

# Introduction to Computer Graphics

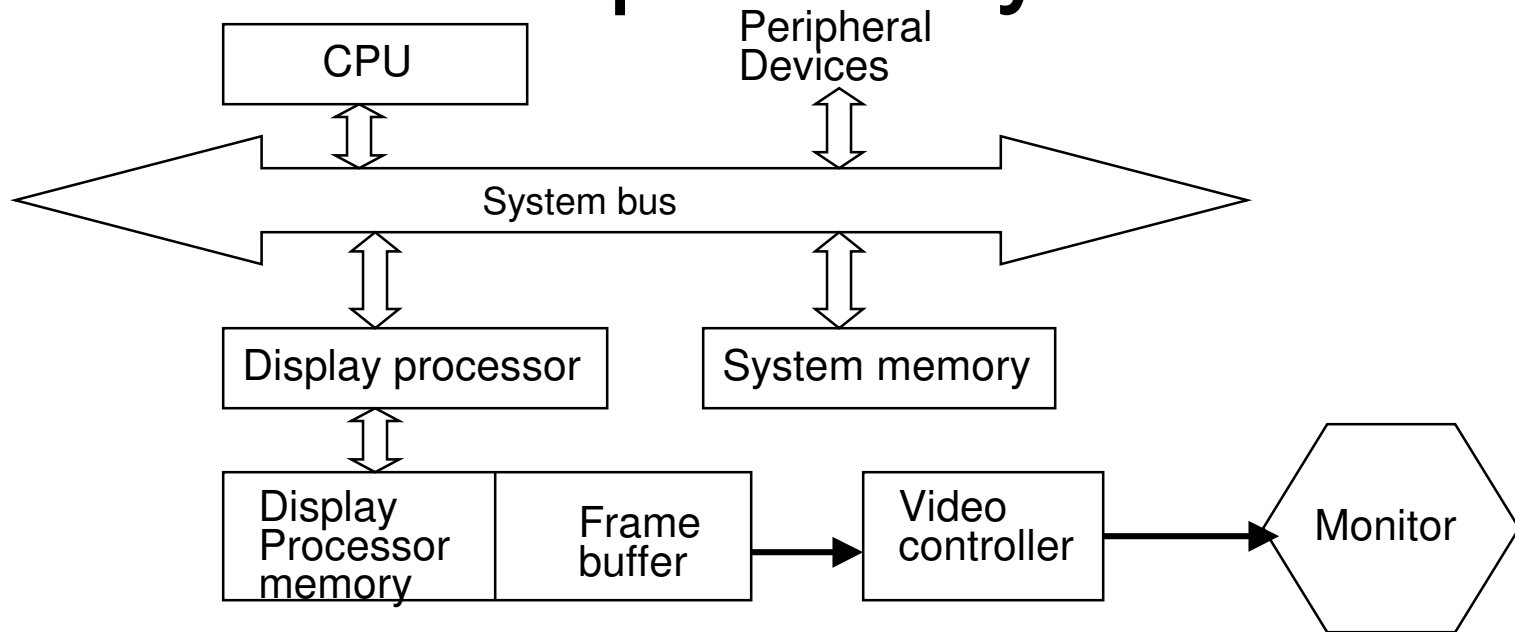
Farhana Bandukwala, PhD

Lecture 3: Graphics System

# Outline

- Raster system architecture
- Input and Output Devices & Interaction  
(Angel, Section 3.2)
- Client/Server operating model (Angel, Section 3.3)
- Graphics pipeline

# Raster Graphics System



- Peripheral display processor for performance
- Display memory: data plus rasterizer software
- Frame buffer: image to display
- Video controller: low level monitor specific driver

# Input devices: locators

- Absolute devices: digitizing drawings
  - Data tablet
  - Touch panel
- Relative devices: navigate arbitrarily large space
  - Mouse
  - Trackball
  - Joysticks
- Direct (light pen) vs indirect (mouse)
- Continuous (mouse) vs discrete (cursor control keys)

# Input devices: others

- Keyboard
- Choice devices: function keys
- Voice recognizers
- 3D devices
  - Joysticks
  - Spaceball
  - Data Glove

# Common output devices

- CRT
  - Electrons directed towards phosphor-coated screen which emits light at discrete points
- LCD
  - Polarizing crystalline molecules
- Active matrix panels
  - LCD with a transistor at each point
- Plasma panels
  - Array of tiny neon bulbs -> no refresh!

# Interaction tasks

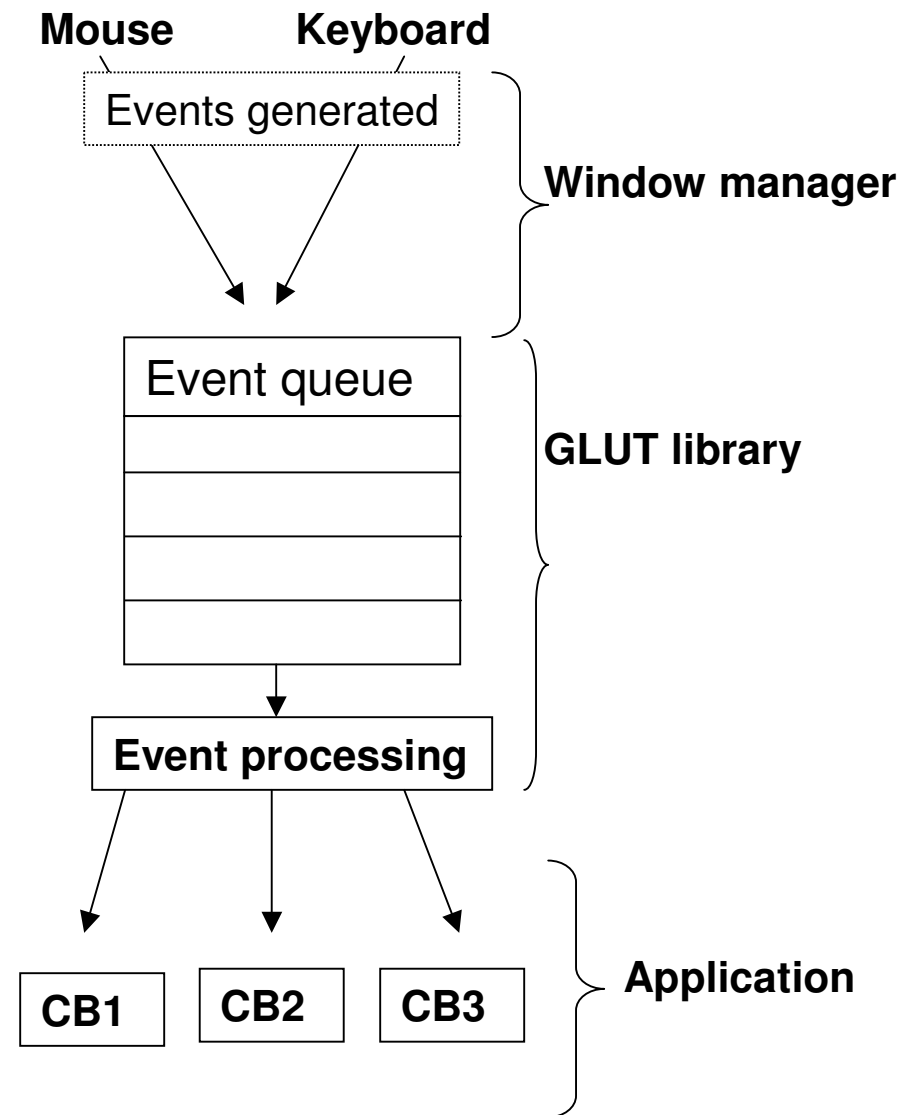
- Positioning
  - Coordinate systems, resolution
- Selection
  - Choice sets: objects, properties, commands
- Text input
  - Application does not interpret input
- Quantification
  - Numeric range

# Interaction handling models

- Request mode:
  - Application requests input from device
  - Graphics package returns control after user “triggers”
  - One device triggered at a time
- Sample mode
  - Application polls all devices
  - Input at other devices can be missed while processing particular one
- Event mode
  - Input is accepted asynchronously from different devices at once
  - Event queue may contain contradicting events

# Event Processing

- Asynchronous events received from different devices
- Application needs to constantly query event queue (`glutMainloop()`)
- Each event processed by appropriate function (callback) (`glutDisplayFunc()`, `glutMouseFunc()`, `glutKeyboardFunc()`, etc.)
- Callback responsible for state-based processing



# Event processing (example)

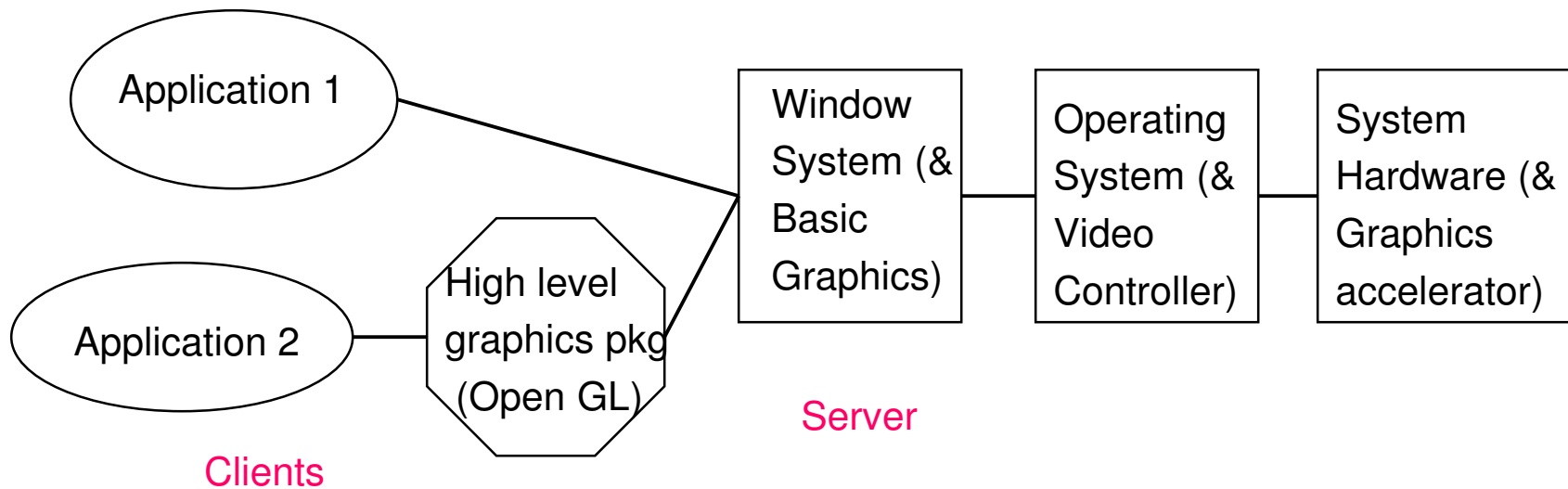
1. Create menu & Register callback `glutCreateMenu(void (*func)(int entryType))`
2. Add Menu Entries `glutAddMenuEntry(char *name, int entryType)`
3. Implement callback function
  - Performs different actions depending on value of `entryType`

```
#define MENU_DISPLAY 1
#define MENU_EXIT 2
// Menu callback function
void menuCB(int menuEntry)
{
    switch (menuEntry)
    {
        case MENU_DISPLAY:
            ...
            break;
        case MENU_EXIT:
            ...
            exit(0);
            break;
    }
}
int main(int argc, char ** argv)
{
    ...
    // Create menu and register callback
    int menuID = glutCreateMenu(menuCB);
    // Add Menu Entries
    glutAddMenuEntry("Display",MENU_DISPLAY);
    glutAddMenuEntry("Exit",MENU_EXIT);
    ;
    // Process events
    glutMainLoop();
}
```

# Client server model

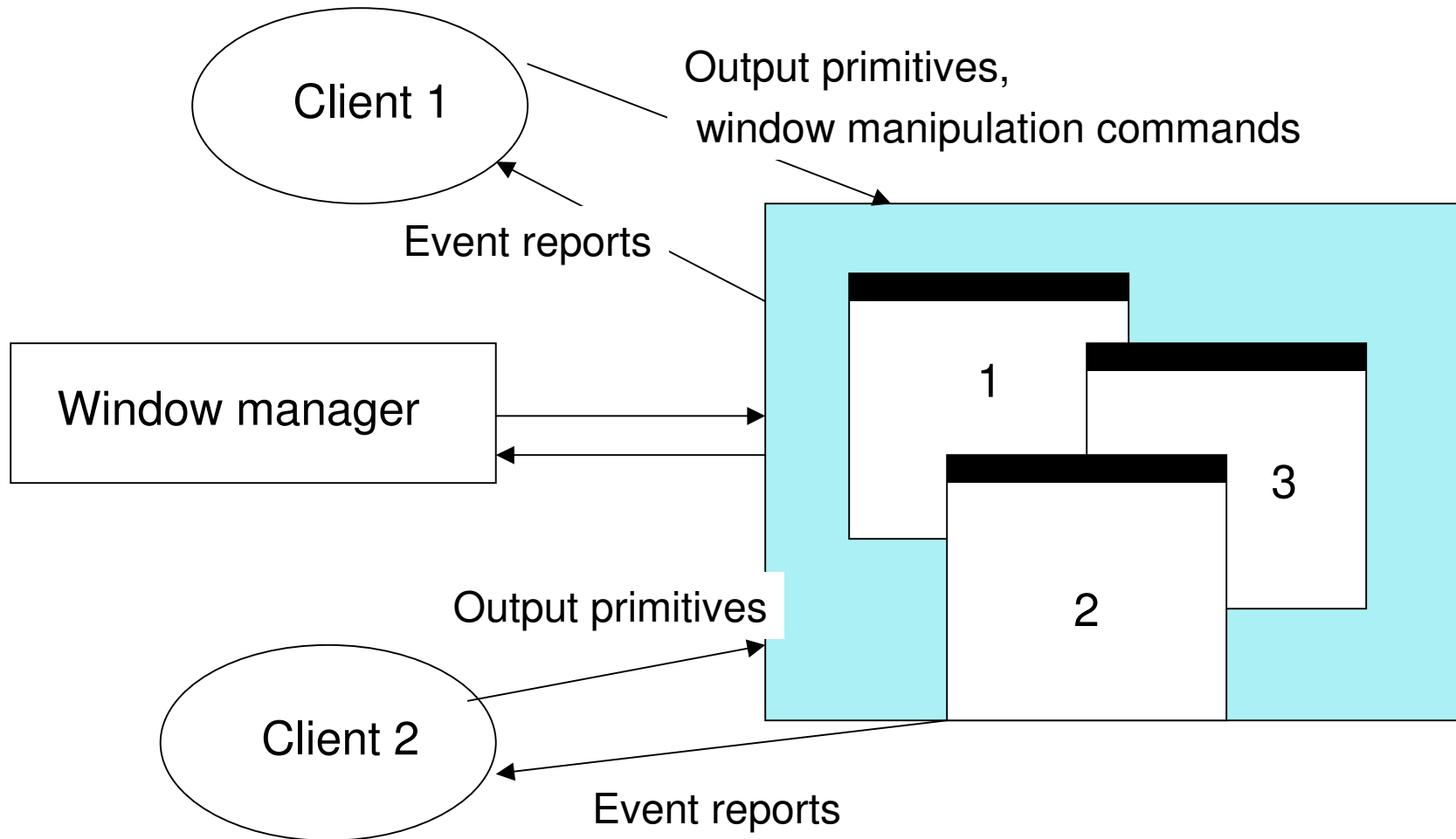
- Window management system
  - Window manager:
    - interacts with applications requesting operations on windows
    - Allocates screen space for applications (windows)
    - Routes events to appropriate applications
  - Window system:
    - Low level functions which manipulate windows (Xwindows)
- Window management system is a **server**
- Applications which use system are **clients**

# Communications for Client/Server

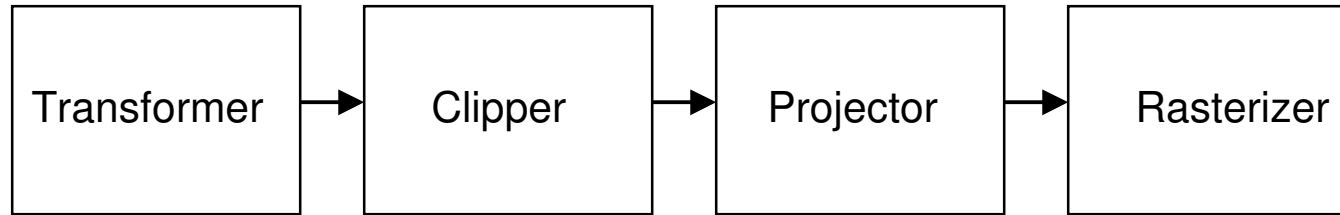


- Interprocess communication protocol
- Need to minimize communication delays
  - Asynchronous messages (events)
  - Design single message to replace multiple messages
  - Move more functionality into server

# Events for Multiple Clients



# Display processor Pipeline



- Transformer: modify object such that viewed from correct angle
- Clipper: clips primitives to screen window
- Projector: project 3-D scene to 2-D image
- Rasterizer: scan convert, anti-aliasing operations