コンピューターグラフィックス基礎 第 5 回課題

情報メディア創生学類 3 年 202313625 藤川興昌

実行環境

- Ubuntu 22.04.3 LTS
- gcc version 11.4.0

課題 1

ソースコード

```
#include <cstdlib>
#include <cmath>
#include <vector>
#include <stdio.h>

// 2�����x�N�g����������Ɖ]N���X
class Vector2d {
public:
        double x, y;
        Vector2d() { x = y = 0; }
        Vector2d(double _x, double _y) { x = _x; y = _y; }
        void set(double _x, double _y) { x = _x; y = _y; }

        // ������1�ɐ��K������
        void normalize() {
                double len = length();
                x /= len; y /= len;
        }

        // ������d�
        double length() { return sqrt(x * x + y * y); }

        // s�{����
        void scale(const double s) { x *= s; y *= s; }

        // ���Z��`
        Vector2d operator+(Vector2d v) { return Vector2d(x + v.x, y + v.y);
}
```

```
        // ???Z??`
        Vector2d operator-(Vector2d v) { return Vector2d(x - v.x, y - v.y);
}

        // ???ς?`
        double operator*(Vector2d v) { return x * v.x + y* v.y; }

        // ??????Z??`
        Vector2d& operator=(const Vector2d& v){ x = v.x; y = v.y; return
(*this); }

        // ???Z?????`
        Vector2d& operator+=(const Vector2d& v) { x += v.x; y += v.y; return
(*this); }

        // ???Z?????`
        Vector2d& operator-=(const Vector2d& v) { x -= v.x; y -= v.y; return
(*this); }

        // ?l???o????
        void print() { printf("Vector2d(%f %f)\n", x, y); }
};

// ?}?C?i?X?????t?????x?N?g??????????恩?ゥ??邯?Ꮵ?` ??Fb=
(-a); ?恩?qL?q?ł???
Vector2d operator-( const Vector2d& v ) { return( Vector2d( -v.x, -v.y ) );
}

// ?x?N?g???Ǝ????ς????????恩?ゥ??邯?Ꮵ?` ??F c=5*a+2*b; c=b*3;
?恩?qL?q?ł???
Vector2d operator*( const double& k, const Vector2d& v ) { return( Vector2d(
k*v.x, k*v.y ) );}
Vector2d operator*( const Vector2d& v, const double& k ) { return( Vector2d(
v.x*k, v.y*k ) );}

// ?x?N?g??????????D??鑀??????????恩?ゥ??邯?Ꮵ?` ??F c=a/2.3;
?恩?qL?q?ł???
Vector2d operator/( const Vector2d& v, const double& k ) { return( Vector2d(
v.x/k, v.y/k ) );}

int main(int argc, char**argv) {
        // ====================================================
        // 2??????x?N?g???N???X Vector2d ?g??????
        // ====================================================

        // 2??????x?N?g??(1, 2)?쐬
        Vector2d v0(1, 2);

        // ?쐬????2??????x?N?g????x,y???W?l???m?F
        v0.print();

        // 2??????x?N?g??(2, 4)?쐬
        Vector2d v1(2, 4);

        // ?x?N?g??????Z
        Vector2d v2 = v0 + v1;
```

```
        // ???Z????????s?m?F
        v2.print();

        // ?x?N?g??????Z
        Vector2d v3 = v1 - v0;

        // ???Z????????s?m?F
        v3.print();

        // ?x?N?g?????????m?F
        printf("v3.length() = %lf\n", v3.length());

        // ?x?N?g???X?J???[?{
        v3 = 5.0 * v3;

        // ?X?J???[?{????????s?m?F
        v3.print();

        // ?x?N?g??????????K??
        v3.normalize();

        // ???K??????????s?m?F
        v3.print();

        // ?x?N?g?????????m?F
        printf("v3.length() = %lf\n", v3.length());


        // ==================================================
        // std::vector ??v?f????úX?□\?Ùz??Б??Ďg????
        // ==================================================

        std::vector<Vector2d> vec; // ?z??鏨 ?iVector2d
?^?I?u?W?F?N?g???i?[?ł???j
        vec.push_back(v0); // ?z??????? v0 ??lj?
        vec.push_back(v1); // ?z??????? v1 ??lj?
        vec.push_back(v2); // ?z??????? v2 ??lj?

        printf("vec.size() = %d\n", vec.size()); //
?z??b????Ă???v?f?????m?F

        // ?z??摩?v?f???擾
        Vector2d firstElement = vec[0];

        // ?擾?????v?f?l???o??
        printf("firstElement=(%lf, %lf)\n", firstElement.x, firstElement.y);

        // ?S?Ą̆v?f???o??
        for(unsigned int i = 0; i < vec.size(); i++) {
                printf("vec[%d]=(%lf, %lf)\n", i, vec[i].x, vec[i].y);
        }


        //
???ゃ?F?ì??ʁR?????g???Ŏw?????????e?v???O?????R?[?h??lj
????邱??
```

```
        // (1) vec ����?���
        vec.clear();

        // (2) ����2������x�N�g���� vec ��Qi�[����
        // (5.0, 2.0), (3.2, -2.3), (4.1, 9.2), (-2.0, 4.0), (0.0, -2.7)
        vec.push_back(Vector2d(5.0, 2.0));
        vec.push_back(Vector2d(3.2, -2.3));
        vec.push_back(Vector2d(4.1, 9.2));
        vec.push_back(Vector2d(-2.0, 4.0));
        vec.push_back(Vector2d(0.0, -2.7));

        // (3) ��L��2������x�N�g����S��ĉ��Z�������ʂ̃ Vector2d
vecSum ��Qi�[����
        Vector2d vecSum(0.0, 0.0);
        for(int i = 0; i < vec.size(); i++) vecSum += vec[i];

        // (4) vecSum ���e(x,y�l)��BAvecSum ��������o����
        vecSum.print();
        printf("vecSum.length() = %lf\n", vecSum.length());

        // Visual Studio ��ŃR���\��[������������ɂ̏Ã��悤?���邪��
        // system("pause");

        return 0;
}
```

実行結果

Vector2d(1.000000 2.000000) Vector2d(3.000000 6.000000) Vector2d(1.000000 2.000000) v3.length() = 2.236068 Vector2d(5.000000 10.000000) Vector2d(0.447214 0.894427) v3.length() = 1.000000 vec.size() = 3 firstElement=(1.000000, 2.000000) vec[0]=(1.000000, 2.000000) vec[1]=(2.000000, 4.000000) vec[2]=(3.000000, 6.000000) Vector2d(10.300000 10.200000) vecSum.length() = 14.495861

課題 2

ソースコード

```
#include <cstdlib>
#include <cmath>
#include <vector>
#include <GL/glut.h>
#include <stdio.h>
```

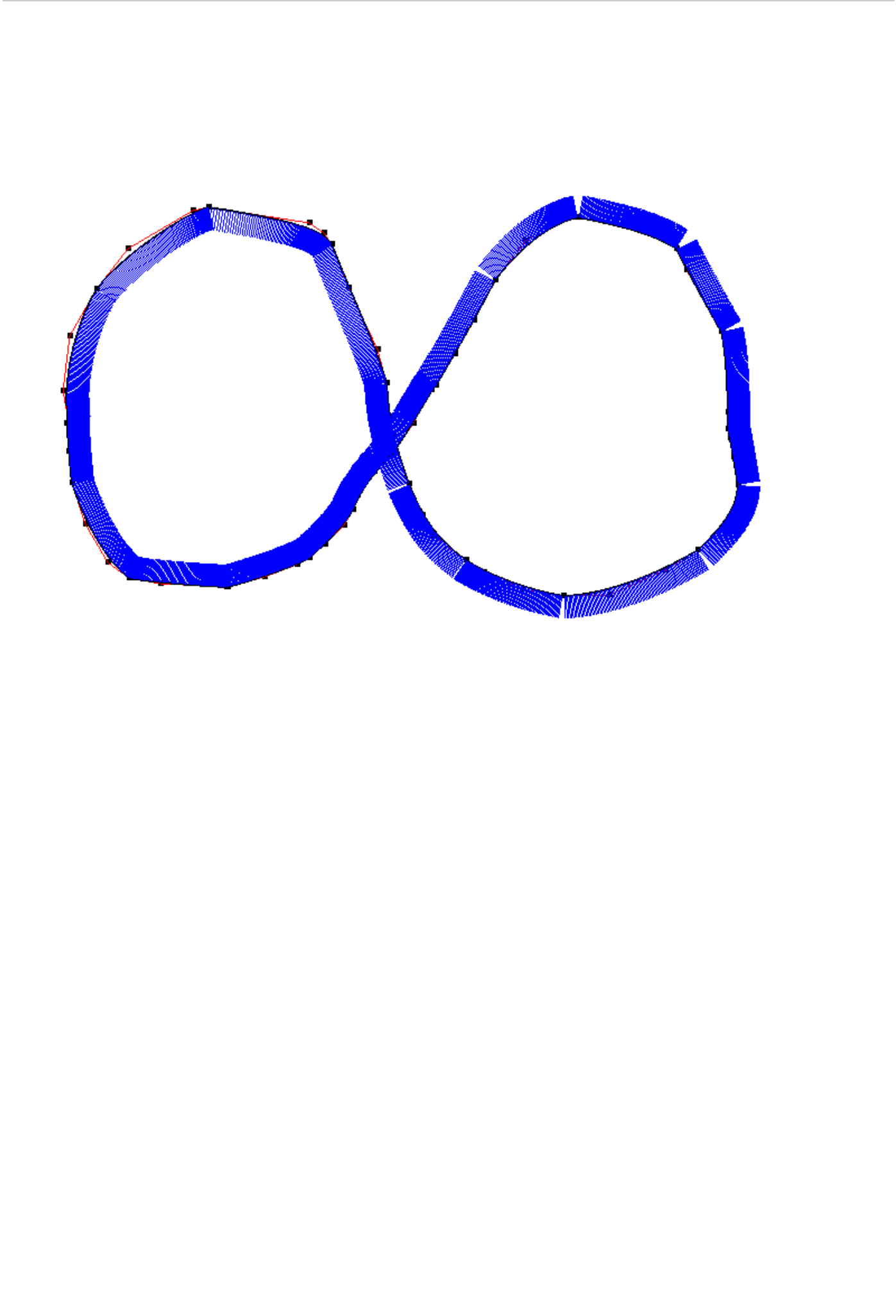
```
#include <utility>
#include <iostream>

// 2�����x�N�g�������������Ԗ
class Vector2d {
public:
        double x, y;
        Vector2d() { x = y = 0; }
        Vector2d(double _x, double _y) { x = _x; y = _y; }
        void set(double _x, double _y) { x = _x; y = _y; }

        // ������1�ɐ��K������
        void normalize() {
                double len = length();
                x /= len; y /= len;
        }

        // ������d�
        double length() { return sqrt(x * x + y * y); }

        // s�{����
        void scale(const double s) { x *= s; y *= s; }

        // ���Z��`
        Vector2d operator+(Vector2d v) { return Vector2d(x + v.x, y + v.y);
}

        // ���Z��`
        Vector2d operator-(Vector2d v) { return Vector2d(x - v.x, y - v.y);
}

        // ���ς̀
        double operator*(Vector2d v) { return x * v.x + y* v.y; }

        // ������Z��`
        Vector2d& operator=(const Vector2d& v){ x = v.x; y = v.y; return
(*this); }

        // ���Z�����`
        Vector2d& operator+=(const Vector2d& v) { x += v.x; y += v.y; return
(*this); }

        // ���Z�����`
        Vector2d& operator-=(const Vector2d& v) { x -= v.x; y -= v.y; return
(*this); }

        // ̀l���ò����
        void print() { printf("Vector2d(%f %f)\n", x, y); }
};

// ̀}̀C̀ìX̀́̀̀̀t̀̀̀̀̀x̀Ǹg̀̀̀̀̀̀̀̀̀̀̀̀恩̀ɂ̀̀邨̀Ԗ̀` ̀̀Fb=
(-a); ̀恩̀qL̀q̀ł̀̀̀
Vector2d operator-( const Vector2d& v ) { return( Vector2d( -v.x, -v.y ) );
}

// ̀x̀Ǹg̀̀̀Ǝ̀̀̀̀̀ς̀̀̀̀̀̀̀̀恩̀ɂ̀̀邨̀Ԗ̀` ̀̀F c=5*a+2*b; c=b*3;
̀恩̀qL̀q̀ł̀̀̀
```

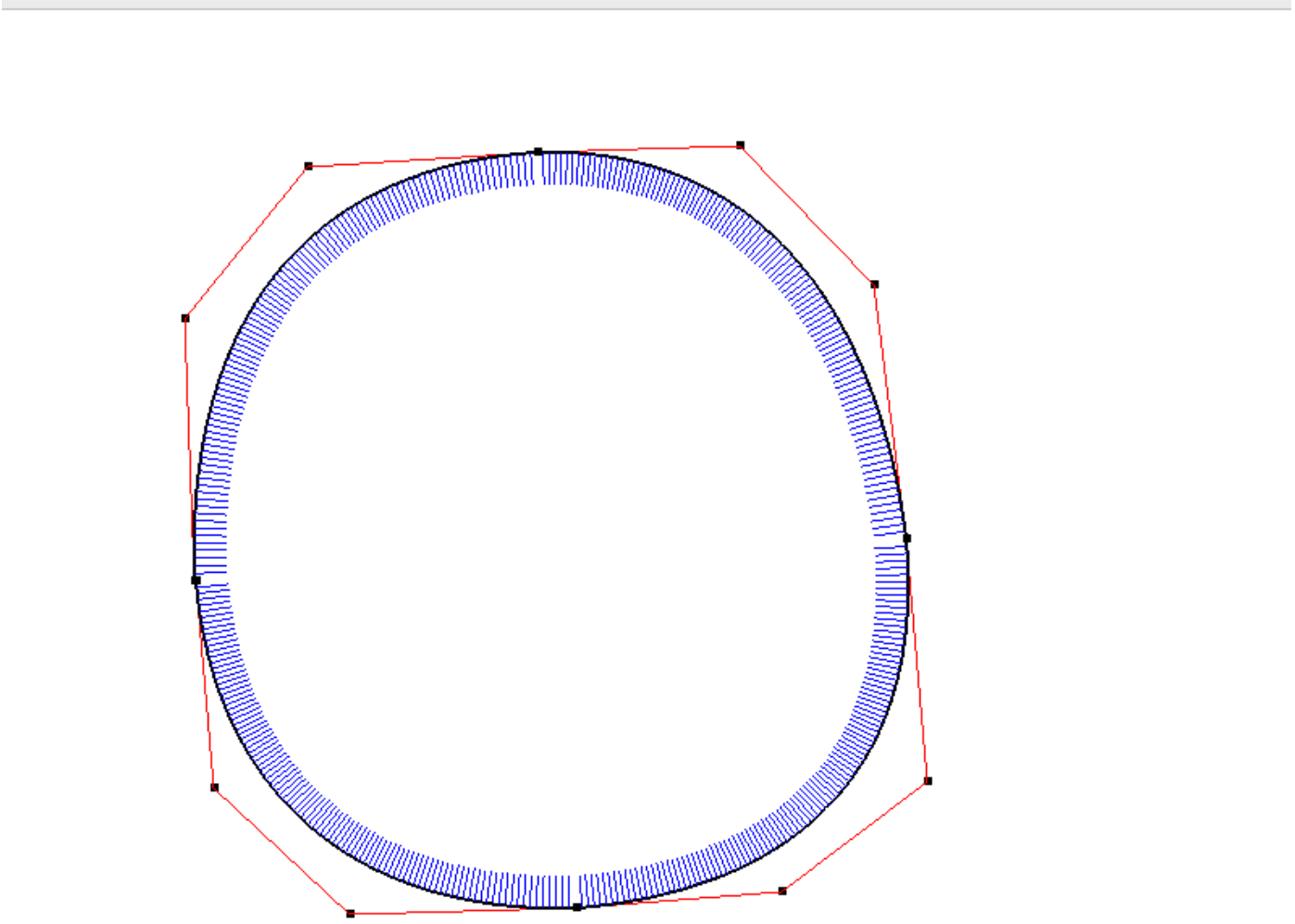
```
Vector2d operator*( const double& k, const Vector2d& v ) { return( Vector2d(
k*v.x, k*v.y ) );}
Vector2d operator*( const Vector2d& v, const double& k ) { return( Vector2d(
v.x*k, v.y*k ) );}

// ?x?N?g??????????D??鑀??????????恩?ゥ??邽?P?` ??F c=a/2.3;
?恩?qL?q?ł???
Vector2d operator/( const Vector2d& v, const double& k ) { return( Vector2d(
v.x/k, v.y/k ) );}

//
==========================================================================
====================


std::vector<Vector2d> g_ControlPoints; // ????_???i?[????
const double STEP = 0.01;

// ?\???????????????□??ŋL??
void display(void) {
        glClearColor (1.0, 1.0, 1.0, 1.0);  // ??????F?w??
        glClear (GL_COLOR_BUFFER_BIT );     // ??ｙ???

        // ????_?`??
        glPointSize(5);
        glColor3d(0.0, 0.0, 0.0);
        glBegin(GL_POINTS);
        for(unsigned int i = 0; i < g_ControlPoints.size(); i++) {
                glVertex2d(g_ControlPoints[i].x, g_ControlPoints[i].y);
        }
        glEnd();

        // ????_??????ε????`??
        glColor3d(1.0, 0.0, 0.0);
        glLineWidth(1);
        glBegin(GL_LINE_STRIP);
        for(unsigned int i = 0; i < g_ControlPoints.size(); i++) {
                glVertex2d(g_ControlPoints[i].x, g_ControlPoints[i].y);
        }
        glEnd();

        // ?? ??????Bx?W?F?R???`?悞??R?[?h??lj??????
        Vector2d p_1, p_2, p_3, p_4;
        std::vector<std::pair<Vector2d, Vector2d>> norm_lines;
        glColor3d(0.0, 0.0, 0.0);
        glLineWidth(1.5);
        glBegin(GL_LINE_STRIP);
        for(int i = 0; i < (int)g_ControlPoints.size() - 3; i+= 3) {
                glVertex2d(g_ControlPoints[i].x, g_ControlPoints[i].y);
                for(double t = STEP; t < 1.0; t += STEP) {
                        p_1 = g_ControlPoints[i];
                        p_1.scale(pow(1.0 - t, 3.0));
                        p_2 = g_ControlPoints[i+1];
                        p_2.scale(3.0 * t * pow(1.0 - t, 2.0));
                        p_3 = g_ControlPoints[i+2];
                        p_3.scale(3.0 * pow(t, 2.0) * (1.0 - t));
                        p_4 = g_ControlPoints[i+3];
```

```
                        p_4.scale(pow(t, 3.0));
                        Vector2d p_t = p_1 + p_2 + p_3 + p_4;
                        glVertex2d(p_t.x, p_t.y);
                        printf("p_t1: (%lf,%lf)\n", p_t.x, p_t.y);

                        double t2 = t * t;
            double t3 = t2 * t;
                        Vector2d tangent = (-3 + 6 * t - 3 * t2) *
g_ControlPoints[i]
                            + (3 - 12 * t + 9 * t2) * g_ControlPoints[i + 1]
                            + (6 * t - 9 * t2) * g_ControlPoints[i + 2]
                            + 3 * t2 * g_ControlPoints[i + 3];
                        tangent.normalize();
                        Vector2d normal(-tangent.y, tangent.x);
            Vector2d n1 = p_t;
            Vector2d n2 = p_t + normal * 10.0 * 2.0;
                        norm_lines.push_back(std::make_pair(n1, n2));
                }
                glVertex2d(g_ControlPoints[i+3].x, g_ControlPoints[i+3].y);
        }
        glEnd();

        glColor3d(0.0, 0.0, 1.0);
        glLineWidth(1.0);
        glBegin(GL_LINES);
        for(int i = 0; i < norm_lines.size(); i++) {
                glVertex2d(norm_lines[i].first.x, norm_lines[i].first.y);
                glVertex2d(norm_lines[i].second.x, norm_lines[i].second.y);
        }
        glEnd();

        glutSwapBuffers();
}

void resizeWindow(int w, int h) {
        h = (h == 0) ? 1 : h;
        glViewport(0, 0, w, h);

        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();

        // ?E?B???h?E?????W?n?_?
        // ?}?E?X?N???b?N???W?h`????W????v????恩?R????e
        glOrtho(0, w, h, 0, -10, 10);

        glMatrixMode(GL_MODELVIEW);
}

// ?L?[?{?[?h?C?x???g????
void keyboard(unsigned char key, int x, int y) {
        switch (key) {
        case 'q':
        case 'Q':
        case '\033':
                exit(0);  /* '\033' ?? ESC ?? ASCII ?R?[?h */
        default:
                break;
```

```
        }
        glutPostRedisplay();
}

// }E[X[C[x[[[g[[[[
void mouse(int button, int state, int x, int y) {
        if(state == GLUT_DOWN) {
                switch (button) {
                case GLUT_LEFT_BUTTON:
                        // N[[[b[N[[[[[zu[e[[[_[[lj[
                        g_ControlPoints.push_back(Vector2d(x, y));
                break;
                case GLUT_MIDDLE_BUTTON:
                break;
                case GLUT_RIGHT_BUTTON:
                        // [[[[[[[[[_[
                        if(!g_ControlPoints.empty()) {
                                g_ControlPoints.pop_back();
                        }
                break;
                default:
                break;
                }
                glutPostRedisplay(); // [e`[[
        }
}

// [[[C[[[v[[[O[[[[
int main (int argc, char *argv[]) {
        glutInit(&argc, argv);          // [[[C[u[[[[[[[[[[[
        glutInitDisplayMode(GLUT_RGBA|GLUT_DOUBLE); // [`[恬[[h[w[[
        glutInitWindowSize(800 , 800);  // [E[B[[[h[E[T[C[Y[[[w[[
        glutCreateWindow(argv[0]);      // [E[B[[[h[E[[[쐬
        glutDisplayFunc(display);       // [\[[[[[[[[w[[
        glutReshapeFunc(resizeWindow);  //
[E[B[[[h[E[T[C[Y[[[úX[[[[[Б[[[[[[[[w[[
        glutKeyboardFunc(keyboard);     //
[L[[{[[h[C[x[[[g[[[[[[[[[[[[w[[
        glutMouseFunc(mouse);           // [}
[E[X[C[x[[[g[[[[[[[[[[[[w[[
        glutMainLoop();                 // [C[x[[[g[≠[
        return 0;
}
```

スクリーンショット


./a.out


./a.out