CONSTRUCTOR UNIVERSITY

Foundations of Graphical User Interfaces

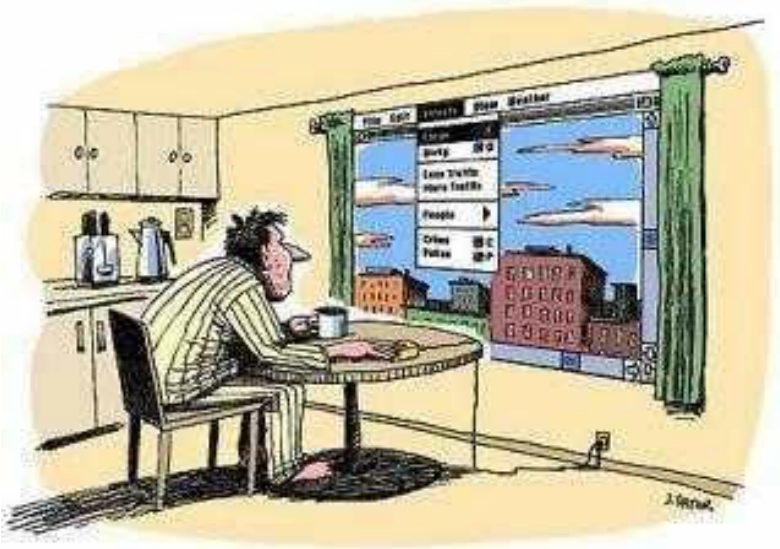
Credits:
Stanford HCI,
Pressman

Instructor: Peter Baumann

email: pbaumann@constructor.university

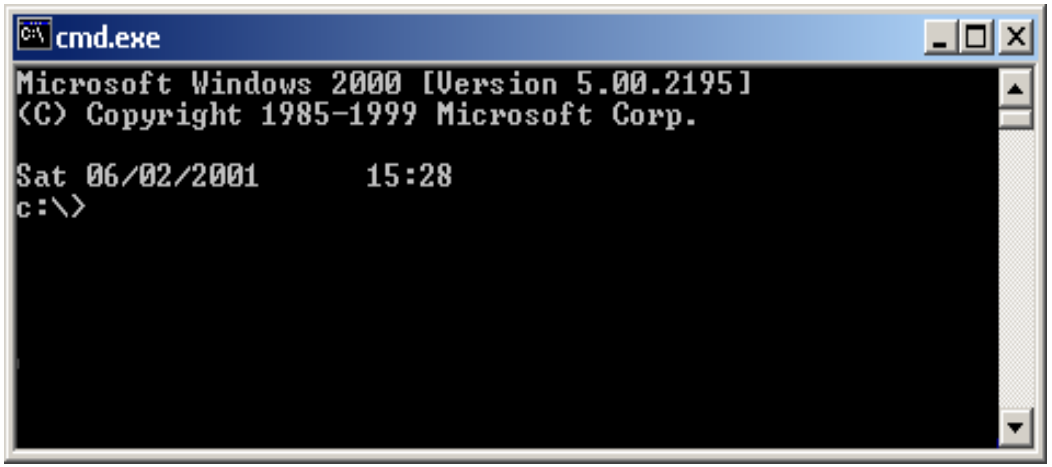
tel: -3178

office: room 88, Research 1



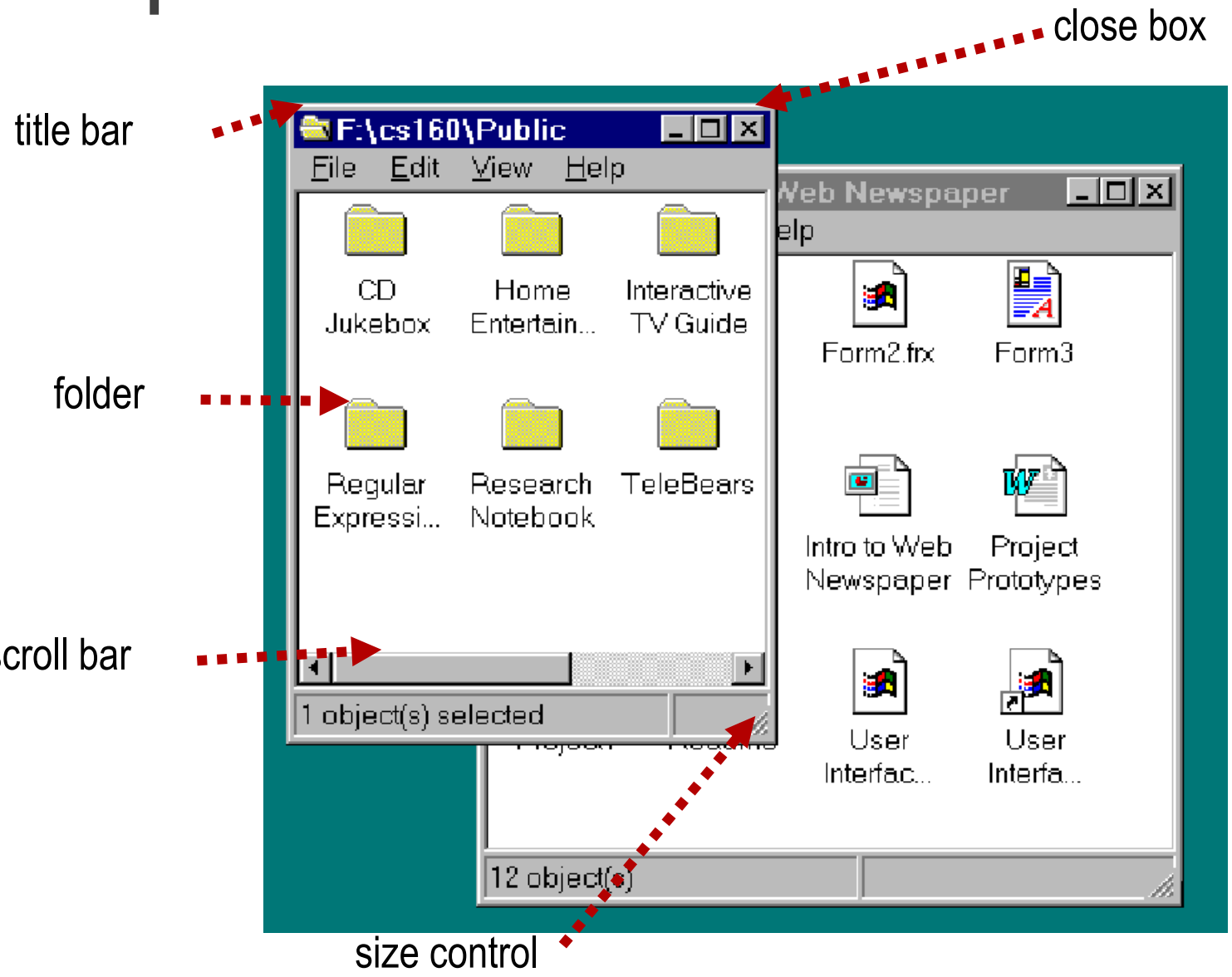
User/Application Interaction

- Program takes control, prompts for input → user waits



- Not suitable for highly interactive applications

Example Interactions



Modern GUI Systems

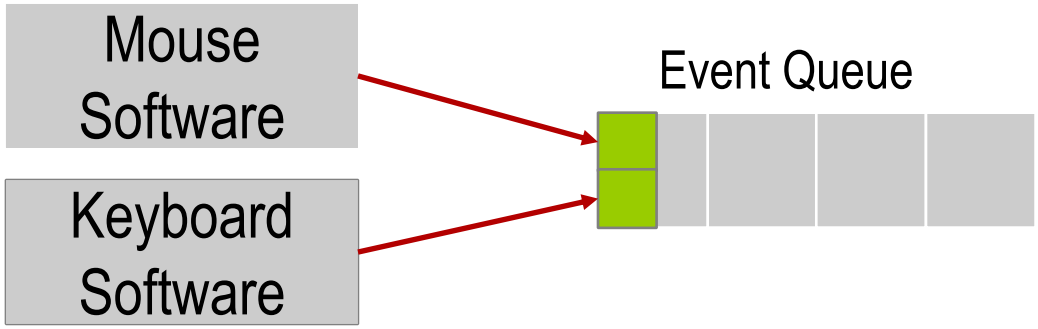
- Three concepts:
  - Event-driven programming
  - Widgets
  - Interactor Tree
- Describes how most GUIs work
  - Coined with SmallTalk
  - Closest to Java
  - But similar to Windows, Apple, Android, ...

Event-Driven Programming

- Instead of the user waiting on program, program waits on the user
- All communication from user to computer is done via “events”
  - “mouse button went down”
  - “item is being dragged”
  - “keyboard button was hit”
- Events have:
  - type of event
  - mouse position or character key + modifiers
  - *...plus possible additional, application-dependent information*

Event-Driven Programming

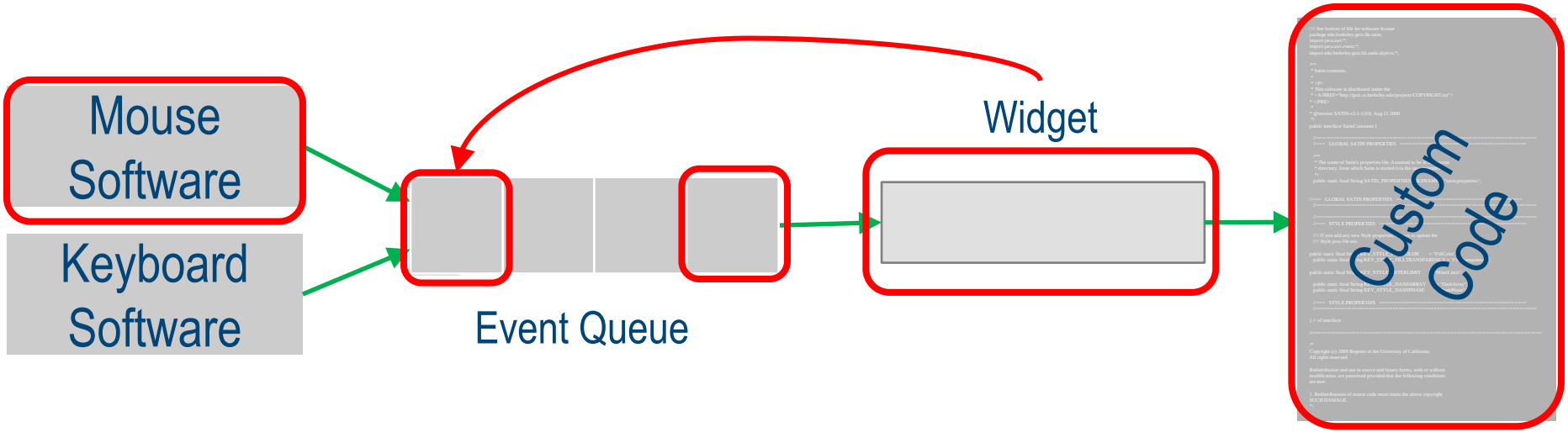
- All events generated go to a *single* event queue
  - provided by operating system
  - ensures that events are handled in the order they occurred
  - hides specifics of input from apps



Widgets

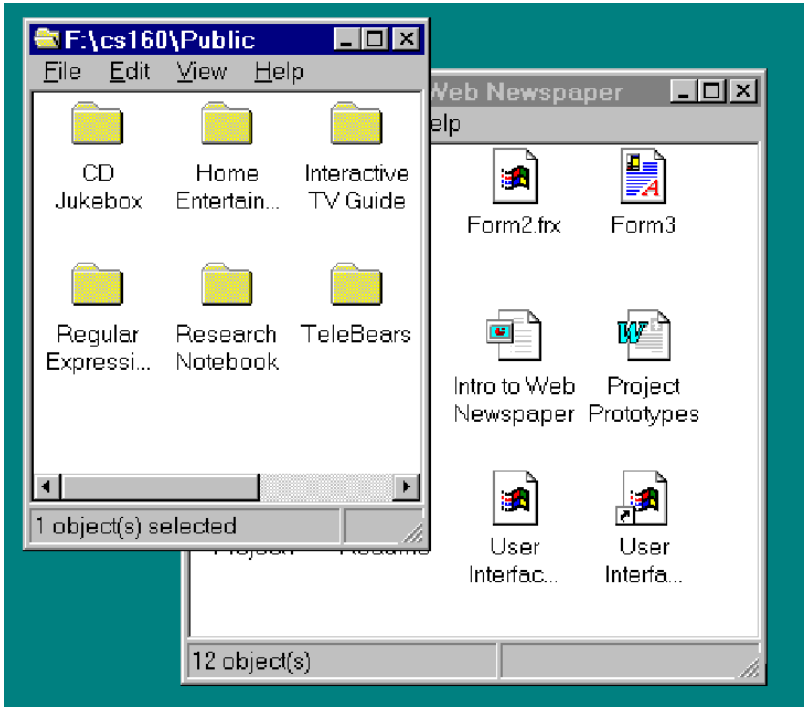
- Widget = (reusable) interactive object
  - "window gadget"
- Widget tasks:
- Handle certain events:
  - widgets say what events they are interested in
  - event queue sends events to the "right" widget
- Update appearance
  - e.g. button up / button down
- Generate some new events, eg
  - "button pressed"
  - "window closing"
  - "text changed"
- But these events are sent to interested listeners instead
  - custom code goes there

Widget in Action



Interactor Tree

- Decompose interactive objects into a tree

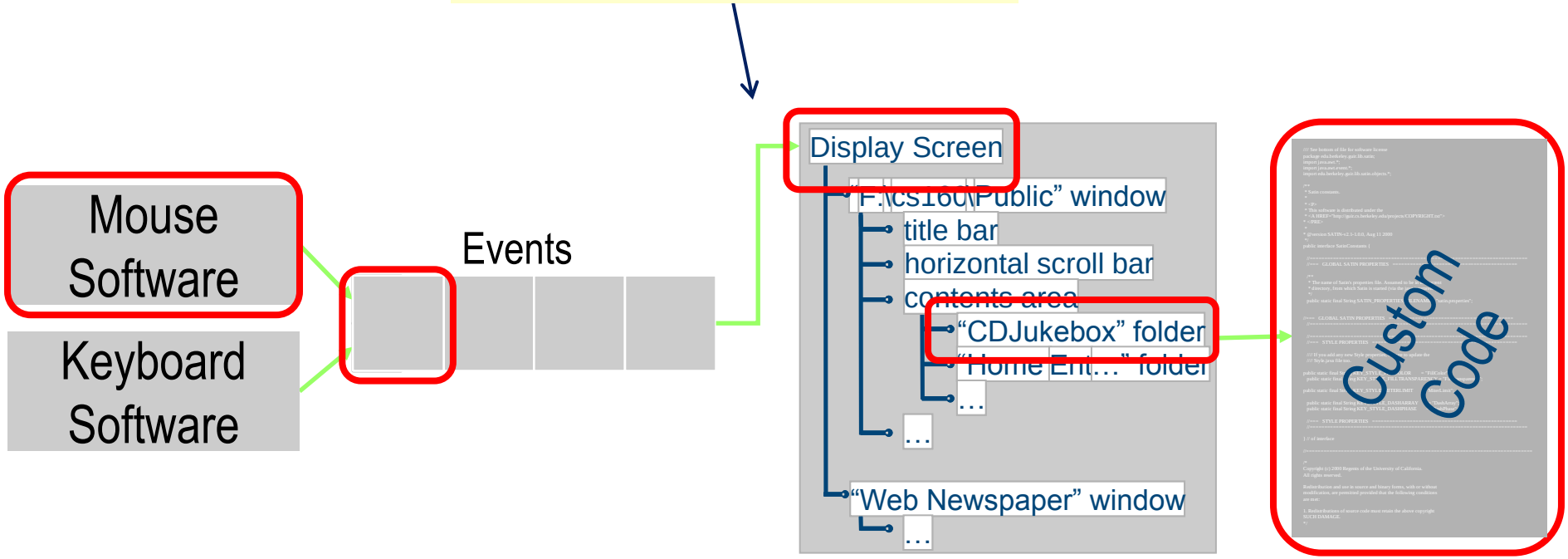


Display Screen

- "F:\cs160\Public" window
  - title bar
  - horizontal scroll bar
  - contents area
    - "CDJukebox" folder
    - "Home Ent…" folder
    - …
  - …
- "Web Newspaper" window
  - …

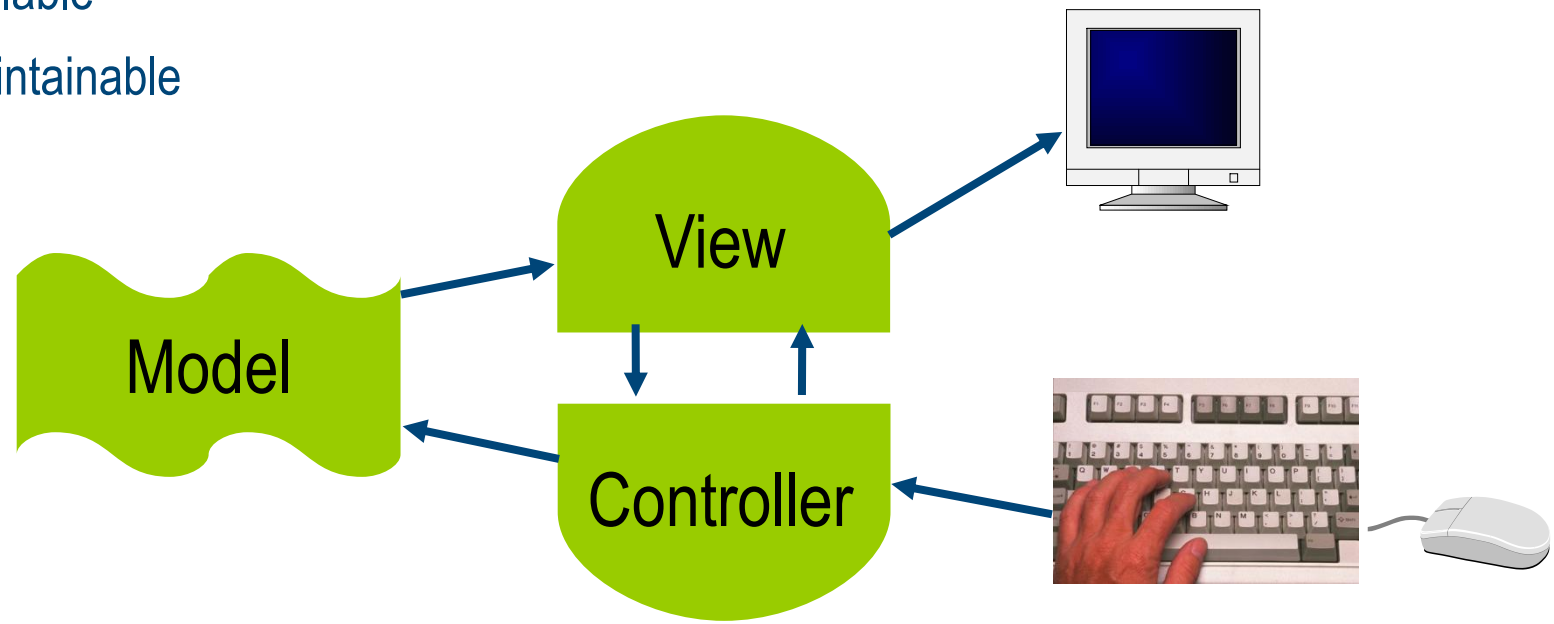
Main Event Loop

```
while (app is running)
{
    get next event;
    send event to right widget;
}
```

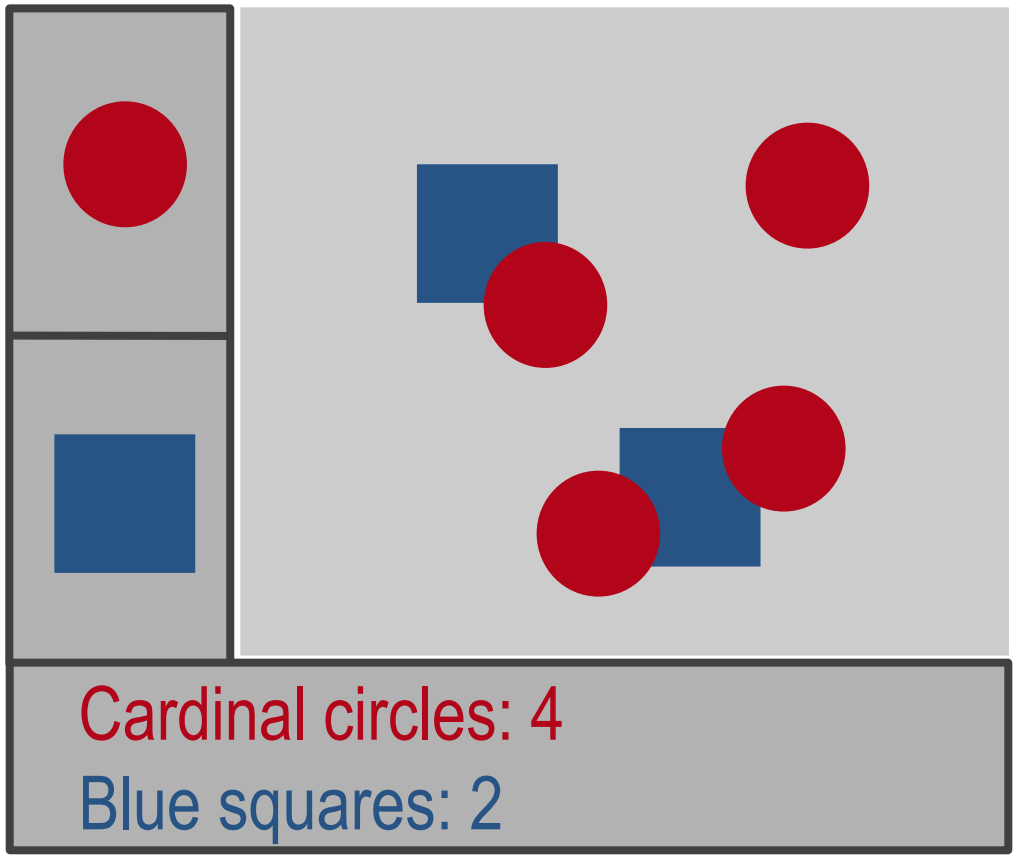


Model-View-Controller

- Architecture for interactive apps
  - introduced by Smalltalk developers at PARC
- Partitions application in a way that is
  - scalable
  - maintainable

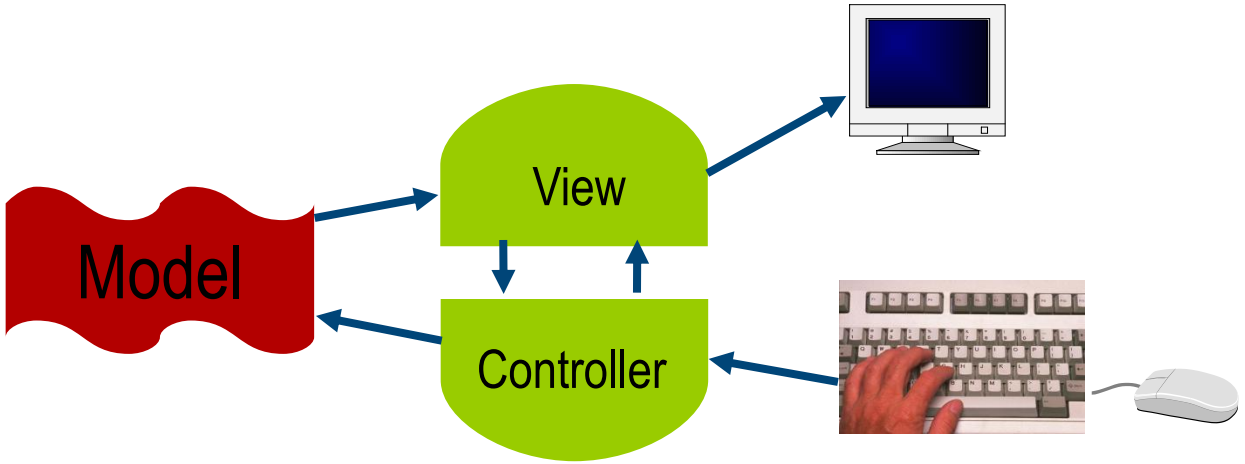


Example Application



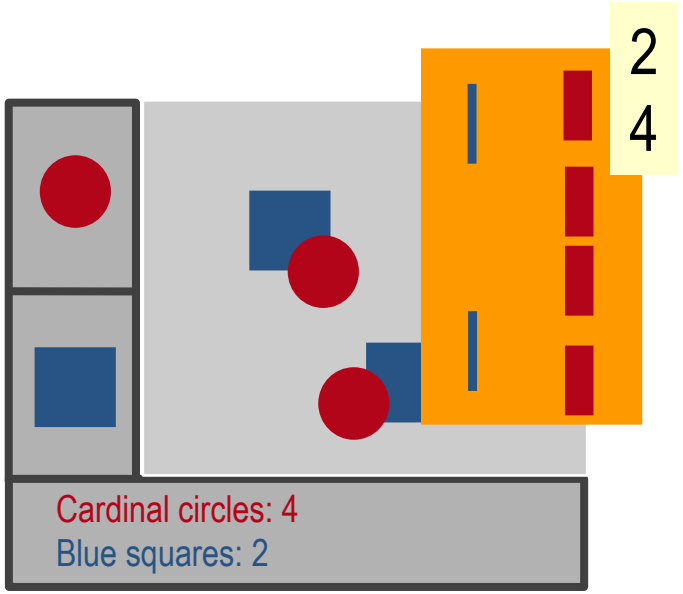
Model

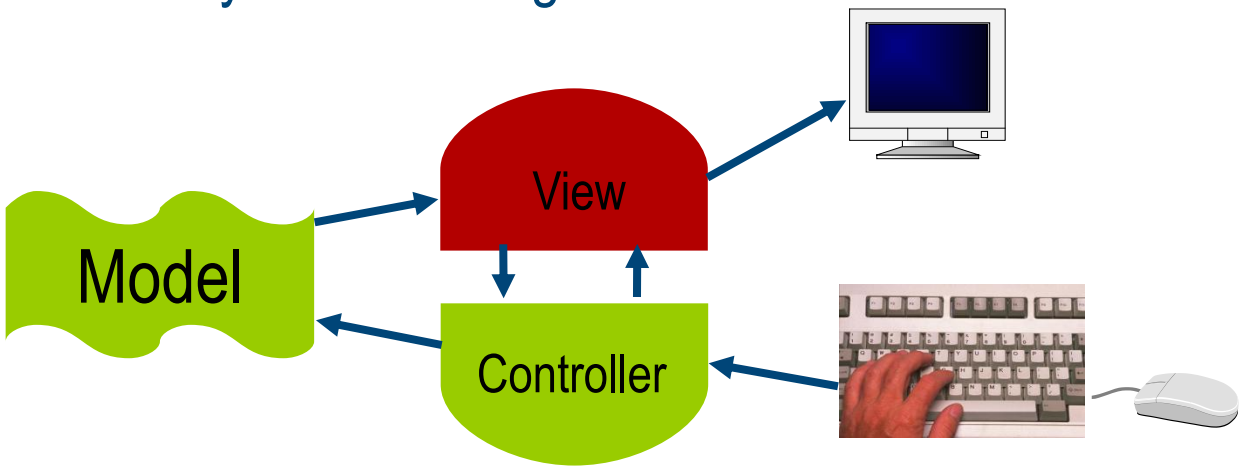
- Model = Information the app is trying to manipulate
- Representation of real world objects
  - circuit for a CAD program – ex: logic gates and wires connecting them
  - shapes in a drawing program – ex: geometry and color



View

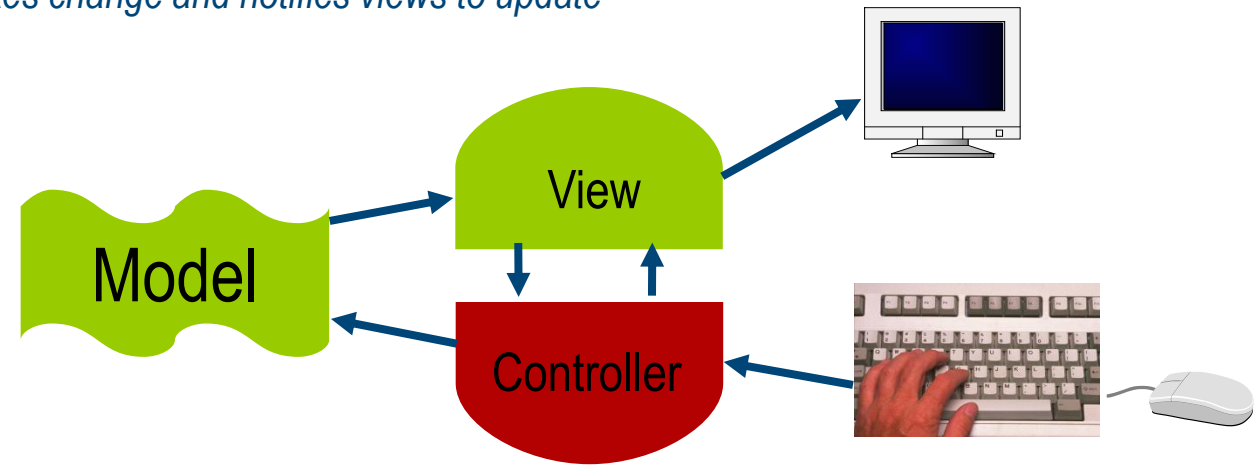
- Implements a (visual) display of the model
  - also audio/speech, alarms, text messages, …
- May have multiple views
  - e.g., shape view and numerical view
- Each view notified on any model change





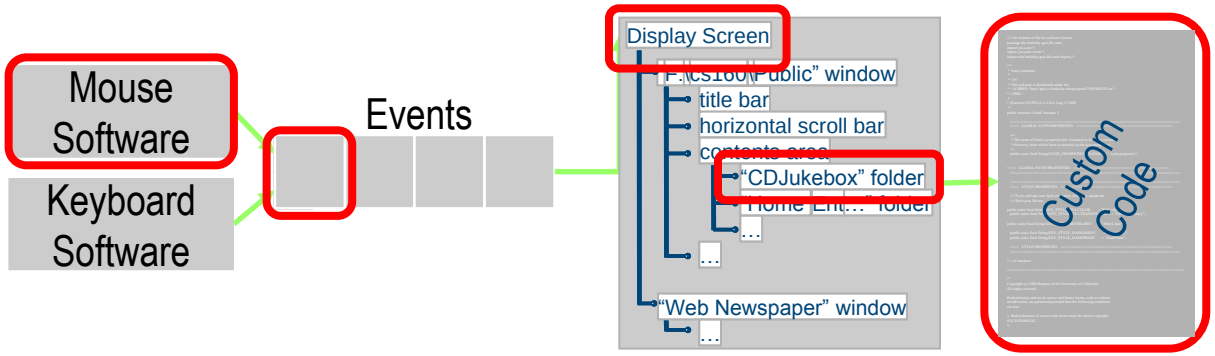
Controller

- Receives all input events from the user
- Decides what they mean and what to do
  - communicates with view to determine which objects are being manipulated (e.g., selection)
  - calls model methods to make changes on objects
    - *model makes change and notifies views to update*



Summary

- Event-driven programming, widgets, event loop



- Model-View-Controller pattern as a GUI paradigm