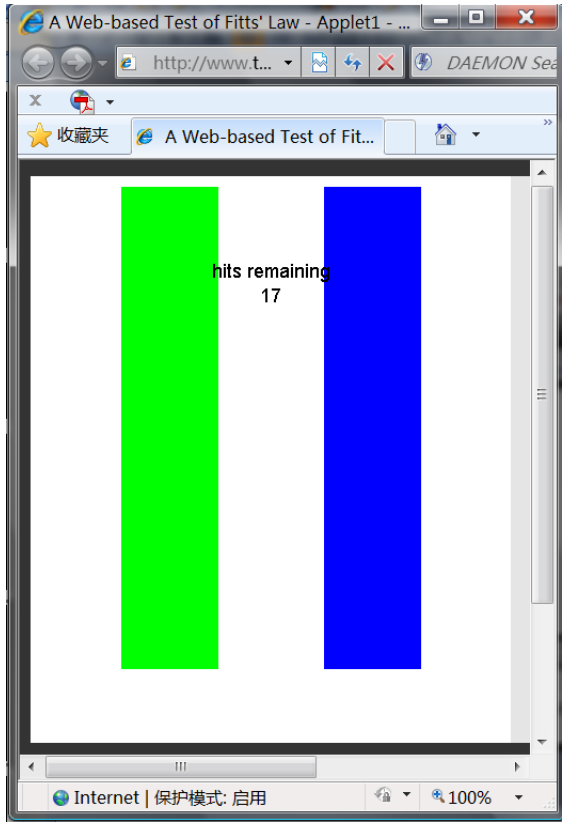


Width:59, height:8
amplitude:367.68600…
Time = ?

Fitts' Law, example 2

- www.tele-actor.net/fitts/

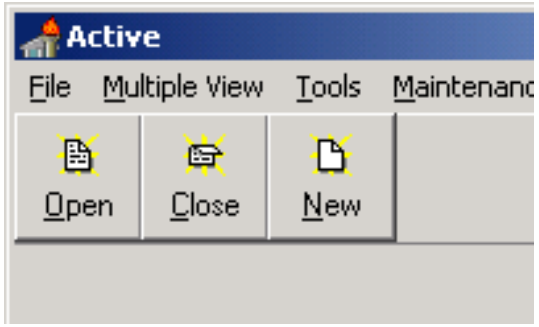




Fitts定律测验一

摘自：

http://www.asktog.com/columns/022DesignedToGiveFitts.html



- 微软工具栏允许用户在图标下方显示图标标签
- 列举一条原因，解释为什么显示标签后工具条的访问速度更快？
  - 假设用户明确每个图标的用途

备选答案

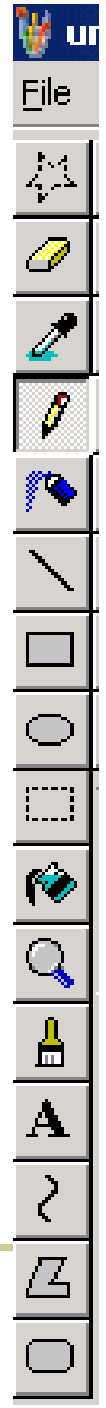
- 标签成为图标的一部分，从而加大了图标面积。根据Fitts定律，在其他条件不变的情况下，目标越大，访问越快。
- 未使用标签的情况下，工具栏图标过于拥挤。

- 图形应用工具中的调色板如右图
  - 每个图标的大小为16X16像素
  - 以2列X8行排列在屏幕左侧

- 问题
  - 不改变图标大小，且保持图标阵列位于屏幕左侧，采取何种方式可减少访问每个图标所需的时间？

可选方案

- 将阵列改为1列X16行，从而使所有图标均分布在屏幕边缘



练习

- 在某窗口管理器中，将窗口关闭有两种方法
  - 采用“L7”功能键
  - 从窗口的弹出式菜单中选择“CLOSE”选项

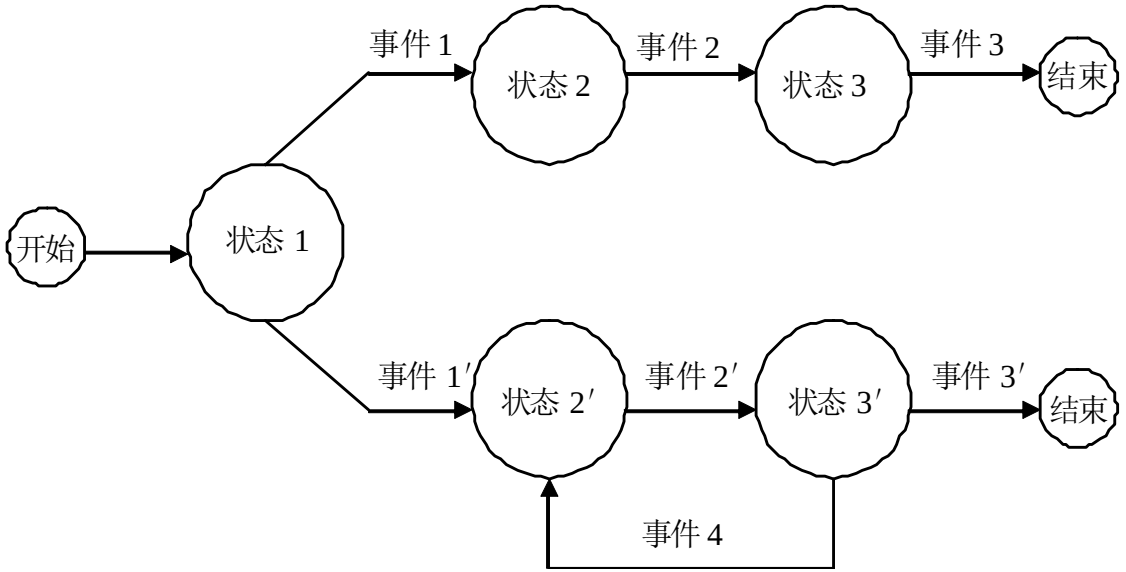
  - 现假设用户已经将手放在鼠标上，给出两种方式分别需要的操作，及计算二者所需的时间

- 动态特性建模

状态转移网

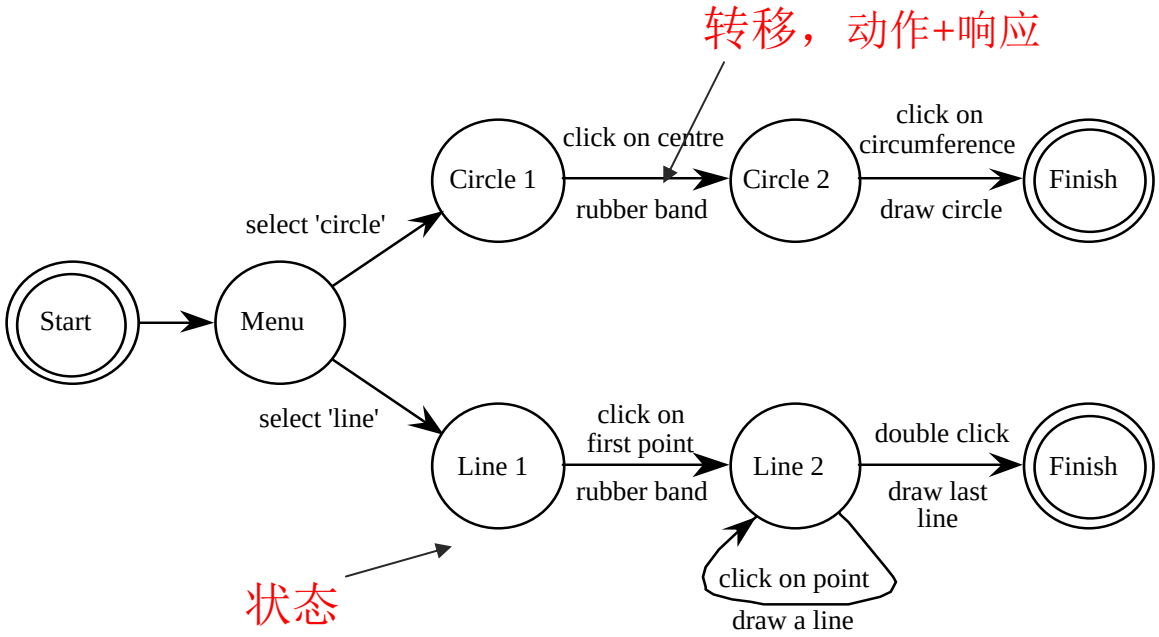
- 用于描述用户和系统之间的对话
  - 自20世纪60年代后期被使用
  - 可被用于探讨菜单、图标和工具条等屏幕元素，还可以展示对外围设备的操作
  - 适合表达顺序操作和循环操作
- 状态转移图
  - 最常用的状态转移网的形式
  - 有向图
  - 图中的结点表示系统的各种状态
  - 图中的边表示状态之间可能的转移

- 状态之间通过转移（用带方向箭头的线段表示）互相连接
- 转移被事件（转移线段上的标记）触发
- 伪状态——初始状态和终止状态
  - 是STNs的起始和终止
  - 可以与系统的其他部分相连接。



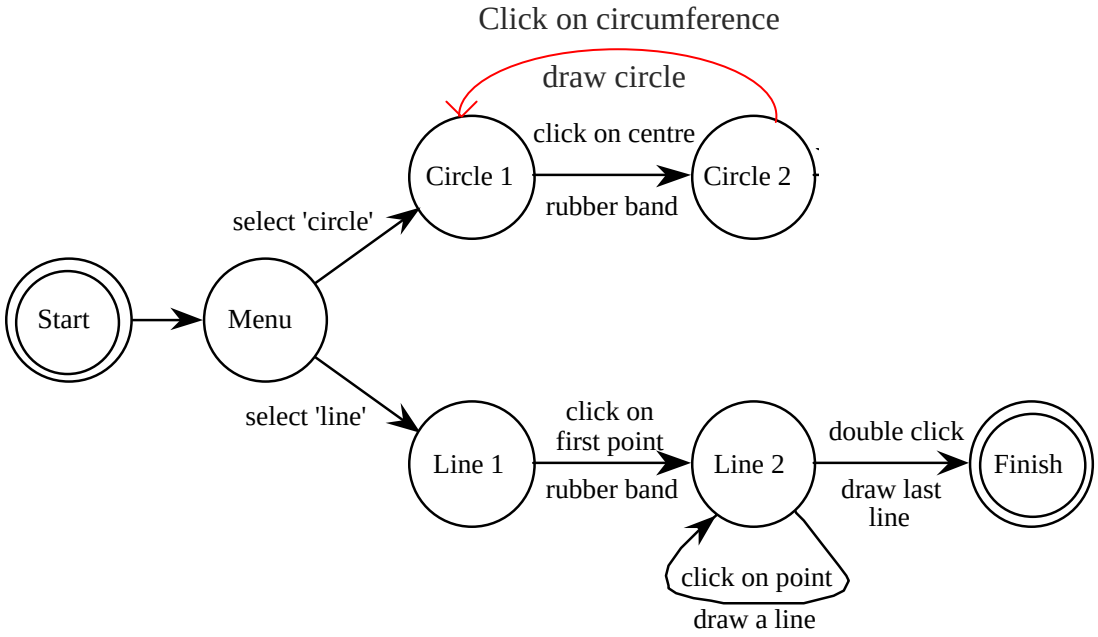
鼠标画图工具举例

- 菜单包含两个选项
  - Circle和Line



STN能够表达用户动作和系统响应的顺序.

- 前一个状态转移图表达的对话只允许画一个圆
  - 每画一个圆都要选择菜单中的“circle”选项
  - 如何改变为选择一次可画任意多个圆？



三态模型

- 帮助设计者为特定交互设计选择合适的I/O设备
  - 三态模型能够体现设备间的关键差别
- 问题
  - 以下交互设备等价吗？
  - 为什么不同设备给用户的感觉不同？

三态模型(Three-State Model)

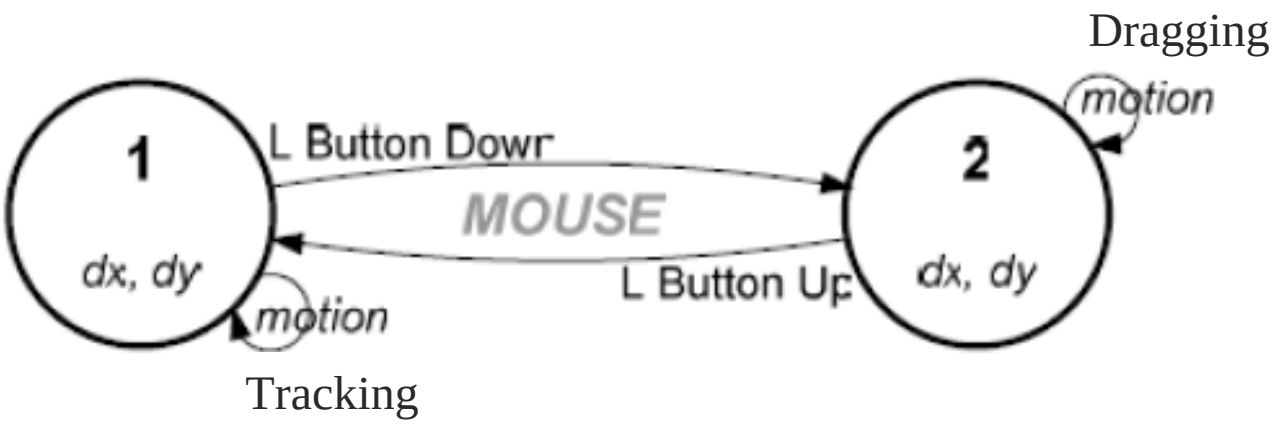
- Buxton提出的，用于对指点设备建模
- 将指点设备的操作使用状态转移来表示
- 指点设备状态
  - 无反馈运动（状态0）
  - 跟踪运动（状态1）
  - 拖动运动（状态2）

State	Description
0	*Out Of Range:* The device is not in its physical tracking range.
1	*Tracking:* Device motion moves only the cursor.
2	*Dragging:* Device motion moves objects on the screen.

Fig. 1 Summary of states in Buxton's 3-state model. Adapted from (Buxton, 1990).

鼠标的三态模型

- 跟踪状态（1）：左键抬起
  - 拖动鼠标跟踪鼠标运动并更新鼠标位置
- 拖动状态（2）：左键按下
  - 文件夹在屏幕范围内被拖动

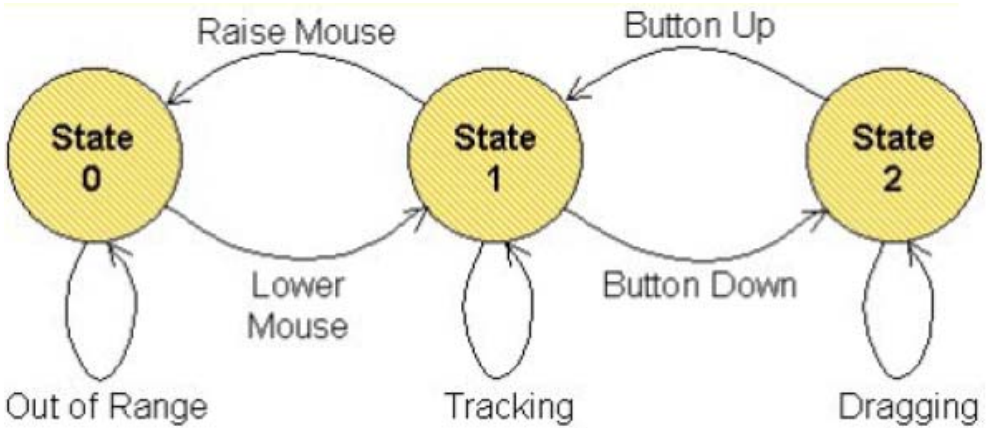


触摸板的三态模型

- 无反馈状态（0）：手指不接触触摸板
  - 系统不跟踪手指运动
- 跟踪状态（1）：手指接触触摸板
  - 系统跟踪手指运动
- 在没有其他组件配合的条件下触摸板没有状态2

鼠标的三态模型-2

- 问题：如何获得状态0？
  - 将鼠标拿起（模仿手写笔或光笔）
  - Buxton(1990)认为鼠标拿起这种状况是未定义的
  - Mackenzie(2003)为鼠标定义了s0

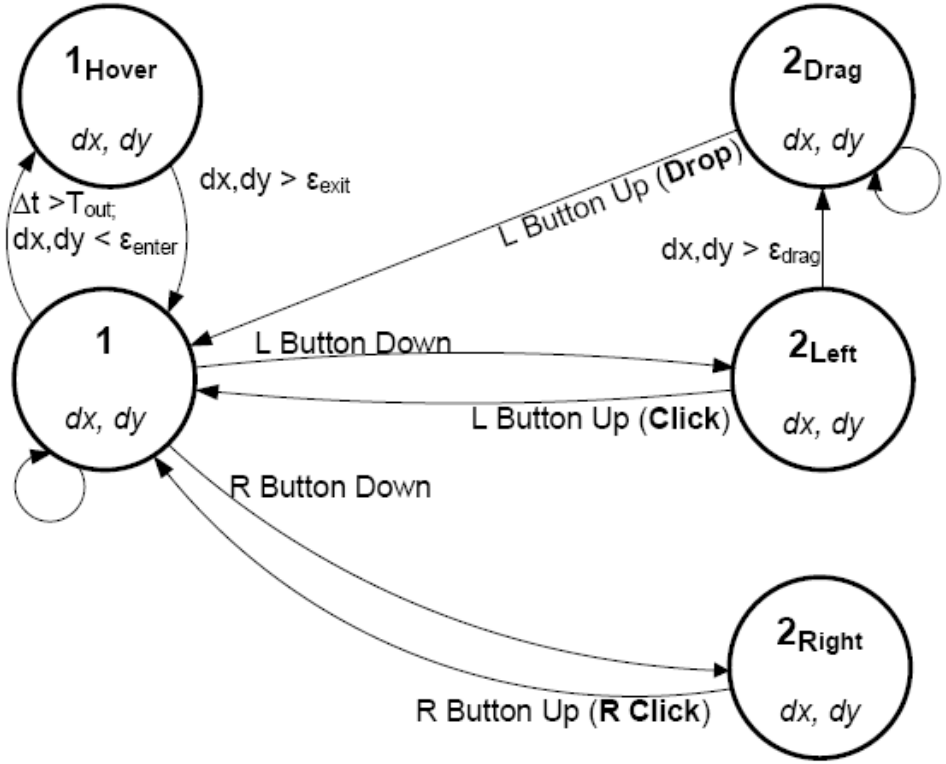


比较

- 鼠标和手写笔等在s0状态下的行为是存在差异
- 鼠标或触摸板
  - 系统保存鼠标被拿起前的位置和手指离开触摸板前的位置
  - 位置独立s0
- 手写笔或光笔
  - 重新参与到系统后指示设备当时的位置依赖于设备的位置
  - 位置依赖s0

完整交互建模

- 通过扩展三态模型，可捕捉所有交互操作



语言模型

BNF, Backus-Naur Form

- 语言模型
  - 用户和计算机的交互通常是通过一种语言进行考察的
  - BNF语法常用于说明对话
  - 目的在于理解用户的行为和分析认知界面的难度
- BNF是语言模型的代表
  - BNF最早用于描述ALGOL 60 编程语言
  - 现在，所有程序设计语言书籍都使用BNF来定义编程语言的语法规则
  - 系统对话很容易使用BNF规则来描述

画线图形系统举例

- 名称类型
  - 非终止型：小写字母
  - 终止型：大写字母
- 符号“::=”读作“定义为”
- 操作符
  - “+”（序列）和“|”（选择）

draw-line ::= select-line + choose-points + last-point

select-line ::= position-mouse + CLICK-MOSUE

choose-points ::= choose-one | choose-one + choose-point

choose-one ::= position-mouse + CLICK-MOUSE

last-point ::= position-mouse + DOUBLE-CLICK-MOUSE

position-mouse ::= empty | MOVE-MOUSE + position-mouse

界面分析

- 方法一：计算规则的数目
  - 规则越多，界面就越复杂
  - 缺点：对于描述界面的确切方式是相当敏感的
    - 如可以定义一个规则代替choose-points和choose-one

choose-points ::= position-mouse + CLICK-MOUSE

| position-mouse + CLICK-MOUSE + choose points

- 方法二：计算“+”和“|”操作符的数目
  - 更健壮
  - 但使较复杂的单个规则处于不利地位
  - 且可能存在错误判断
    - 如select-line和choose-one的复杂度相同

系统模型

为什么研究系统模型？

- 出发点
  - 为评估系统的可用性，需要知道系统做的是什么
- 分类
  - 基于模型的标记法，描述系统的状态和操作
    - Z标记法
  - 代数标记法，描述动作序列的作用
  - 时序和义务逻辑，描述事情什么时候发生，以及由谁负责

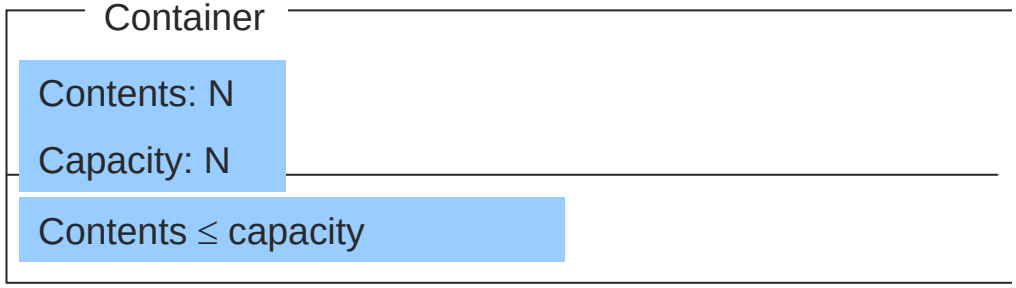
面向模型的标记法

- 源于20世纪70年代后期
- 反映了用在真实编程语言中的多种结构
- 提供一种描述软件系统行为的方法
- 用一种和系统怎样编程密切相关的方式和更抽象的语言描述软件系统行为
  - 允许设计人员在设计的早期不考虑机器或者实现
  - 使设计或者规约的推理更为严密
- Z标记法和VDM标记法

Z (Zed) 标记法

- Developed At Oxford in 1970's
- 基于集合和函数
  - 最简单的集合对应于编程语言中的标准类型
  - 如实数R、整数Z和自然数N等
- 非标准类型定义举例
  - 显式列出集合中有限的可能值就可以定义新集合
  - 举例：包含所有图形包中使用的集合形状(线、椭圆和矩形)的集合
    - *Shape_type* ::= *Line* | *Ellipse* | *Rectangle*
  - 举例：一个所有可能按键的集合
    - *Keystroke* ::= *a* | *b* |...| z | A | ... | 9 | *Cursor_left…*

- 定义方式二
  - 用方括号括起集合符号，如[*Keystroke*]
  - 可表示集合的存在而不提供集合内容的定义
- 基本集合可构建复杂集合
  - 如已命名无序元组、序列和函数等
  - 举例：空间中的点定义为实数的二元组(x,y)
    - *Point* ::= R*R
  - 已命名的无序元组称为模式Scheme



模式举例

- 几何形状
  - Shape_type*R*R*Point

Shape

type: Shape_type

wid: R

Ht: R

Center: Point

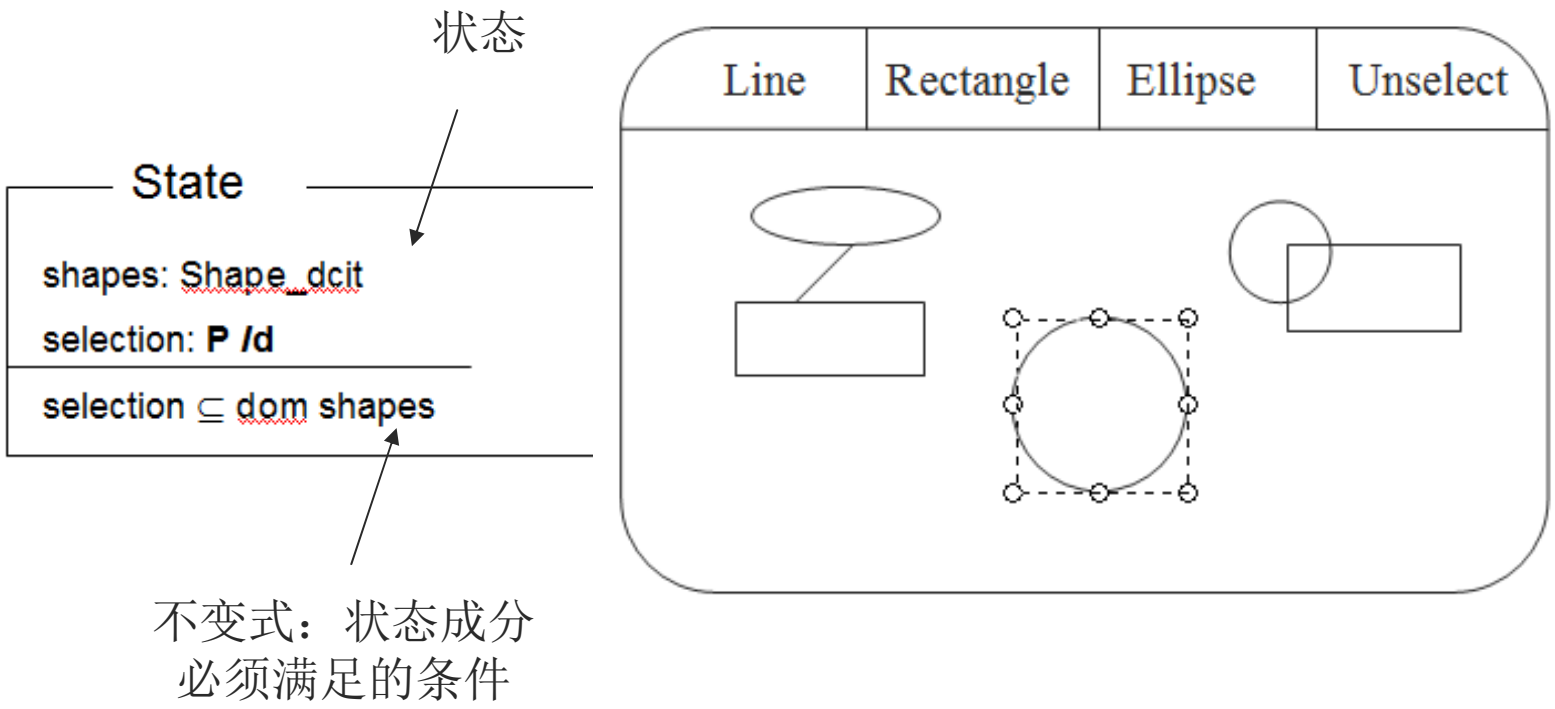
- 模式对应C语言中的结构类型

- 函数
  - 具有程序语言中标准计算的功能
  - 表达用户在某个图形编辑器的绘画中建立的形状集合时，Shape模式不能挑出单个单个或多个形状
  - 通过命名一个把形状和标识符映射起来的函数，可以表达一组相同形状

  - [Id]: 标识符集合，标注每个形状
  - *Shape*_dict ::= Id⤔Shape
  - 局部函数，不是每个ID都是*shapes*的有效参数
    - 如 dom *shapes* = {5,1,7,4}
    - 则shapes(3)无效

状态和不变式

- 绘图系统举例



- 初始状态模式Init

Init

State

dom shapes = { }

selection = { }

- 简化

Init

State

dom shapes = { }

操作符

- 举例：新建一个椭圆操作符
  - 由实现人员解决新添标识符如何提供给操作符

NewEllipse

State
State'
newid' ?: Id
newshape?: Shape

---

newid? ∉ dom shapes
newshape?. type = Ellipse
newshape?. wid = 1
newshape?. ht = 1
newshape?. center = (0,0)
shapes' = shapes ⋃ {newid?↦ newshape?}
selection' = {newid?}

- 举例：Unselect操作符
  - 使得当前选择对象为空
  - 注意：说明操作后形状词典保持原样很重要
    - 可用来判断最终程序的外部一致性

Unselect

State
State'

selection' = { }
shapes' = shapes

生日簿系统

- 记录用户生日的系统，当某人生日到来时发出提醒

- 功能
  - 添加一个人的姓名和生日
  - 保存该条信息
  - 重新检索信息

[NAME; DATE]

Birthday book

known: NAME

birthday: NAME ⇸ DATE

known ={}

- 一种可能的系统状态

known = { John; Mike; Susan }

birthday = { John 25-Mar,

Mike 20-Dec,

Susan 20-Dec }

- △ 符号表示该操作会使生日簿的状态发生改变

Add Birthday

△ Birthday Book

name?: NAME

date?: DATE

---

name? ∉ known

birthday' = birthday U { name? ⤔ date ?}

- Ξ 符号表示该操作不会修改生日簿中的值

Find Birthday

Ξ Birthday book

name?: NAME

Date! : DATE

---

name? ∈ Known

date ! = birthday(name?)

小结

- 预测模型
  - GOMS
    - KLM
  - Fitts
- 动态特性建模
  - 状态转移网
  - 三态模型
- 语言模型
  - BNF
- 系统模型
  - Z标记法