Better Interactive Programs

Ed Angel
Professor of Computer Science, Electrical and Computer Engineering, and Media Arts
University of New Mexico
Objectives

• Learn to build more sophisticated interactive programs using
  - Picking
    • Select objects from the display
    • Three methods
  - Rubberbanding
    • Interactive drawing of lines and rectangles
  - Display Lists
    • Retained mode graphics
Picking

• Identify a user-defined object on the display
• In principle, it should be simple because the mouse gives the position and we should be able to determine to which object(s) a position corresponds

• Practical difficulties
  - Pipeline architecture is feed forward, hard to go from screen back to world
  - Complicated by screen being 2D, world is 3D
  - How close do we have to come to object to say we selected it?
Three Approaches

• Hit list
  - Most general approach but most difficult to implement

• Use back or some other buffer to store object ids as the objects are rendered

• Rectangular maps
  - Easy to implement for many applications
  - See paint program in text
Rendering Modes

- OpenGL can render in one of three modes selected by `glRenderMode(mode)`
  - `GL_RENDER`: normal rendering to the frame buffer (default)
  - `GL_FEEDBACK`: provides list of primitives rendered but no output to the frame buffer
  - `GL_SELECTION`: Each primitive in the view volume generates a hit record that is placed in a name stack which can be examined later
Selection Mode Functions

- `glSelectBuffer()`: specifies name buffer
- `glInitNames()`: initializes name buffer
- `glPushName(id)`: push id on name buffer
- `glPopName()`: pop top of name buffer
- `glLoadName(id)`: replace top name on buffer

- `id` is set by application program to identify objects
Using Selection Mode

• Initialize name buffer
• Enter selection mode (using mouse)
• Render scene with user-defined identifiers
• Reenter normal render mode
  - This operation returns number of hits
• Examine contents of name buffer (hit records)
  - Hit records include id and depth information
Selection Mode and Picking

• As we just described it, selection mode won’t work for picking because every primitive in the view volume will generate a hit.

• Change the viewing parameters so that only those primitives near the cursor are in the altered view volume.

  - Use `gluPickMatrix` (see text for details)
Using Regions of the Screen

• Many applications use a simple rectangular arrangement of the screen
  - Example: paint/CAD program

    +---+       +---+
    | tool |       | menus |
    +-----+       +------+
          |       |
          |       |
          |       |
          |       |
    +-----+       +------+
    | s     |       |      |
    +-------+       +-----+
          |drawing area |
          +-----------+

• Easier to look at mouse position and determine which area of screen it is in than using selection mode picking
Using another buffer and colors for picking

- For a small number of objects, we can assign a unique color (often in color index mode) to each object
- We then render the scene to a color buffer other than the front buffer so the results of the rendering are not visible
- We then get the mouse position and use `glReadPixels()` to read the color in the buffer we just wrote at the position of the mouse
- The returned color gives the id of the object
Writing Modes

- Application
- Source
- Bitwise logical operation
- Destination
- Frame buffer
- Read_pixel
- Write_pixel
XOR write

- Usual (default) mode: source replaces destination (d’ = s)
  - Cannot write temporary lines this way because we cannot recover what was “under” the line in a fast simple way

- Exclusive OR mode (XOR) (d’ = d ⊕ s)
  - x ⊕ y ⊕ x = y
  - Hence, if we use XOR mode to write a line, we can draw it a second time and line is erased!
Rubberbanding

- Switch to XOR write mode
- Draw object
  - For line can use first mouse click to fix one endpoint and then use motion callback to continuously update the second endpoint
  - Each time mouse is moved, redraw line which erases it and then draw line from fixed first position to to new second position
  - At end, switch back to normal drawing mode and draw line
- Works for other objects: rectangles, circles
Rubberband Lines

- Initial display
- Draw line with mouse in XOR mode
- Mouse moved to new position
- Original line redrawn with XOR
- New line drawn with XOR
XOR in OpenGL

- There are 16 possible logical operations between two bits
- All are supported by OpenGL
  - Must first enable logical operations
    - `glEnable(GL_COLOR_LOGIC_OP)`
  - Choose logical operation
    - `glLogicOp(GL_XOR)`
    - `glLogicOp(GL_COPY)` (default)
Immediate and Retained Modes

• Recall that in a standard OpenGL program, once an object is rendered there is no memory of it and to redisplay it, we must re-execute the code for it
  - Known as immediate mode graphics
  - Can be especially slow if the objects are complex and must be sent over a network

• Alternative is define objects and keep them in some form that can be redisplayed easily
  - Retained mode graphics
  - Accomplished in OpenGL via display lists
Display Lists

• Conceptually similar to a graphics file
  - Must define (name, create)
  - Add contents
  - Close

• In client-server environment, display list is placed on server
  - Can be redisplayed without sending primitives over network each time
Display List Functions

- Creating a display list
  
  ```c
  GLuint id;

  void init()
  {
    id = glGenLists( 1 );
    glNewList( id, GL_COMPILE );
    /* other OpenGL routines */
    glEndList();
  }

  Call a created list
  
  void display()
  {
    glCallList( id );
  }
  ```
Display Lists and State

- Most OpenGL functions can be put in display lists
- State changes made inside a display list persist after the display list is executed
- Can avoid unexpected results by using `glPushAttrib` and `glPushMatrix` upon entering a display list and `glPopAttrib` and `glPopMatrix` before exiting
Hierarchy and Display Lists

- Consider model of a car
  - Create display list for chassis
  - Create display list for wheel

```cpp
glNewList( CAR, GL_COMPILE );
glCallList( CHASSIS );
glTranslatef( ... );
glCallList( WHEEL );
glTranslatef( ... );
glCallList( WHEEL );
...  
glEndList();
```