Computer Graphic Assignment 5

Xiang Zhang CS560

Surface rendering is achieved by applying natural lighting effects to the visible surfaces. In this assignment, you are to implement the visible surface detection algorithm (i.e., back-surface removal) to display a Bezier surface patch. The illumination models, including the ambient light, diffuse reflection and specular reflection, are used for the surface shading. Two shading methods (flat shading and Gouraud shading) are applied. Surface details can also be added to the patch by using the texture mapping method. Through this assignment, you will practice the basic methods in surface rendering and display (back-face removal, shading and texture mapping), and simulate the lighting effect by using OpenGL.

(1) (6%) By using the curve interpolation method, we use a cubic parametric polynomial ($f(u) = au^3 + bu^2 + cu + d$) to fit four control points P0, P1, P2, and P3. This cubic polynomial curve passes through the four control points P0, P1, P2 and P3 when parameter u=0, 1/3, 2/3, 1 respectively. Find a matrix M which satisfies the condition f(u) = UMP

(where U= $\begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}$, P= $\begin{bmatrix} P0 & P1 & P2 & P3 \end{bmatrix}^T$, f(u) is a vector which has three components (x(u), y(u), z(u)); Each control point $Pi\ (i = 0,1,2,3)$ is a vector representing x, y, z coordinate as (Pi_x, Pi_y, Pi_z)).

Given the boundary condition:

$$f(0) = p1$$
$$f(1) = p2$$
$$f\prime(0) = (p2 - p0)/2$$
$$f\prime(1) = (p3 - p1)/2$$

Given a parametric cubic polynomial:

$$f(u) = au^3 + bu^2 + cu + d$$
$$f\prime(u) = 3au^2 + 2bu + c$$

$\Rightarrow$

$$f(0) = d$$
$$f(1) = a + b + c + d$$
$$f\prime(0) = c$$
$$f\prime(1) = 3a + 2b + c$$

$\Rightarrow$

$$a = 2f(0) - 2f(1) + f\prime(0) + f\prime(1)$$
$$b = -3f(0) + 3f(1) - 2f\prime(0) - f\prime(1)$$
$$c = f\prime(0)$$
$$d = f(0)$$

so:

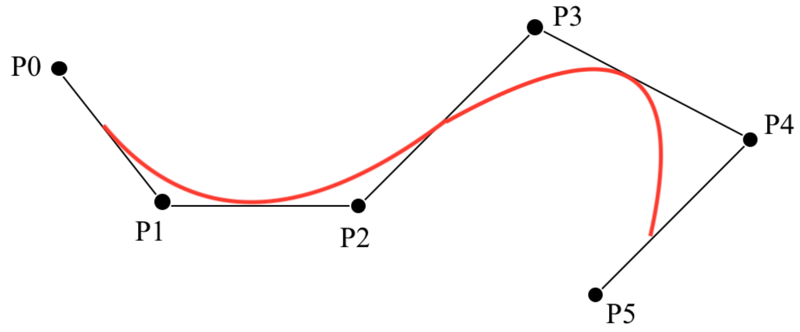
$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} p0 \\ p1 \\ p2 \\ p3 \end{bmatrix}$$

$$M = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix}$$

(2) (6%) The uniform B-spline curves can be used to approximate the curve segments which are determined by a number of control points. Given six control points as shown below, use the B-spline curves to approximate the six control points in the following case: the degree of the B-spline polynomial is 2.

Indicate how many curve segments there are in this case;
Sketch the B-Spline curves on the following graph and label the curve segments.

n +1 = 6 => n = 5
so curve segments: n - 2 = 3



(3) (8%) If the control point P5 is repeated as shown below. P5 will be used more than once in the evaluation for one curve segment. If we use the uniform cubic B-spline curves to fit the control points: p0, p1, p2, p3, p4, p5, p5, p5.

(a) (2%) Indicate the range of the knot values (i.e., 0, … ,m).
(b) (3%) Plot the blending functions within the knot value range (0, … ,m).
(c) (3%) Sketch the uniform cubic B-spline curves to fit the control points: p0, p1, p2, p3, p4, p5, p5, p5. Label all the curve segments explicitly.

n + 1 = 8 => n = 7
d – 1 = 2 => d = 3
knot = n + d + 1 = 11

(a) : {0,1,2,3,4,5,5,5,5,5}

(b) :

$$\mathbf{P}(u) = \sum_{k=0}^{n} \mathbf{p}_k B_{k,d}(u) \qquad (1)$$

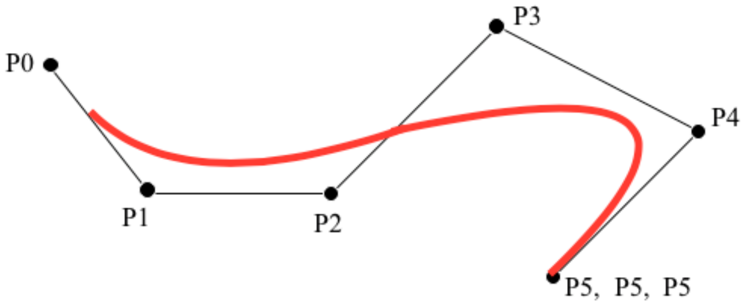
where
n+1 control points: $\mathbf{p}_k$
d is any integer value in $[2, n+1]$
$B_{k,d}$ is defined by the Cox-deBoor formulas:

$$B_{k,1}(u) = \begin{cases} 1, & if\ u_k \le u < u_{k+1} \\ 0, & otherwise \end{cases}$$

$$B_{k,d}(u) = \frac{u - u_k}{u_{k+d-1} - u_k} B_{k,d-1}(u) + \frac{u_{k+d} - u}{u_{k+d} - u_{k+1}} B_{k+1,d-1}(u) \qquad (2)$$

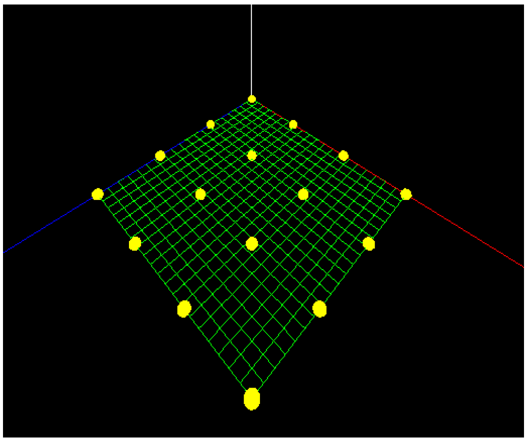
where $u_k \le u < u_{k+d}$

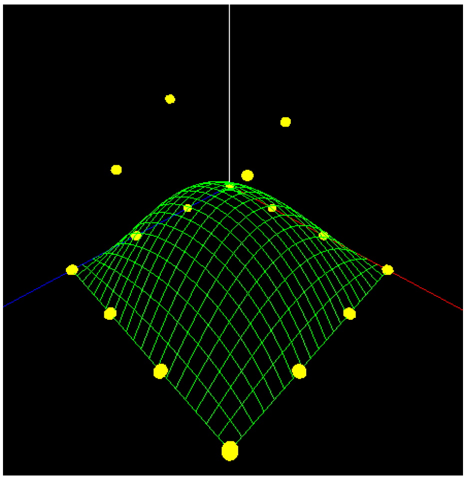
n = 7, d = 3



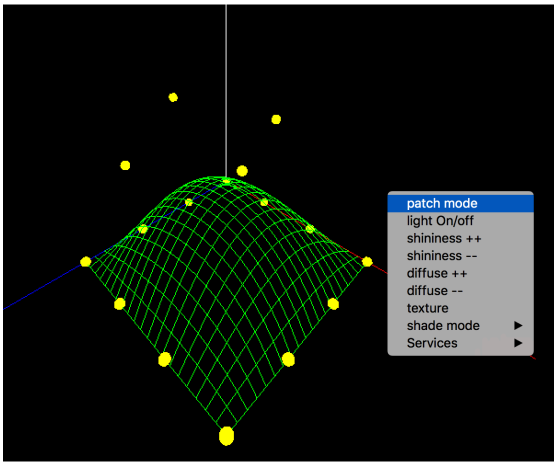
Programming part:
g++ assignment5.cpp -o assignment5-framework OpenGL -framework GLUT

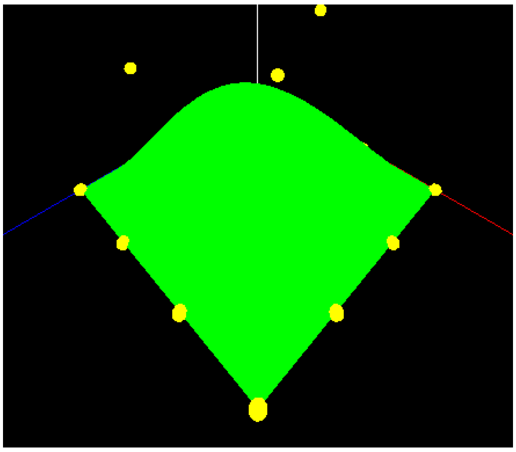
assigment5.cpp contains the whole codes for Bezier surface deformation, Surface Shading Generation and Bezier surface rendering.
evaluateBezierSurface is the method of generating Bezier patches.It invoke the evalBezierCurve() function for both U and V.
Further more, bool canbeVisible() calculates the triangle whether should be seen or not from the view port.
Drag any of the 16 control points could generate the Bezier surface synchronously.



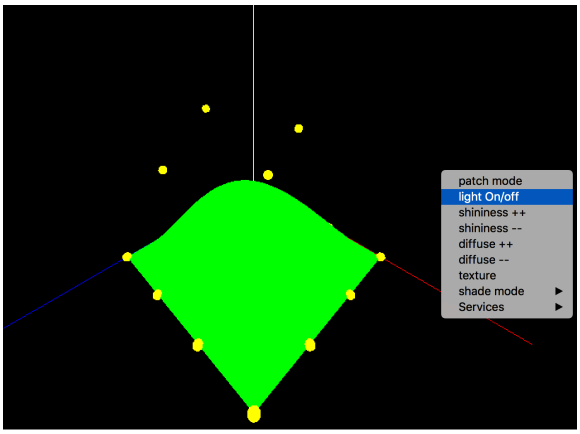


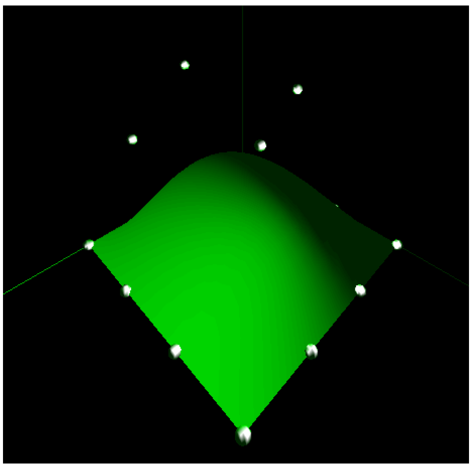
Click the right mouse to show the menu:
Choose patch mode



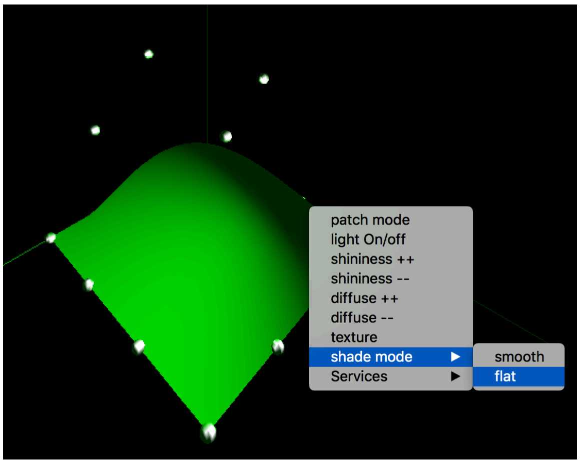


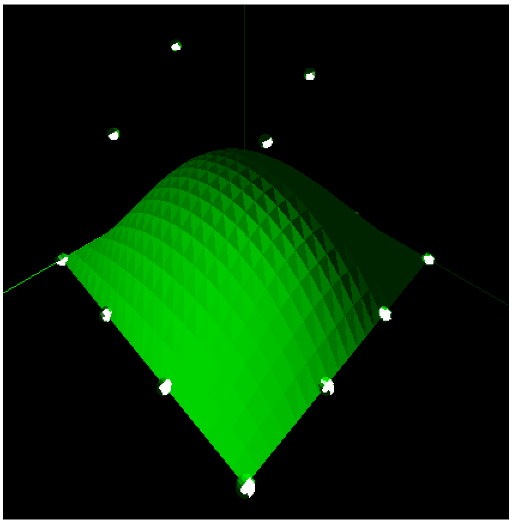
Choose light on/off



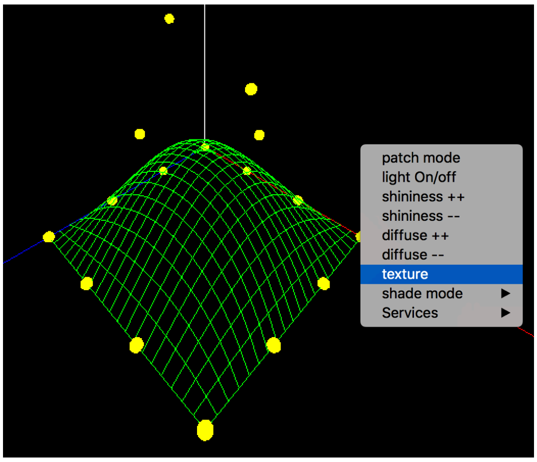


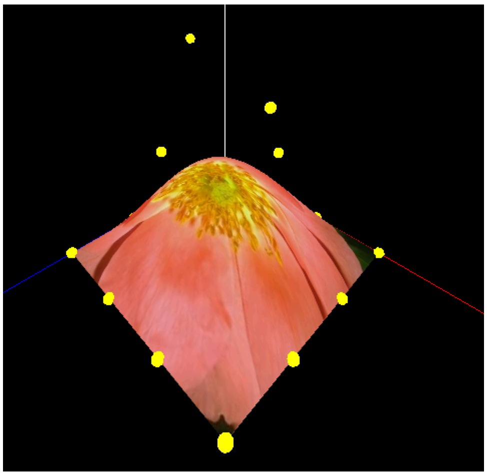
Shade mode is to change the shade mode between flat mode and smooth mode (default is smooth)





Texture is to set the Bezier surface with “flower.bmp”





Others, shininess and diffuse is changing the lighting parameters.

Last, the for direction key in keyboard:(Up, Down, Left, Right) works for rotation of the camera.