

```

/* IT Text CG, Chapter 8, Exercise 3, B Spline Curve */
#include <GL/glut.h>

float BN(int j, int m, float t);

/* Bスプライン曲線の制御点 */
GLfloat controlpoints[4][3] = {{-40.0, -40.0, 0.0},
                                {-20.0, 40.0, 0.0},
                                {20.0, 40.0, 0.0},
                                {40.0, -40.0, 0.0}};

/* Bスプライン曲線のノットベクトル */
/*  $T = [t_0 \ t_1 \ \dots \ t_{n+M}] = [-(M-1) \ -(M-2) \ \dots \ n+1]$  */
float knotvec[8] = {-3.0, -2.0, -1.0, 0.0, 1.0, 2.0, 3.0, 4.0};

/* 両端で制御点と一致するBスプライン曲線の
ノットベクトル(多重ノット) */
/* float knotvec[8] =
{0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0}; */

void display(void)
{
    int i, j;
    float t1, tpitch1;
    GLfloat px1, py1, pz1;

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glFrustum(-50.0, 50.0, -50.0, 50.0, 50.0, 1000.0);
    gluLookAt(100.0, 100.0, 100.0,

```

```

        0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();

/* Bスプライン曲線の表示 */
glColor3f(1.0, 0.0, 1.0);
glBegin(GL_LINE_STRIP);
t1 = 0.0;
tpitch1 = 1.0/60.0;
for(i = 1; i <= 60; ++i)
{
    px1 = 0.0;
    py1 = 0.0;
    pz1 = 0.0;
    for(j= 0; j <= 3; ++j)
    {
        px1 = px1 +
            (GLfloat)(BN(j, 4, t1) * controlpoints[j][0]);
        py1 = py1 +
            (GLfloat)(BN(j, 4, t1) * controlpoints[j][1]);
        pz1 = pz1 +
            (GLfloat)(BN(j, 4, t1) * controlpoints[j][2]);

    } /* for j */
    glVertex3f(px1, py1, pz1);

    t1 = t1 + tpitch1;
} /* for i */
glEnd();

```

```

/* Bスプライン曲線の制御点の表示 */
glColor3f(1.0, 0.0, 0.0);
glBegin(GL_LINE_STRIP);
for(i = 0; i <= 3; ++i)
{
    glVertex3f((GLfloat)controlpoints[i][0],
                (GLfloat)controlpoints[i][1],
                (GLfloat)controlpoints[i][2]);

} /* for i */
glEnd();

glFlush();

}

```

```

/* Bスプライン関数 $N_{j,M}(t)$ の計算をする関数 */
float BN(int j, int m, float t)
{
    float s1, r1, r2, r3, r4, r12, r34;

    if(m == 1)
    {
        if(t >= knotvec[j] && t < knotvec[j+1])
        {
            s1 = 1.0;

        }
        else

```

```

{
    s1 = 0.0;

} /* if */

return s1;

} /* if */

if(m > 1)
{
    r1 = (t - knotvec[j]);
    r2 = (knotvec[j+m-1] - knotvec[j]);
    r3 = (knotvec[j+m] - t);
    r4 = (knotvec[j+m] - knotvec[j+1]);

    if(r1 == 0.0 && r2 == 0.0)
    {
        r12 = 0.0;

    }
    else
    {

        if(r2 == 0.0)
        {
            r2 = r2 + 0.0000001;

        } /* if */

        r12 = r1/r2;
    }
}

```

```

    } /* if */

    if(r3 == 0.0 && r4 == 0.0)
    {
        r34 = 0.0;

    }
    else
    {
        if(r4 == 0.0)
        {
            r4 = r4 + 0.0000001;

        } /* if */

        r34 = r3/r4;

    } /* if */

    return r12 * BN(j, m-1, t) + r34 * BN(j+1, m-1, t);

} /* if */

} /* end of BN */

void keyboard(unsigned char key, int x, int y)
{
    switch(key) {
        case 27: exit(0); break;
    }
}

```

```
    }  
  
}  
  
int main(int argc, char** argv)  
{  
    glutInitWindowSize(1000,1000);  
    glutInitWindowPosition(0,0);  
    glutInit(&argc, argv);  
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA | GLUT_DEPTH);  
    glutCreateWindow("BsplineCurve");  
  
    glClearColor(0.0, 0.0, 0.0, 0.0);  
  
    glutDisplayFunc(display);  
    glutKeyboardFunc(keyboard);  
    glutMainLoop();  
}
```