TP 2 Modélisation

Mateusz Birembaut

September 23, 2024

Contents

1 Exercice 1 : Courbe cubique d'Hermite

$$P(u) = F_1(u)P_0 + F_2(u)P_1 + F_3(u)V_0 + F_4(u)V_1$$

$$\text{avec} \begin{cases} F_1(u) = 2u^3 - 3u^2 + 1 \\ F_2(u) = -2u^3 + 3u^2 \\ F_3(u) = u^3 - 2u^2 + u \\ F_4(u) = u^3 - u^2 \end{cases}$$

Figure 1: Calculer la position des points de la courbe cubique d'Hermite

Avec P0, le point de départ et V0 sa tangente, P1 point d'arrivée et V1 sa tangente. Appuyer sur "2" pour l'afficher / cacher.

```
void HermiteCubicCurve (Vec3* points,Vec3 p0, Vec3 p1, Vec3 v0, Vec3 v1, long nbU){
    for (int i = 0 ; i < nbU; i++ ) {

        float j = float(i)/(nbU-1);

        float F1 = 2*pow(j, 3) - 3*pow(j, 2) + 1;
        float F2 = -2*pow(j, 3) + 3*pow(j, 2);
        float F3 = pow(j, 3) - 2*pow(j, 2) + j;
        float F4 = pow(j, 3) - pow(j, 2);

        Vec3 position = F1*p0 + F2*p1 + F3*v0 + F4*v1;

        points[i] = position;
    }

}
```

Figure 2: Code courbe cubique d'Hermite

1.1 Résultat

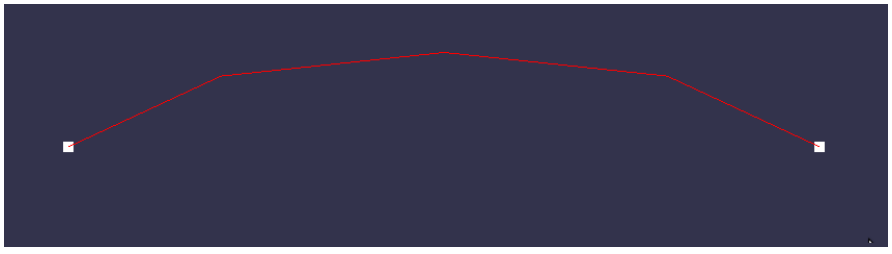

Figure 3: Courbe cubique d'Hermite

2 Exercice 2 : Courbe de Béziers par les polynômes de Bernstein

$$B_i^n(u) = \frac{n!}{i!(n-i)!} u^i (1-u)^{n-i} \quad , \quad i=0, \dots, n$$

Figure 4: Calculer la position des points d'une courbe de bézier

Avec n = degrés du polynôme (nombre de points de contrôles - 1). Appuyer sur "3" pour l'afficher / cacher.

```
void BezierCurveByBernstein (Vec3* pts,Vec3 pts_ctrl[], long nb_pts_ctrl, long nbU){
    int n = nb_pts_ctrl-1;
    float Bu;
    for (int j = 0 ; j < nbU ; j++) {
        float u = float(j)/(nbU-1);

        Vec3 somme = Vec3(0,0,0);

        for (int i = 0 ; i < nb_pts_ctrl; i++ ) {
            Bu = ((factorial(n) / (factorial(i)*factorial(n - i))) * pow(u,i) * pow((1-u), n-i));
            somme = somme + (  Bu * pts_ctrl[i] );
        }

        pts[j] = somme;
    }
}
```

Figure 5: Code Courbe de Bézier par les polynômes de Bernstein

2.1 Résultat

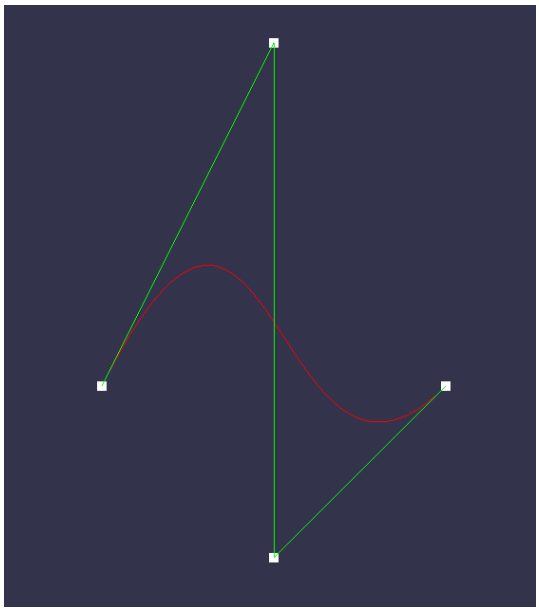

Figure 6: Courbe de Bézier par les polynômes de Bernstein

3 Exercice 3 : Courbe de Béziers par l'algorithme de Casteljau

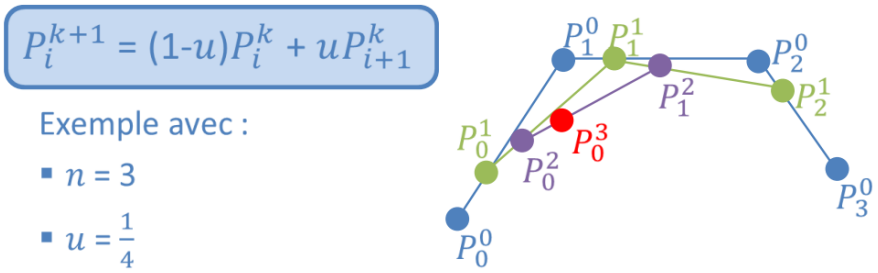

Figure 7: Calculer la position des points d'une courbe de bézier avec l'algo Casteljau

Appuyer sur "4" pour l'afficher / cacher.

```
void BezierCurveByCasteljau(Vec3* pts,Vec3 pts_ctrl[], long nb_pts_ctrl, long nbU){
    int n = nb_pts_ctrl-1;
    float Bu;

    for (int j = 0 ; j < nbU ; j++) {
        float u = float(j)/(nbU-1);

        Vec3 temp_pts[nb_pts_ctrl];

        for (int i= 0; i< nb_pts_ctrl; i++) {
            temp_pts[i] = pts_ctrl[i];
        }

        for (int r = 1; r <= n; r++) {
            for (int i = 0; i <= n - r; i++) {
                temp_pts[i] =  (1 - u)* temp_pts[i] + u * temp_pts[i + 1] ;
            }
        }

        pts[j] = temp_pts[0];

    }
}
```

Figure 8: Code courbe de Bézier par l'algorithme de Casteljau

3.1 Résultat

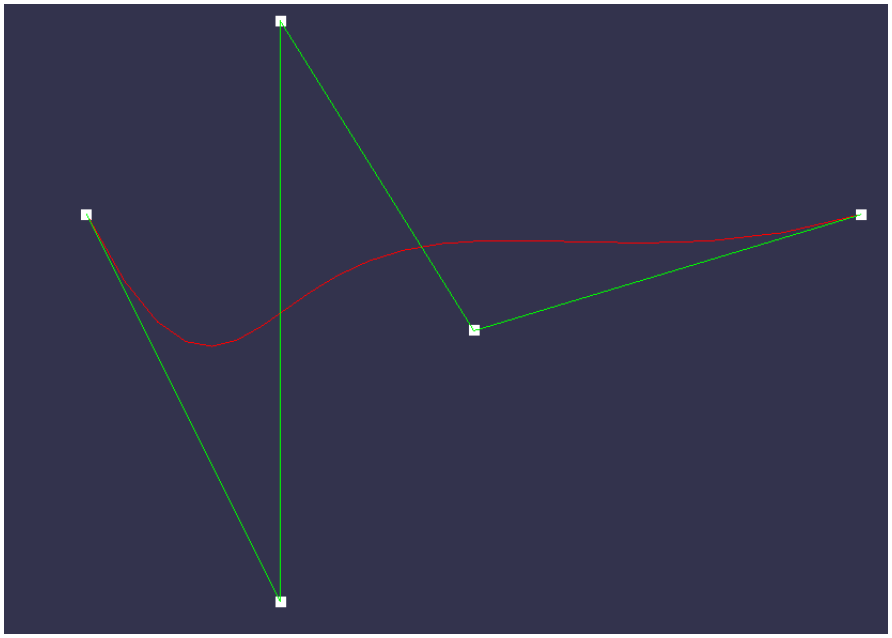

Figure 9: Courbe de Bézier par l'algorithme de Casteljau

4 Annexes

```
if(display_hermit){
    Vec3 p0 = Vec3 (0., 0., 0.);
    Vec3 p1 = Vec3 (2., 0., 0.);

    Vec3 v0 = Vec3 (1., 1., 0.);
    Vec3 v1 = Vec3 (1., -1., 0.);

    Vec3 pts[2] = {
        Vec3(0., 0., 0.),
        Vec3 (2., 0., 0.)
    };

    int nbU = 50;

    Vec3* points = new Vec3[nbU];

    HermiteCubicCurve(&points[0], pts[0],pts[1],v0,v1, nbU);

    DrawCurve(points, nbU, Vec3(1.,0.,0.));

    DrawPts(pts, 2, Vec3(1.,1.,1.));

}
```

Figure 10: Ex 1 : Appelle fonction

```
if(display_bezier_bernstein){
    Vec3 pts_ctrls[4] = {
        Vec3(0., 0., 0.),
        Vec3(0.5, 1, 0.),
        Vec3(0.5, -0.5, 0.),
        Vec3(1, 0., 0.)
    };

    // faire varier pour controler la "précision"
    int nbU = 50;

    Vec3* points = new Vec3[nbU];

    BezierCurveByBernstein(&points[0], pts_ctrls, 4 , nbU);

    //dessine la courbe verte entre les pts de controles
    DrawCurve(pts_ctrls, 4,Vec3(0.,1.,0.));

    //dessine la courbe rouge (résultat))
    DrawCurve(points, nbU, Vec3(1.0,0.,0.));

    //dessine les points de controles
    DrawPts(pts_ctrls, 4, Vec3(1.,1.,1.));
}
```

Figure 11: Ex 2 : Appelle fonction

```
if(display_bezier_casteljau){

    Vec3 pts_ctrls[5] = {
        Vec3(0., 0., 0.),
        Vec3(0.5, -1, 0.),
        Vec3(0.5, 0.5, 0.),
        Vec3(1, -0.3, 0.),
        Vec3(2, 0., 0.)
    };
    int nbU = 20;

    Vec3* points = new Vec3[nbU];

    BezierCurveByCasteljau(&points[0], pts_ctrls, 5 , nbU);

    DrawCurve(pts_ctrls, 5,Vec3(0.,1.,0.));

    DrawCurve(points, nbU, Vec3(1.0,0.,0.));

    DrawPts(pts_ctrls, 5, Vec3(1.,1.,1.));
}
```

Figure 12: Ex 3 : Appelle fonction

```
void DrawCurve( Vec3 TabPointsOfCurve[], long nbPoints, Vec3 color ) {
    glBegin(GL_LINE_STRIP);
    glColor3f(color[0], color[1], color[2]);
    for (unsigned int j = 0 ; j < nbPoints ; ++j ) {
        glVertex3f( TabPointsOfCurve[j][0] , TabPointsOfCurve[j][1] , TabPointsOfCurve[j][2] );
    }
    glEnd();
}
```

Figure 13: Code afficher une courbe (points = GL POINTS à la place de GL LINE STRIP)