

# Display Lists, Menus, and Picking

Tom Kelliher, CS 320

Feb. 23, 2011

## 1 Administrivia

### Announcements

### Assignment

Read `polygon.c`, Section 3.9.

### From Last Time

Input devices and interaction introduction.

### Outline

1. Display lists.
2. Menus.
3. Pick selection.

### Coming Up

`polygon.c`

## 2 Display Lists, Menus, and Picking

Display lists and distributed computing.

### 2.1 Text Display

Idea:

1. Specify starting location of text (world coordinates).
2. Start writing, specifying font and character.

Example:

```
void renderString(GLdouble x, GLdouble y, void *font, char *text)
{
    glRasterPos2d(x, y);

    while (text)
    {
        glutBitmapCharacter(font, *text);
        ++text;
    }
}
```

See man page for `glutBitmapCharacter` for list of available bitmap fonts. Example:

```
renderString(0.0, 0.0, GLUT_BITMAP_9_BY_15, "OpenGL rocks!");
```

Idea similar to font cache: display lists. Program example:

```

base = glGenLists(128);

for(i=0;i<128;i++)
{
    glNewList(base+i, GL_COMPILE);
    glutBitmapCharacter(GLUT_BITMAP_9_BY_15, i);
    glEndList();
}

glListBase(base);

```

Use:

```

// Dump time string into out.

glRasterPos2i(ww-80, wh-15); // Window units = world units.
glColor3f(0.0,0.0,0.0);
glBegin(GL_QUADS);          // Erase current time.
    glVertex2i(ww-80, wh-15);
    glVertex2i(ww, wh-15);
    glVertex2i(ww, wh);
    glVertex2i(ww-80, wh);
glEnd();
glColor3f(1.0,1.0,1.0);
glCallLists(strlen(out), GL_BYTE, out);

```

Another example:

```

list = glGenLists(1)

glNewList(list, GL_COMPILE);
    glBegin(GL_POLYGON);
        glVertex2f(...);
        ...
    glEnd();
glEndList();

...

glCallList(list);

```

## 2.2 Menus and Sub-Menus

Why do we use menus? — What's the alternative?

```
int main(int argc, char** argv)
{
    int c_menu;

    /* ... */

    c_menu = glutCreateMenu(color_menu);
    glutAddMenuEntry("Black",0);
    glutAddMenuEntry("Red",1);
    glutAddMenuEntry("Green",2);
    glutAddMenuEntry("Blue",3);
    glutAddMenuEntry("Cyan",4);
    glutAddMenuEntry("Magenta",5);
    glutAddMenuEntry("Yellow",6);
    glutAddMenuEntry("White",7);

    glutCreateMenu(main_menu);
    glutAddMenuEntry("new polygon", 1);
    glutAddMenuEntry("end polygon", 2);
    glutAddMenuEntry("delete polygon", 3);
    glutAddMenuEntry("move polygon", 4);
    glutAddMenuEntry("quit",5);
    glutAddSubMenu("Colors", c_menu);
    glutAttachMenu(GLUT_MIDDLE_BUTTON);

    /* ... */
}
```

Polygon representation:

```

typedef struct polygon
{
    int color; /* color index */
    bool used; /* TRUE if polygon exists */
    int xmin, xmax, ymin, ymax; /* bounding box */
    float xc, yc; /* center of polygon */
    int nvertices; /* number of vertices */
    int x[MAX_VERTICES]; /* vertices */
    int y[MAX_VERTICES];
} polygon;

```

Setting a polygon's color:

```

GLfloat colors[8][3]={0.0, 0.0, 0.0}, {1.0, 0.0, 0.0}, {0.0, 1.0, 0.0},
                    {0.0, 0.0, 1.0}, {0.0, 1.0, 1.0}, {1.0, 0.0, 1.0},
                    {1.0, 1.0, 0.0}, {1.0, 1.0, 1.0}};

```

```

void color_menu(int index)
{
    present_color = index;
    if(in_polygon>=0) polygons[in_polygon].color = index;
}

```

Rendering polygons:

```

void myDisplay()
{
    /* display all active polygons */

    int i, j;

    glClear(GL_COLOR_BUFFER_BIT);
    for(i=0; i<MAX_POLYGONS; i++)
    {
        if(polygons[i].used)
        {
            glColor3fv(colors[polygons[i].color]);
            glBegin(GL_POLYGON);
            for(j=0; j<polygons[i].nvertices; j++)
                glVertex2i(polygons[i].x[j], polygons[i].y[j]);
            glEnd();
        }
    }
    glFlush();
}

```

## 2.3 Coordinate systems

Recall:

1. Rendering origin for OpenGL.
2. Window and mouse coordinate origin for the window system.
3. Translating.

## 2.4 Pick Determination and Drawing States

What's the problem here?

1. Mouse function called into play here.

2. Note that middle button taken out of action.

```
int pick_polygon(int x, int y)
{
    /* find first polygon in which we are in bounding box */

    int i;

    for(i=0; i<MAX_POLYGONS; i++)
    {
        if(polygons[i].used)
            if((x>=polygons[i].xmin) && (x<=polygons[i].xmax) &&
                (y>=polygons[i].ymin)&&(y<polygons[i].ymax))
            {
                in_polygon = i;
                moving = TRUE;
                return(i);
            }
        printf("not in a polygon\n");
        return(-1);
    }
}

void main_menu(int index)
{
    int i;
    switch(index)
    {
    case(1): /* create a new polygon */
        {
            moving = FALSE;
            for(i=0; i<MAX_POLYGONS; i++) if(polygons[i].used == FALSE) break;
            if(i == MAX_POLYGONS)
            {
                printf("exceeded maximum number of polygons\n");
                exit(0);
            }
            polygons[i].color = present_color;
            polygons[i].used = TRUE;
            polygons[i].nvertices = 0;
            in_polygon = i;
        }
    }
}
```

```

    picking = FALSE;
    break;
}
case(2): /* end polygon and find bounding box and center */
{
    moving = FALSE;
    if(in_polygon>=0)
    {
        polygons[in_polygon].xmax =
            polygons[in_polygon].xmin =
            polygons[in_polygon].x[0];

        polygons[in_polygon].ymax =
            polygons[in_polygon].ymin =
            polygons[in_polygon].y[0];

        polygons[in_polygon].xc = polygons[in_polygon].x[0];
        polygons[in_polygon].yc = polygons[in_polygon].y[0];

        for(i=1;i<polygons[in_polygon].nvertices;i++)
        {
            if(polygons[in_polygon].x[i]<polygons[in_polygon].xmin)
                polygons[in_polygon].xmin =
                    polygons[in_polygon].x[i];
            else if(polygons[in_polygon].x[i] >
                polygons[in_polygon].xmax)
                polygons[in_polygon].xmax =
                    polygons[in_polygon].x[i];

            if(polygons[in_polygon].y[i]<polygons[in_polygon].ymin)
                polygons[in_polygon].ymin =
                    polygons[in_polygon].y[i];
            else if(polygons[in_polygon].y[i] >
                polygons[in_polygon].ymax)
                polygons[in_polygon].ymax =
                    polygons[in_polygon].y[i];

            polygons[in_polygon].xc += polygons[in_polygon].x[i];
            polygons[in_polygon].yc += polygons[in_polygon].y[i];
        }

        polygons[in_polygon].xc =
            polygons[in_polygon].xc/polygons[in_polygon].nvertices;

        polygons[in_polygon].yc =

```



```
        polygons[in_polygon].yc/polygons[in_polygon].nvertices;
    }
    in_polygon = -1;
    glutPostRedisplay();
    break;
}
case(3): /* set picking mode */
{
    picking = TRUE;
    moving = FALSE;
    break;
}
case(4): /* set moving mode */
{
    moving = TRUE;
    break;
}
case(5): /* exit */
{
    exit(0);
    break;
}
}
}
```