Use Case Scenarios — play by play of what the user does & sees.

Each line has 4 cols.

User Action | Immediate Feedback (transient) | Visible Result | Model State Change

- click on line tool
- line tool button is selected & cursor changes to \( \Rightarrow \)
- press mouse button in canvas area
- mouse drags to another point
- see a line from original pt. to current position

(remember the initial point of the line)
(remember the 2nd point)
more dragging
mouse release

line moves
(temporary line goes away)

final line is displayed
line is added to the shapes list
Data Streaming

Saving it all in memory & putting in a file at the end is not an option.

Diagram:
- Computer
- External drive (Terabyte)
- Filter
- Disk
- Producer/Consumer
- Application (Appl.)
throughput — how much data \[\frac{\text{bits}}{\text{sec}}\] \[\frac{\text{songs}}{\text{hour}}\] \[\frac{\text{how much data}}{\text{how much time}}\]

latency — how long things take — wait time

disk latency — \(x\) ms

memory latency — \(y\) ns

mask latency by pipelining

jitter — temporary delays due to emptying out the incoming buffer

[Diagram of a server and streaming media]
1. As user takes the survey, stream user events to a file

   Event Recorder
   AWT Event Loop
   Thread
   enqueue
   (Un) Bounded Buffer

   Second Thread
   Writing to disk
   dequeue

2. Psychologist wants to analyze behavior during survey taking

   EventPlayer
   pull from Bounded Queue
   Bounded Buffer

   Read from disk
To avoid interference w/ mouse events during playback —
let event player also put something on top of event queue.
Interprocess Communication

Threads
- run within a single app.
- with a single shared heap
- communication is through shared state

Processes
- run in different address spaces (protected by OS)

Communication
1. O.S.-supported shared memory
2. Message passing
Abstractions
1. Streams
2. Socket — network connection to another process

IP = Internet Protocol
agreed-upon way of interacting

IP address
A send(x)
B

Network

z = receive()
IP — datagrams (U.S. Postal Service on steroids)

"Best effort"

- message loss
- garbled messages

 Sender

\[ X \quad X \quad Y \quad Y \quad W \quad \leftarrow \quad X \quad Y \quad Z \quad W \]  

TCP — Transmission Control Protocol

1. handles message loss/duplication → reliable stream
2. rate control (friendly to other network traffic) (FIFO)
TCP
message loss & reordering

Idea 1: put a sequence # in header of each message

Receiver
1 3 4 2
reordering is a local operation

Sender
1 2 3 4

Idea 2: retransmit missing packets

a. Negative Acknowledgment (NACK)
NACK 2 \rightarrow \rightarrow

b. Positive Acknowledgments (ACK)
\leftarrow 2
ACK 6 — I got 1, 2, 3, 4, 5, 6
If sender sent 7, 8, 9, could retransmit

Protocol
  Receiver
    constantly ACK highest complete seq. #
    reorder packets locally
  Sender
    send a fixed amount of stuff
    & waits for ack before sending more
should I send yes, if got ack for something bigger
TCP slow start

Window Size

time