Samuel LE BERRE
TP7 : Parametric surface

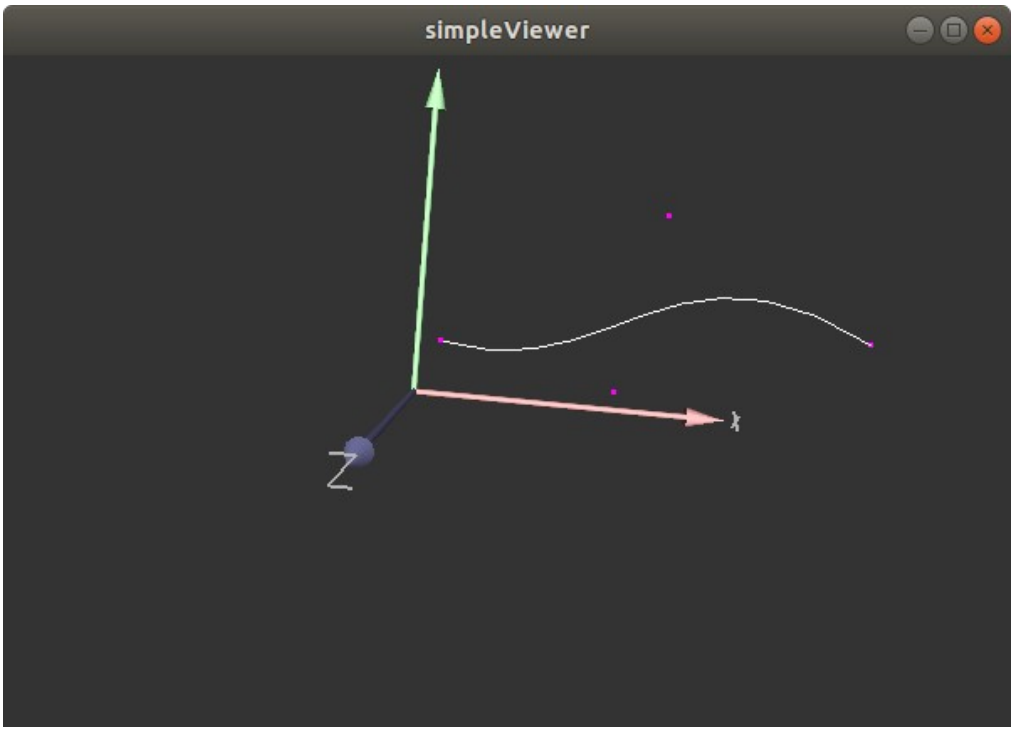
Exercice 1

Code :

```
    Vec p1 = Vec(1,2,3);
    Vec p2 = Vec(6,1,3);
    Vec p3 = Vec(7,6,3);
    Vec p4 = Vec(13,3,3);

    unsigned int i;
    vector<Vec> courbe = vector<Vec>();
    for(i=0; i<=10;i++){
        float t = i/10.0;
        courbe.push_back(p1*pow(1-t,3) + 3*p2*t*pow(1-t, 2.0) + 3*pow(t,2)*(1-
t)*p3 + pow(t,3.0)*p4);
    }
    glColor3f(1,0,1);
    glPointSize(3);
    glBegin(GL_POINTS);
    glVertex3fv(p1);
    glVertex3fv(p2);
    glVertex3fv(p3);
    glVertex3fv(p4);
    glEnd();
    glColor3f(1,1,1);
    glBegin(GL_LINE_STRIP);
    for(i=0; i<courbe.size();i++){
        glVertex3fv(courbe.at(i));
    }
    glEnd();
    glFlush();
```

Resultat



Exercice 2 :

Code :

```
Vec p1 = Vec(1,2,3);
    Vec p2 = Vec(6,1,3);
    Vec p3 = Vec(7,6,3);
    Vec p4 = Vec(13,3,3);

    Vec p5 = Vec(3,2,3);
    Vec p6 = Vec(8,1,3);
    Vec p7 = Vec(9,6,6);
    Vec p8 = Vec(15,3,3);

    Vec p9 = Vec(5,2,3);
    Vec p10 = Vec(10,4,3);
    Vec p11 = Vec(11,6,3);
    Vec p12 = Vec(17,3,5);

    Vec p13 = Vec(7,2,1);
    Vec p14 = Vec(12,1,3);
    Vec p15 = Vec(13,6,3);
    Vec p16 = Vec(19,3,3);

    vector<vector<Vec>> matrice = vector<vector<Vec>>();
    vector<Vec> ligne0 = vector<Vec>();
    ligne0.push_back(p1);
    ligne0.push_back(p2);
    ligne0.push_back(p3);
    ligne0.push_back(p4);
    vector<Vec> ligne1 = vector<Vec>();
    ligne1.push_back(p5);
    ligne1.push_back(p6);
    ligne1.push_back(p7);
    ligne1.push_back(p8);
    vector<Vec> ligne2 = vector<Vec>();
    ligne2.push_back(p9);
    ligne2.push_back(p10);
    ligne2.push_back(p11);
    ligne2.push_back(p12);
    vector<Vec> ligne3 = vector<Vec>();
    ligne3.push_back(p13);
    ligne3.push_back(p14);
    ligne3.push_back(p15);
    ligne3.push_back(p16);

    matrice.push_back(ligne0);
    matrice.push_back(ligne1);
    matrice.push_back(ligne2);
    matrice.push_back(ligne3);

    glColor3f(1,0,1);
    glPointSize(3);
    glBegin(GL_POINTS);
    glVertex3fv(p1);
    glVertex3fv(p2);
    glVertex3fv(p3);
    glVertex3fv(p4);
    glVertex3fv(p5);
    glVertex3fv(p6);
    glVertex3fv(p7);
    glVertex3fv(p8);
    glVertex3fv(p9);
```

```
    glVertex3fv(p10);
    glVertex3fv(p11);
    glVertex3fv(p12);
    glVertex3fv(p13);
    glVertex3fv(p14);
    glVertex3fv(p15);
    glVertex3fv(p16);
    glEnd();

    vector<Vec> courbe = vector<Vec>();
    for(int u=0; u<16; u++){
        for(int v=0; v<16; v++){
            int x=0,y=0,z=0;
            for(int i=0; i<4; i++){
                for(int j=0; j<4;j++){
                    x = x + bernstein_poly(u,4,i) * bernstein_poly(v,4,j) *
matrice.at(i).at(j).x;
                    y = y + bernstein_poly(u,4,i) * bernstein_poly(v,4,j) *
matrice.at(i).at(j).y;
                    z = z + bernstein_poly(u,4,i) * bernstein_poly(v,4,j) *
matrice.at(i).at(j).z;
                }
            }
            courbe.push_back(Vec(x,y,z));
        }
    }

    glBegin(GL_POINTS);
    for(unsigned int i=0; i<courbe.size();i++){
        glVertex3fv(courbe.at(i));
    }
    glEnd();
    glFlush();
```

Remarque :

Le problème provenant de mon code sst du à une incompréhension de l'algorithme de Bézier. U et v ne devrait pas être égale à 16 qui est le nombre de Points de Controle.