Introduction to Computer Graphics

Farhana Bandukwala, PhD
Lecture 6: User Interfaces
Outline

• Examples
• Goals
• Styles
• Design considerations
• Window management (revisited)
• User Interface management
Tiled Display
Multiple Views & Users
Goals

- Easy to learn and use
  - Shallow learning curve
- Low error rate
  - Minimize error possibilities
- Intuitive, providing easy recall
  - Familiarity
- Esthetically pleasing
Styles used commonly in Graphics

- **WYSIWYG**
  - Fundamental to interactivity
  - Spatial, color resolution of display/printer not same
- **Direct manipulation**
  - Visual objects, attributes
  - can be invoked by these representations
  - Can be slow for experienced users (vs command line, for eg.)
- **Iconic**
  - Pictorial representation of action, object, property
  - Note: does not always imply above (textual buttons)
  - Language independent, fast recall
  - But very dependent on user’s personal choice
- **Novel styles (for eg. Gestures) subject of much HCI research!**
Design considerations

• Consistency
  – Color code operations (red=stop, green=go)
  – Status messages shown at fixed place
  – Use “muscle memory”

• Feedback
  – Hardware level: for eg. key presses are echoed
  – Sequencing level: for eg. menus highlighted
  – Functional level: for eq. “work in progress” messages, hour glass

• Error minimization
  – Gray out invalid menu items
  – Context sensitive options
Design considerations (contd.)

- **Error recovery**
  - Provide undo, abort
  - Multilevel recovery

- **Accommodate multiple skill levels**
  - Function keys (accelerators)
  - Help
  - Prompts

- **Minimize memorization**
  - Explicit picking
  - Icons
State-based UI

- State variables determine behavior
- State changes occur:
  - User action (menu, mouse action)
  - Data (e.g. error states)
- State or Context specific UI options
  - (e.g. Paste available only after Copy)
Example: Picking - Software

1. Register Mouse callback (glutMouseFunc(mouseCB))
2. In callback:
   1. Check type of mouse button
   2. Check state (down, up)
   3. Find an intersecting object, if any
   4. Set state variable picked_object=<intersecting object>
3. Edit/Update/Delete applied to picked_object

```c
enum picked_object
// Menu callback function
void mouseCB(int type, int state, int x, int y)
{
    if (type == GLUT_LEFT_BUTTON)
    {
        if (state == GLUT_UP)
        {
            loop through all objects
            picked_object = min_object_dist(x, y);
        }
    }
}
int main(int argc, char ** argv)
{
    ...
    // register callback
    glutMouseFunc(mouseCB);

    // Process events
    glutMainLoop();
}
Example: Picking - Hardware

1. Register Mouse callback (glutMouseFunc(mouseCB))
2. In callback:
   1. Check type of mouse button
   2. Check state (down, up)
   3. Render objects in back buffer
   4. Draw objects in different colors, background in different color
   5. Read back picked pixel
   6. Color of pixel determines picked_object
3. Edit/Update/Delete applied to picked_object

```c
enum picked_object;
Bool pick_state=FALSE;
// Menu callback function
Void renderScene(){
  if (pickState)
    drawEncodedObjects();
  else{
    drawObjects();
    glutSwapBuffers();
  }
}

void mouseCB(int type, int state, int x, int y) {
  if (type == GLUT_LEFT_BUTTON) {
    if (state == GLUT_DOWN) {
      pick_state=TRUE;
      renderScene();
      glReadBuffer(GL_BACK);
      glReadPixel(x,y,1,1,GL_RGB,GL_UNSIGNED_BYTE,&pixel);
      picked_object = decodeRGB(&pixel);
      pick_state=FALSE;
    }    
  }    
}

int main(int argc, char ** argv) {
  glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
  // register callback
  glutMouseFunc(mouseCB);
  // Process events
  glutMainLoop();
  return 0;
}
```
Window management (revisited)

• Output handling
  – Provides screen space as output resource to clients
  – Strategies for screen allocation
    • not responsible for newly exposed window regions,
    • clients get “display”, ”reshape” event (MS Windows, X-Windows)
Window management (contd.)

- Input handling
  - Input resource is devices generating events
  - Routes events to proper client (demultiplexing)
  - Use window hierarchy
- Strategies
  - Real-estate based: uses cursor position
  - Listener based: “click to type” -> User indicates which window is active
User Interface Management Systems

- Assists in implementing user interface (GLUT, Tcl-TK, MFC)
- Provide screen design, help & error messages, undo, manage data associated with application
- Specific functionality
  - Dialogue sequencing
    - Using state diagram, variables
  - Visual design: “look & feel”