COLLEGE OF
COMPUTER STUDIES

QUEZON CITY
UNIVERSITY

WEEK 12 -13
Implementation Support

HCI101 - INTRODUCTION TO HUMAN COMPUTER INTERACTION

v2020

Implementation support

programming tools
– levels of services for programmers
• windowing systems
– core support for separate and simultaneous usersystem activity
• programming the application and control of
dialogue
• interaction toolkits
– bring programming closer to level of user perception
• user interface management systems
– controls relationship between presentation and
functionality

Introduction

How does HCI affect of the programmer?

Advances in coding have elevated programming

Hardware specific

→ interaction-technique specific

Layers of development tools

– windowing systems

– interaction toolkits

– user interface management systems

Elements of windowing systems

Device independence

-programming the abstract terminal device drivers

image models for output and (partially) input

- pixels
- PostScript (MacOS X, NextStep)
- Graphical Kernel System (GKS)
- Programmers' Hierarchical Interface to Graphics (PHIGS)

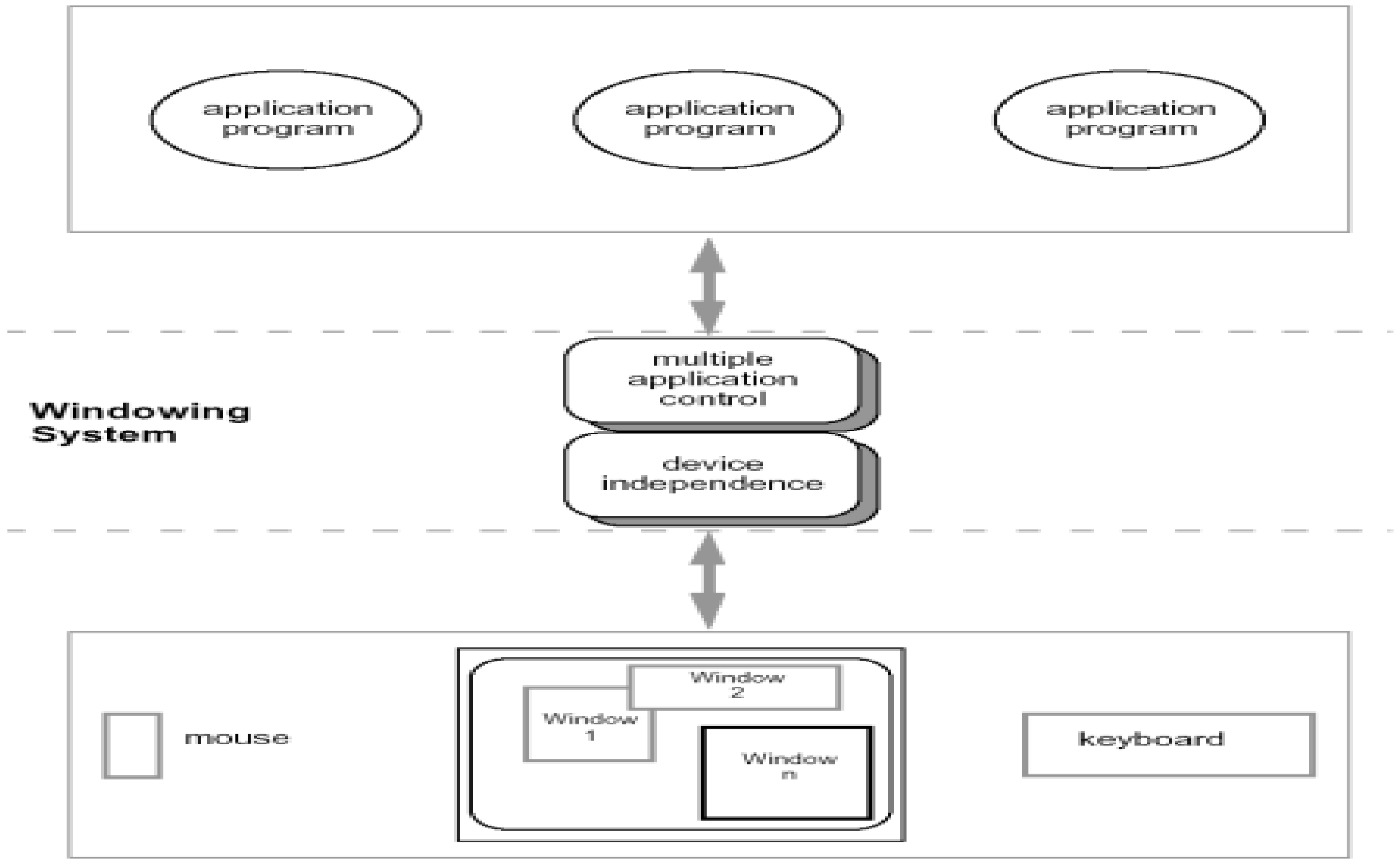
-Resource sharing

achieving simultaneity of user tasks

window system supports independent processes

isolation of individual applications

Roles of a windowing system



Architectures of windowing systems

Three possible software architectures

1. each application manages all processes

– everyone worries about synchronization

– reduces portability of applications

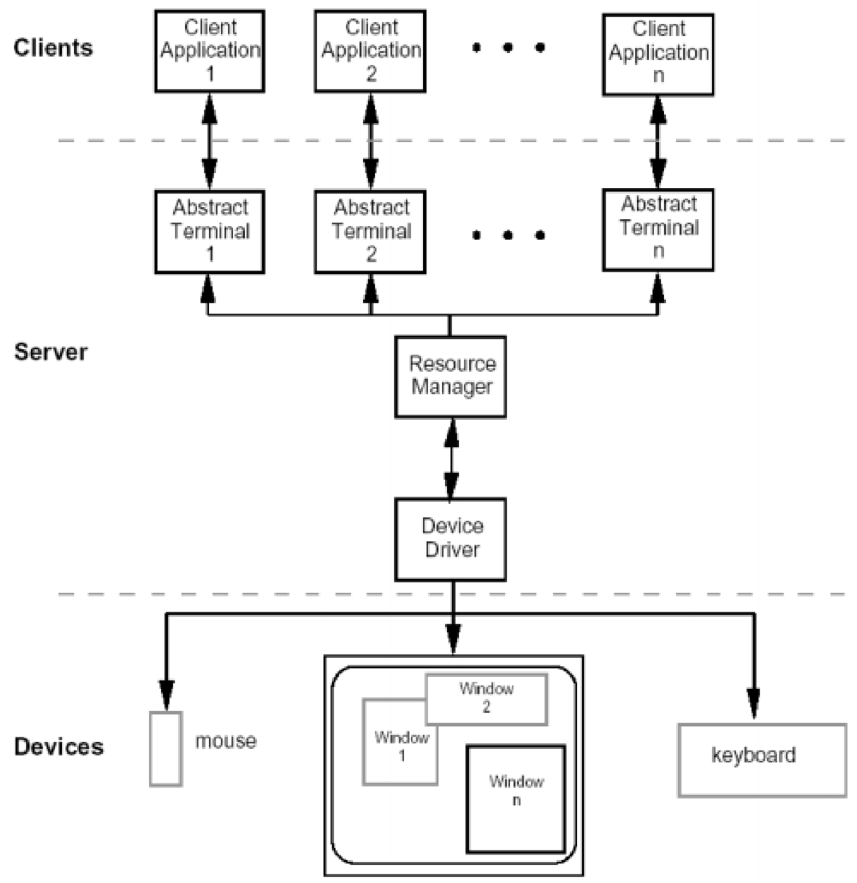
2. management role within kernel of operating system

– applications tied to operating system

3. management role as separate application

maximum portability

The client-server architecture



System Style Affects The Interfaces

– modal dialogue box

• easy with event-loop (just have extra read-event loop)

• hard with notification (need lots of mode flags)

– non-modal dialogue box

• hard with event-loop (very complicated main loop)

• easy with notification (just add extra handler)

beware!

if you don’t explicitly design it will just happen

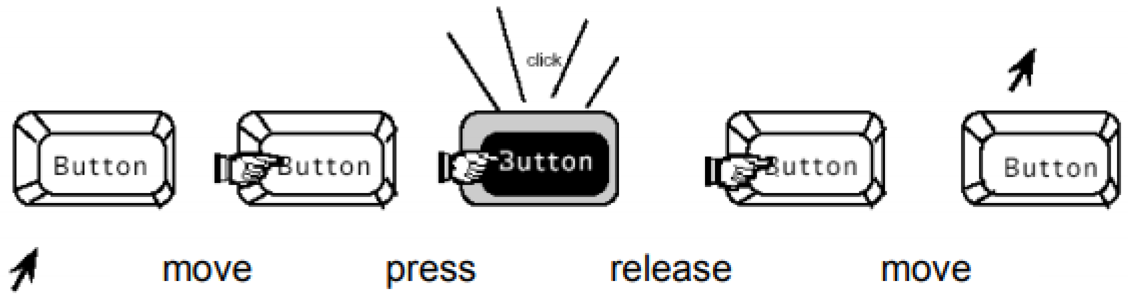
implementation should not drive design

Using toolkits

Interaction objects

– input and output

intrinsically linked



Toolkits provide this level of abstraction

– programming with interaction objects (or

– techniques, widgets, gadgets)

– promote consistency and generalizability

– through similar look and feel

– amenable to object-oriented programming

User Interface Management Systems (UIMS)

separation between application semantics and presentation

- improves:

– portability – runs on different systems

– reusability – components reused cutting costs

– multiple interfaces – accessing same functionality

– customizability – by designer and user

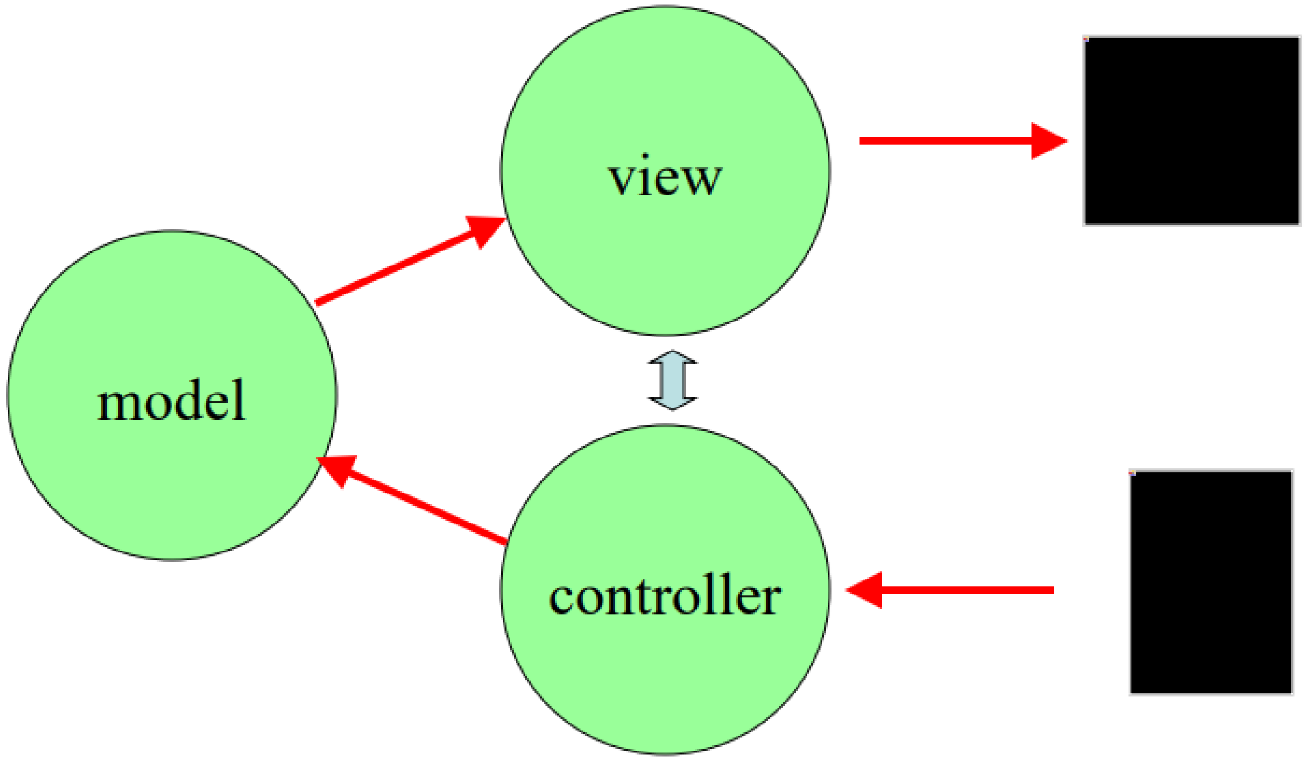
Implementation of UIMS

Techniques for dialogue controller

- menu networks
- grammar notations
- declarative languages
- graphical specification
- state transition diagrams
- event languages
- constraints

MVC

- model - view - controller



MVC issues

MVC is largely pipeline model:

input → control → model → view → output

• but in graphical interface

– input only has meaning in relation to output

e.g. mouse click

– need to know what was clicked

– controller has to decide what to do with click

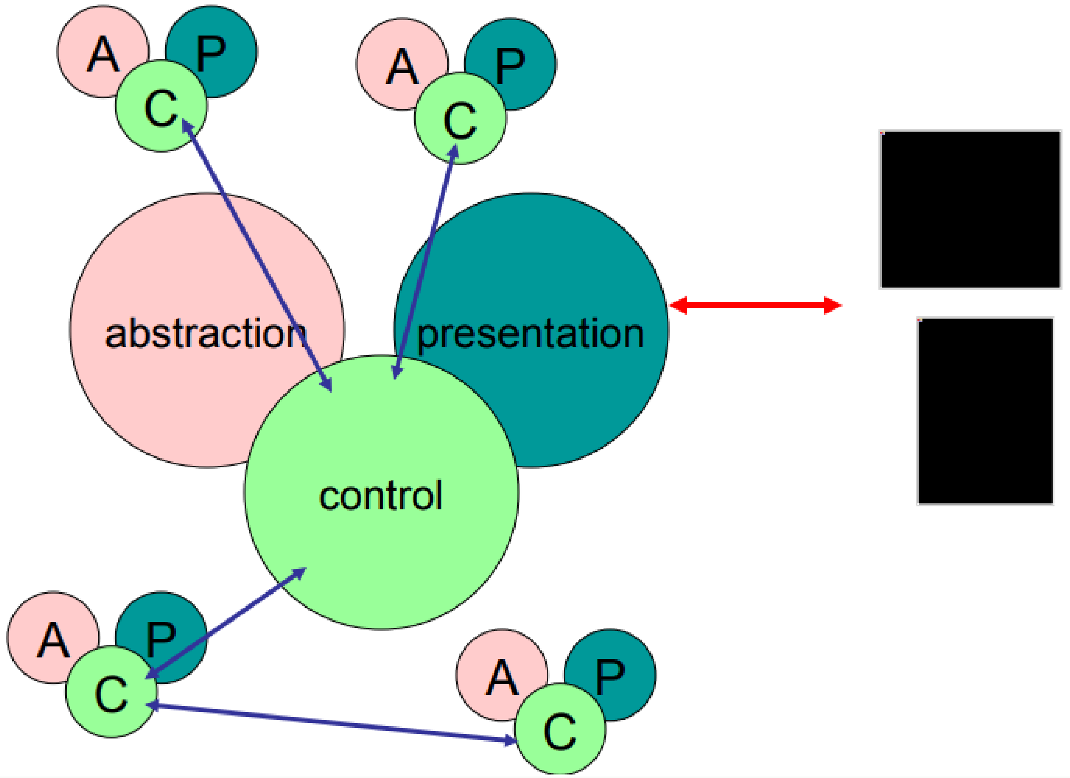
– but view knows what is shown where!

• in practice controller 'talks' to view

– separation not complete

PAC

- presentation - abstraction - control



PAC model

PAC model

– abstraction – logical state of component

– presentation – manages input and output

– control – mediates between them

• manages hierarchy and multiple views

– control part of PAC objects communicate

• PAC cleaner in many ways …

but MVC used more in practice

(e.g. Java Swing)

Graphical Specification

what it is

– draw components on screen

– set actions with script or links to program

• in use

– with raw programming most popular technique

– e.g. Visual Basic, Dreamweaver, Flash

• local vs. global

– hard to ‘see’ the paths through system

– focus on what can be seen on one screen

The drift of dialogue control

internal control

(e.g., read-evaluation loop)

- external control

(independent of application semantics or presentation)

- presentation control

(e.g., graphical specification)

Summary

Levels of programming support tools

- Windowing systems
  - device independence
  - multiple tasks
- Paradigms for programming the application
  - read-evaluation loop
  - notification-based
- Toolkits
  - programming interaction objects
- UIMS
  - conceptual architectures for separation
  - techniques for expressing dialogue