MENUS AND PICKING IN GL

3.7 Menus

- GLUT supports pop-up menus
  - A menu can have submenus
- Three steps
  - Define entries for the menu
  - Define action for each menu item
    - Action carried out if entry selected
  - – Attach menu to a mouse button

Defining a simple menu

```
menu_id = glutCreateMenu(mymenu);
glutAddmenuEntry(“clear Screen”, 1);
gluAddMenuEntry(“exit”, 2);
glutAttachMenu(GLUT_RIGHT_BUTTON);
Menu callback

void mymenu(int id)
{
        if(id == 1) glClear();
        if(id == 2) exit(0);
}
```

– Note each menu has an id that is returned when it is created

Add submenus by

glutAddSubMenu(char *submenu_name, submenu id)

3.8 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?

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(GLsizei n, GLuint *buff): specifies name buffer
- glInitNames(): initializes name buffer
- glPushName(GLuint name): push id on name buffer
- glPopName(): pop top of name buffer
- glLoadName(GLuint name): 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)

```
void mouse (int button, int state, int x, int y)
{
        GLUint nameBuffer[SIZE];
        GLint hits;
        GLint viewport[4];
        if (button == GLUT_LEFT_BUTTON && state== GLUT_DOWN)
        {
                /* initialize the name stack */
                glInitNames();
                glPushName(0);
                glSelectBuffer(SIZE, nameBuffer)l

                /* set up viewing for selection mode */

                glGetIntegerv(GL_VIEWPORT, viewport); //gets the current viewport
                glMatrixMode(GL_PROJECTION);

                /* save original viewing matrix */
                glPushMatrix();
                glLoadIdentity();

                /* N X N pick area around cursor */
                gluPickMatrix( (GLdouble) x,(GLdouble)(viewport[3]-y),N,N,viewport);

                /* same clipping window as in reshape callback */
                gluOrtho2D(xmin,xmax,ymin,ymax);

                draw_objects(GL_SELECT);
                glMatrixMode(GL_PROJECTION);

                /* restore viewing matrix */
                glPopMatrix();
                glFlush();

                /* return back to normal render mode */

                hits = glRenderMode(GL_RENDER);
                /* process hits from selection mode rendering*/

                processHits(hits, nameBuff);

                /* normal render */
                glutPostRedisplay();
        }
}

void draw_objects(GLenum mode)
{
        if (mode == GL_SELECT)
                glLoadName(1);
        glColor3f(1.0,0.0,0.0)
        glRectf(-0.5,-0.5,1.0,1.0);
        if (mode == GL_SELECT)
                glLoadName(2);
        glColor3f(0.0,0.0,1.0)
        glRectf(-1.0,-1.0,0.5,0.5);
}

void processHits(GLint hits, GLUint buffer[])
{
        unsigned int i,j;

}
```

Source : http://elearningatria.files.wordpress.com/2013/10/cse-vi-computer-graphics-and-visualization-10cs65-notes.pdf