2IV60 Computer Graphics

Examination, January 25 2016, 9:00 – 12:00

This examination consist of **four** questions with in total 16 subquestions. Each subquestion weighs equally. In all cases: **EXPLAIN YOUR ANSWER. Use sketches where needed to clarify your answer. Read first all questions completely.** If a function is asked, then a description in steps or pseudo-code is expected, which is clear enough to be easily transferred to real code. Aim at compactness and clarity. The use of the book, copies of slides, notes and other material is not allowed.

1 We consider some basic concepts of computer graphics.

a) Explain what the view-up vector is and what the effect is when it is changed.

A view-up vector is used in the specification of the projection of a scene to the screen. It is a vector in world-coordinates that should be projected such that it is mapped on the vertical axis of the screen. Changes in the view-up vector lead to a rotation of the projected image around the center of the screen.

b) A basic model for specular reflection is $I_s = k_s I_L (V \cdot R)^n$. What do the parameters k_s , I_L , V, R, and n mean? Use a sketch to illustrate this.

k_s is a material property that indicates how strongly the material of the surface reflects light. I_L is the intensity of the incident light. V is a unit vector in the direction of the viewer. R is a unit vector that points in the direction of perfectly reflected light. n models the smoothness of the surface: a high value gives a small highlight.

c) Describe the z-buffer (or depth-buffer) algorithm for hidden surface removal.

In the z-buffer algorithm, for each pixel the depth of scanned surfaces is maintained in an 2-dimensional array. Initially, all values are set to a large value. Next, geometric objects (typically polygons) are rendered in arbitrary order. During rendering, polygons are scanned, and per pixel the depth is calculated. This depth is compared with the value stored in the z-buffer. If the value is less than the stored value, the new value is stored, and the corresponding pixel is set to the color of the surface.

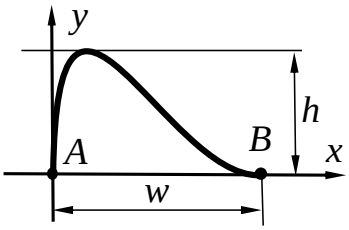
In pseudo-code:

```
var zbuf: array[N,N] of real;          { z-buffer: 0=near, 1=far }
    fbuf: array[N,N] of color;         { frame-buffer }
For all 1<= i, j <=N do
    zbuf[i,j] := 1.0; col[i,j] := BackgroundColour;
For all polygons do                    { scan conversion }
    For all covered pixels (i,j) do
        Calculate depth z;
        If z < zbuf[i,j] then { closer! }
            zbuf[i,j] := z;
            fbuf[i,j] := surfacecolor(i,j); zbuffer: array[0..N, 0..N];
```

d) Describe a method to test if a point P is inside or outside a polygon.

Take a half-line $P + Vt$, $t > 0$, where V is an arbitrary direction. Calculate the number of intersections of this half-line with the polygon. If this number is odd, the point is inside the polygon, else it is outside.

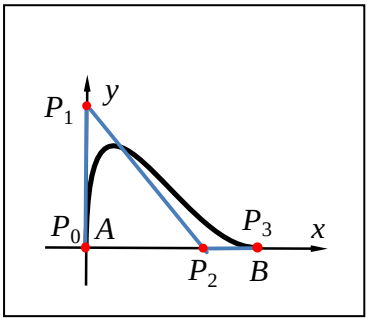
2 We want to draw a curve. The curve starts at point $A = (0, 0)$ and ends at point $B = (w, 0)$; at A the curve is tangent with the y-axis; at B the curve is tangent with the x-axis (see figure). The curve must be smooth everywhere. Answer the following questions, with formal definitions of the points *and* sketches:



a) Suppose we use one cubic Bézier segment for the curve, where a segment is given by

$$P(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3.$$

Draw a control polygon and give the possible positions of the control points (ignoring h), using new parameters where necessary.



The first and last point coincide with A and B:

$P_0 = A = (0, 0)$

$P_3 = B = (w, 0)$

The point P_1 must be located at the y-axis, such that the curve is initially tangent to the y-axis:

$P_1 = (0, a)$.

The point P_2 must be located at the x-axis, such that the curve is finally tangent to the x-axis:

$P_2 = (b, 0)$.

We have two new parameters a and b that control where on the axes the intermediate points are located. To get a similar shape as in the figure, one can pose the additional constraints that $a > 0$ and $b < w$.

b) What is the value of h, the maximum value of $P_y(t)$, $0 \le t \le 1$, given a set of (parametrized) control points?

Using the fact that only P_1 has a non-zero y-coordinate, we find that the value of $P_y(t)$ is given by

$$P_y(t) = 3t(1-t)^2 a = (3t - 6t^2 + 3t^3)a$$

$P_y(t)$ reaches a maximum value when its derivative is zero:

$$P'_y(t) = (3 - 12t + 9t^2)a = 0$$

Solving for t gives t=1/3 and 1. The last value corresponds with P_3 , the first with the interior maximum. Substitution gives $h = P_y(1/3) = 4a/9$.

c) Suppose we use two quadratic Bézier segments P and Q for the curve, given by

$P(t) = (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2$, and $Q(t) = (1-t)^2 Q_0 + 2t(1-t)Q_1 + t^2 Q_2$.

Draw a control polygon and give the possible positions of the control points, using new parameters where necessary.

The first control point of P and the last control point of Q coincide with A and B:

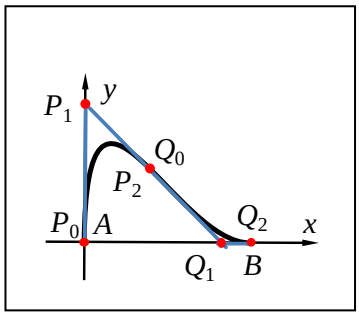
$P_0 = A = (0, 0)$

$Q_2 = B = (w, 0)$

The point P_1 must be located at the y-axis, and point Q_1 must be located at the x-axis, similar to the previous case:

$P_1 = (0, a)$

$Q_1 = (b, 0)$.



The points P_2 and Q_0 should coincide, to insure continuity of position. Also, they should be located on the line segment P_1Q_1 to insure continuity of direction (and hence, smoothness of the curve). This can be specified as:

$P_2 = Q_0 = (1-u)P_1 + u\,Q_1 = (bu, a-au)$, with $0 < u < 1$.

In total, we have three parameters: a, b, and u.

d) Again, we use two quadratic Bézier segments P and Q, but now for arbitrary points A and B, and arbitrary tangent directions U and V at start and end. Give the possible positions of the control points, using new parameters where necessary.

The positions of the control points are as follows:

$P_0 = A$

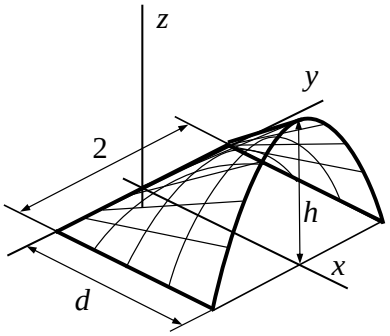
$Q_2 = B$

$P_1 = A + aU$

$Q_1 = B + bV$

$P_2 = Q_0 = (1-u)P_1 + u\,Q_1$ with $0 < u < 1$.

3 We consider a surface $z = axy^2+bx$, with $a < 0$, $0 \le x \le d$, and $-1 \le y \le 1$, see the figure. Answer the following questions.



a) What values should a and b have such that for all corners $z=0$ and that the maximum value of z is equal to h?

For the corners with $x=0$, $z = 0$ is satisfied for arbitrary a and b. The corner with $x=d$ and $y=1$ gives the constraint

$ad + bd = 0$.

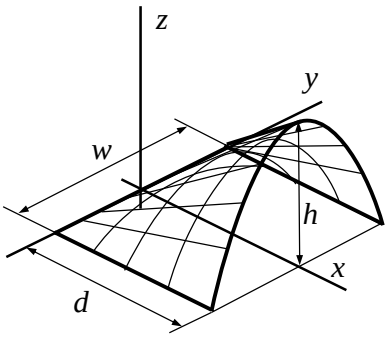
The corner with $x=d$ and $y=-1$ gives the same constraint.

The maximum value h is achieved at the point with $x=d$ and $y=0$. Substitution gives the constraint $h = bd$, hence

$b = h/d$.

Substitution in the first constraint gives $ad + h = 0$, hence

$a = -h/d$



b) Give a parametric description of the surface: define the surface as $P(u,v)$, $0 \le u,v \le 1$.

We align the parameter u with the coordinate x: If $x = 0$, then $u = 0$; if $x = d$, then $u = 1$. This leads to $x = ud$. More formally, we need a linear mapping $x = pu + q$. Substitution of the requirements gives $0 = q$ and $d = p$.
Furthermore, we align the parameter v with the coordinate y. Here, if $y = -1$, then $v = 0$; if $y = 1$, then $v = 1$; and also, if $y = 0$, then $v = 1/2$. This leads to $y = 2v - 1$.

We can now define $P(u,v)$ as $P(u,v) = (x(u), y(v), z(x(u), y(v)))$.
Substitution of the results and using the given definition of z in terms of coordinates gives

$P(u,v) = (ud, 2v - 1, aud(2v-1)^2 + bud)$

$= (ud, 2v - 1, -hu(2v-1)^2 + hu)$

$= (ud, 2v - 1, 4huv(1-v))$

c) Given a point on the surface (by coordinates or by parameter value, choose yourself), give an expression for a normal vector N on the surface for that point.

The easiest route is to define the surface as implicitly by $f(x, y, z) = 0$, and to use $N = \nabla f = (\partial f / \partial x, \partial f / \partial y, \partial f / \partial z)$. Here $f(x,y,z) = axy^2+bx - z$, elaboration gives: $N = (ay^2 + b, 2axy, -1)$. Substitution of a and b gives $N = (-hy^2/d + h/d, 2hxy/d, -1)$. Scaling with d/h gives a slightly simpler version: $N = (1-y^2, 2xy, -d / h)$.

d) We want to apply texture mapping. When viewed from the top (in the direction of the negative z- axis) we want to see a pattern of square tiles with width r. The base texture is given in a unit square $0 \leq s, t \leq 1$; furthermore, clamping (use of only the fractional value of texture parameters) is enabled for texture wrapping. Give the texture coordinates $s(u,v)$ and $t(u, v)$ to achieve this.

We can obtain this effect by mapping s linearly to u and t linearly to v, and we only have to apply appropriate scaling. In the u-direction (aligned with the x-axis) the width is d. Here we can fit d/r tiles, hence we use $s(u,v)=ud/r$. In the v-direction (aligned with the y-axis) the width is 2. We can fit $2/r$ tiles, hence we use $t(u,v)=2v/r$.

4 We consider the animation of a rotating object in 2D. Suppose that a round object (say, a wheel) is given in local coordinates, defined inside a circle with radius 1, centered around the origin. We want to make an animation of this wheel. The wheel should have a radius r, and roll over the x-axis: the line y=0. At time $t = 0$ the wheel touches the origin, the center has a constant velocity v. The wheel rotates such that there is no slip between the wheel and the x-axis.

a) Derive a formula for the rotation angle $\alpha(t)$ of the wheel as a function of time t.

We observe that if the center is displaced over distance vt, the wheel should be rotated such that a point on the boundary moves over a corresponding distance along the boundary of the wheel. This distance is $|\alpha(t)|r$, and should be equal to vt. Hence, the absolute value of the rotation angle is equal to vt / r.
This is a correct answer (and in line with the arrow drawn in the figure). If we adopt the convention that rotations are counterclockwise, we get $\alpha(t) = -vt/r$.

b) Let A be the point on the wheel that touches the origin at $t = 0$.
What is the position $A(t) = (A_x(t), A_y(t))$ of this point as a function of time t?

The center of the wheel has position $C(t) = (vt, r)$. A vector $V(t)$ from the center to the point A is given by $V(t) = r(\cos(\alpha(t) - \pi/2), \sin(\alpha(t) - \pi/2))$: take a unit-vector (1,0), rotate over $\alpha(t)$, apply an extra rotation over $\pi/2$ counterclockwise, scale with r, Adding $C(t)$ and $V(t)$ gives the position $A(t)$:

$A(t) = (vt + r \cos(\alpha(t) - \pi/2), r + r \sin(\alpha(t) - \pi/2))$.

Using $\alpha(t) = -vt/r$ and standard goniometry (and/or looking at the picture), this can be rewritten as

$A(t) = (vt - r \sin(vt/r), r - r \cos(vt/r))$.

Assume that a function *DrawWheel*(*M*) is available to draw the wheel, subject to a homogenous transformation *M*. Points *X'* in world coordinates are derived from local coordinates *X* by *X'=MX*, where *M* is a homogenous transformation matrix.

Furthermore, suppose that a standard set of functions to define transformation matrices is available:

- $T(d_x, d_y)$ gives a translation matrix over a vector (d_x, d_y);
- $S(s_x, s_y)$ gives a scaling matrix with scale factors s_x and s_y for the axes; and
- $R(a)$ gives a counterclockwise rotation over a radians around the origin.

c) Define a matrix $M_1(t)$ such that the wheel rolls along the *x*-axis according to the specification.

In terms of global transformations:

1. we first scale the wheel ($S(r, r)$);
2. we rotate the wheel over an angle $\alpha(t)$;
3. next we translate it vertically, such that the bottom touches the origin ($T(0, r)$);
4. we translate it horizontally over a distance vt : ($T(vt, 0)$).

Combining all this gives $M_1(t) = T(vt, r)\ R(-vt/r)\ S(r, r)$

d) Define a matrix $M_2(t)$ such that the wheel rolls over a line $y = ax$ instead of over the *x*-axis. Matrix $M_1(t)$ can be reused.

Looking at the figure, we see that the complete figure is rotated over an angle arctan(*a*). (A unit step in *x* direction gives a displacement *a* in *y*-direction, hence the tangens of the angle between the line and the *x*-axis is *a*.) Hence, we can obtain $M_2(t)$ by obtaining a global rotation on the transformed wheel of the previous question:

$M_2(t) = R(\arctan(a))\ M_1(t)$.