Selection, Pick, and Feedback
OpenGL Modes

- **Render**
  - the “normal” operation, commands to color buffer

- **Selection and Pick**
  - return primitives intersecting view volume (selection) or close to the cursor (pick)

- **Feedback**
  - collect drawing commands for other use (e.g., control a vector plotter)
Selection

- Prepare an array to store returned primitive IDs
- Enter the selection mode
- Initialize name stack
- Define a view volume
- Draw primitives and record them in the name stack
- Exit selection and process returned data
Selection (cont.)

- `void glSelectBuffer (GLsizei, GLuint *)`
  - before entering selection mode
- `GLint glRenderMode (GLenum)`
  - `GL_RENDER, GL_SELECT, GL_FEEDBACK`
  - return the number of hits, or -1 for overflow
- `void glInitNames(void), glPushMatrix(GLuint), glPopMatrix(void), glLoadName(GLuint)`
  - at least 64 spaces
glSelectBuffer(buffersize, buffer);

glRenderMode (GL_SELECT);
   glInitNames();
   glPushMatrix(-1);
   glLoadIdentity(1);
   draw(
   glLoadIdentity(2);
   draw(
   glLoadIdentity(3);
   draw(
   
   hits = glRenderMode(GL_RENDER);
Hit Records

- Number of names on the stack when the hit occurred (useful for hierarchical model)
- Min and max z coordinates of primitives between hits
- Contents of the name stack (again, useful for hierarchical model, later)
void processHits (GLint hits, GLuint buffer[]) {
    unsigned int i, j;
    GLuint names, *ptr;

    printf("hits = %d\n", hits);
    ptr = (GLuint *) buffer;
    for (i = 0; i < hits; i++) { /* for each hit */
        names = *ptr;
        printf(" number of names for hit = %d\n", names); ptr++;
        printf(" z1 is %g", (float) *ptr/0x7fffffff); ptr++;
        printf(" z2 is %g\n", (float) *ptr/0x7fffffff); ptr++;
        printf(" the name is ");
        for (j = 0; j < names; j++) { /* for each name */
            printf("%d ", *ptr); ptr++;
        }
        printf("\n");
    }
    printf("\n");
}
Example - select.c

hits = 2
  number of names for hit = 1
  z1 is 2147483648; z2 is 2147483648
  the name is 1
  number of names for hit = 1
  z1 is 0; z2 is 0
  the name is 3

Computer Graphics
How this gets done?

- The user selects a view volume which can be different from that used in display.
- Remember that GL_MODELVIEW and GL_PROJECTION will be applied.
- If we can transform everything, i.e.,
  - The selection view volume
  - graphic primitives
    - into a “canonical” volume
- Then selection is equivalent to clipping!
Recall that the normalization transform accomplishes the mapping of arbitrary volumes into a canonical volume.
Clipping with 6-bit outcode

- Perspective
- Above $y > -z$
- Below $y < z$
- Right $x > -z$
- Left $x < z$
- Behind $z < -1$
- In front $z > z_{\text{min}}$

- Parallel
- Above $y > 1$
- Below $y < -1$
- Right $x > 1$
- Left $x < -1$
- Behind $z < -1$
- In front $z > 0$
An Even Better Solution

A single volume for both parallel and perspective

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1/Z_{\text{min}} & -Z_{\text{min}} \\
0 & 0 & 0 & 1/Z_{\text{min}}
\end{bmatrix}
\]
Picking

- Similar to Selection, but
  - usually triggered by a mouse event
  - need to set up a special gluPickMatrix (usually specify a region around mouse pointer)
- void gluPickMatrix(x, y, width, height, viewpoint[4])
- `glGetIntegerv(GL_VIEWPORT, viewport);`
- `glSelectBuffer(Bufsize, Buffer);`
- `glRenderMode(GL_SELECT);`
  - `glInitNames();`
  - `glPushName(-1);`
  - `glMatrixMode(GL_PROJECTION);`
  - `glPushMatrix();`
    - `glLoadIdentity();`
    - `gluPickMatrix(x, viewport[3]-y, 5.0, 5.0, viewport);`
  - `glOrtho, gluPerspective, gluFustrum, draw();`
  - `glPopMatrix();`
- `hits = glRenderMode(GL_RENDER);`
Example: picksquare.c

hits = 1
number of names for this hit = 2
z1 is 2147483648; z2 is 2147483648
names are 1 2
Hierarchical Picking

- Identify components in a complicated model using push and pop
  - `glPushMatrix();`
  - `translate_to_1st_car();`
  - `glPushName(1);`
    - `draw_body();`
  - `glPopName();`
  - `glPopMatrix();`
  - `glPushMatrix();`
  - `translate_to_2nd_car();`
  - `glPushName(2);`
    - `draw_body();`
  - `glPopName();`
  - `glPopMatrix();`
How pick gets done?

- Similar to selection, with an extra `gluPickMatrix` specified
- One more matrix is concatenated into the normalization transform
- the pick region becomes a unit cube
\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & -z_{\text{min}} \\
0 & 0 & 1 + z_{\text{min}} & 0
\end{bmatrix}, \quad z_{\text{min}} \neq -1
\]
Draw_body()
  - glPushMatrix();
    - glPushName(1);
      - draw_wheel();
    - glPopName();
  - glPopMatrix();
  - glPushMatrix();
    - glPushName(2);
      - draw_wheel();
    - glPopName();
  - glPopMatrix();

Draw_wheel()
  - glPushMatrix();
    - glPushName(1);
      - draw_bolts();
    - glPopName();
  - glPopMatrix();
  - glPushMatrix();
    - glPushName(2);
      - draw_bolts();
    - glPopName();
  - glPopMatrix();
Empty

Pick is outside all cars

2 d1 d2 2 1 Car 2, wheel 1

1 d1 d2 3 Car 3 body

3 d1 d2 1 1 0 Bolt 0 on wheel 1 on car 1
Display List

- A static list of
  - OpenGL (or Glu) commands
  - Cached at server site for efficiency
  - Best for
    - matrix operations
    - raster maps and images
    - lights, material properties
    - textures, etc.
Display List (cont.)

- Gluint glGenLists (Glsizei range);
  - range number of unused list indices
- void glNewList(Gluint list, Glenum mode);
  - mode: GL_COMPLETE or GL_COMPLETE_AND_EXECUTE
- void glEndList(void);
- void glCallList( Gluint list);
  - call the list back
Other Useful Functions

- Too many to mention
- Just have to experiment with them