2IV60 Computer Graphics

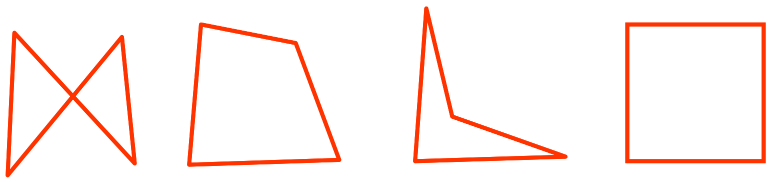
Examination, January 26 2015, 9:00 – 12:00

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 an algorithm 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. Use additional functions and procedures if desired. Give a short description of input and output for each function and procedure. 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) Draw examples of a non-simple, a convex, a concave, and a regular polygon.

For instance:



non-simple   convex   concave   regular

b) Suppose that we use a standard set-up of a virtual camera: an eye-point, a center-point, and a view-up vector. What is the effect on the screen if we change the view-up vector?

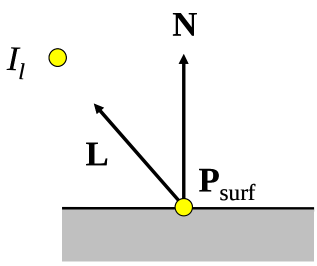
The effect is that the image shown will rotate around the center of the viewport: the view-up vector has to be specified in addition to eye-point and center-point to fix the rotation of the camera along the line through these points.

c) Give the equation for the standard model of diffuse reflection, and a sketch and a description of the parameters.

$I = k_d I_l (\mathbf{N} \cdot \mathbf{L})$ if $\mathbf{N} \cdot \mathbf{L} > 0$

a.   otherwise

where $\mathbf{N}$ is a unit-length normal on the surface; $\mathbf{L}$ is a unit-length vector pointing from the point $\mathbf{P}_{surf}$ at the surface towards the light-source; k_d is the diffuse reflection coefficient of the surface; and I_l is the intensity of the light source. See the sketch below.



d) What are three optical effects that can be simulated with ray-tracing and not with a basic shading model?

Cast shadows, mirroring reflection, and transparency with refraction.

2 A *ruled surface* can be described as $S(u, v) = P(u) + vQ(u)$, with $u_0 \le u \le u_1$ and $v_0 \le v \le v_1$: a line segment with direction $Q(u)$ is moved along a curve P, sweeping out a surface. We use this to model surfaces that result from twisting flexible strips.

a) Take a flexible strip with length d and width w and align its center-line with the positive z-axis. Next, fix the bottom of the strip and rotate the top around the z-axis over 180 degrees. Give $P(u)$, $Q(u)$, and the associated values for u_0, u_1, v_0, and v_1 for the resulting surface.

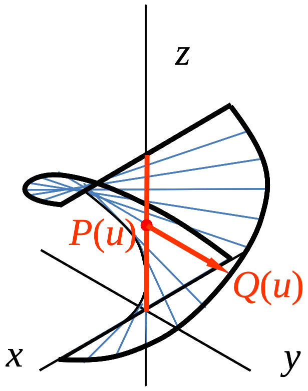
A natural choice is to match P with a part of the z-axis:

$P(u) = (0, 0, ud)$, with $0 \le u \le 1$.

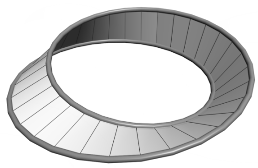
For $Q(u)$ we use a vector parallel to the plane z=0 that rotates around the z-axis with increasing u, with length $w/2$:

$Q(u) = (\cos(\pi u), \sin(\pi u), 0)\, w/2$.

Note that for u=1 this vector is rotated over an angle of 180 degrees. Finally, we choose $-1 \le v \le 1$, such that the line segment that is moved is centered around the center line. See the figure below.



b) The figure on the right shows a *Moebius strip*, obtained by gluing the begin and end of a twisted strip with length d and width w. For the curve P we use again the centerline, which is now a circle in the plane z = 0. Give $P(u)$, $Q(u)$, and the values for u_0, u_1, v_0, and v_1.



The centerline of the strip is a circle with radius $r = d/2\pi$. Hence, we define $P(u)$ as:

$P(u) = (\cos(u), \sin(u), 0)\, r$, with $0 \le u \le 2\pi$.

Several variations are possible here, like for instance:

$P(u) = (r\cos(2\pi u),\ r\sin(2\pi u), 0)$, with $0 \le u \le 1$.

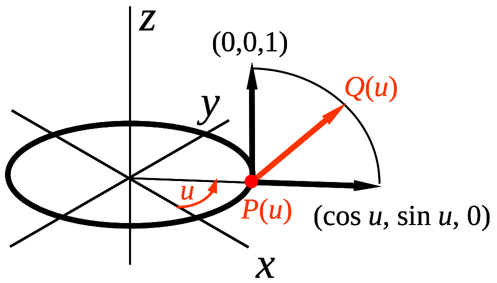
The vector $Q(u)$ rotates around the centerline again. Hence, $Q(u)$ should be located at a circle, in a plane that contains a vector from the origin to $P(u)$, and the z-axis (see figure). We define two vectors to define this circle: $\mathbf{a} = (\cos u, \sin u, 0)$ and $\mathbf{b} = (0, 0, 1)$. $Q(u)$ can be defined as $Q(u) = \cos(t(u))\, \mathbf{a} + \sin(t(u))\, \mathbf{b}$. The parameter t should be chosen such that $t(0) = 0$ and $t(2\pi) = \pi$, to get a half turn. We can achieve this by using $t(u) = u/2$. Substituting everything we get:

$Q(u) = \cos(u/2)(\cos u, \sin u, 0) + \sin(u/2)\,(0, 0, 1)$

or

$Q(u) = (\cos(u)\cos(u/2), \sin(u)\cos(u/2), \sin(u/2))$.

For the range of the parameter v we use $-w/2 \le v \le w/2$.



c) A Moebius strip has one single boundary, a curve $B(t)$ with $t_0 \le t \le t_1$. Give $B(t)$ and the associated values for t_0 and t_1 for the model of question b).

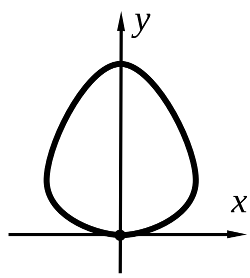
The curve can be defined as $B(t) = P(t) + Q(t)w/2$: we use the maximum value for parameter v to get a point at the boundary. To get the complete curve, we must follow the curve for two rotations around the z-axis. Hence, we use $0 \le t \le 4\pi$.

d) We want to decorate the Moebius strip with a pattern of tiles via texture mapping. We want M tiles along the strip and N tiles across the width of the strip. We use a standard unit square texture, representing one tile, and use texture wrapping in all directions to get a pattern of tiles. Give the texture parameters $s(u,v)$ and $t(u, v)$ for this.

Suppose that we align the s parameter with the u parameter, along the centerline. We should then satisfy $s(u_0, v) = 0$ and $s(u_1, v) = M$. We can achieve this by choosing $s(u, v) = M (u - u_0) / (u_1 - u_0)$, hence $s(u, v) = Mu / 2\pi$.
Following the same reasoning, we find $t(u, v) = N (v - v_0) / (v_1 - v_0)$, hence $t(u, v) = N (v - w/2) / w$. Also, we can use $t(u, v) = N v / w$, as we could also use $t(u, v_0) = -M / 2$ and $t(u, v_1) = M / 2$ for the constraints on the boundaries.

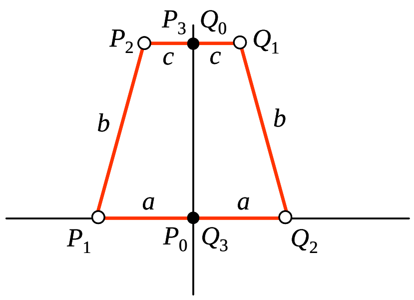
---

3 We want to draw a closed, *smooth* curve. The curve must be tangent to the x-axis at the origin, and left-right symmetric across the y-axis. We use Bézier segments, and one of these starts at the origin. Answer the following questions, with formal definitions of the points *and* sketches:



a) Suppose we use two cubic Bézier segments for the curve, where a segment is given by $P(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3$.

Give the possible positions of the control points. How many degrees of freedom remain?



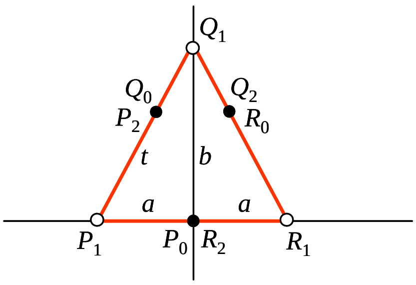
We use (see figure):

$P_0 = (0, 0)$	$Q_0 = (0, b)$
$P_1 = (-a, 0)$	$Q_1 = (c, b)$
$P_2 = (-c, b)$	$Q_2 = (a, 0)$
$P_3 = (0, b)$	$Q_3 = (0, 0)$

This assures that all constraints are met. We have three free parameters a, b and c.

b) How to select the points in this cubic case such that a circle with radius r is approximated?
First, to obtain symmetry across a horizontal axis, we use $c = a$. We use the two remaining free parameters to fix a few points on a circle. First, we use $b = 2r$, such that the point P_3 is located on a circle with radius r, centered around $(0, r)$.
To make the width of the curve equal to $2r$, we require that a point half-way a segment is located at a distance r from the y-axis. For the x-value of the first segment we find that here $P_x(1/2) = -3a/4$. Equating this to $-r$ gives $a = 4r/3$.

c) Suppose we use three quadratic Bézier segments for the curve. Give the possible positions of the control points. How many degrees of freedom remain?



We use (see figure):

$P_0 = R_2 = (0, 0)$;

$P_1 = (-a, 0)$ and $R_1 = (a, 0)$; and

$Q_1 = (0, b)$;

similar as in the previous question. The points Q_0 and Q_2 should be equal to P_2 and R_0 , respectively. Furthermore, Q_0 should be located on the line segment P_1Q_1 to ensure continuity at the transition from segment P to segment Q. We can enforce that by setting

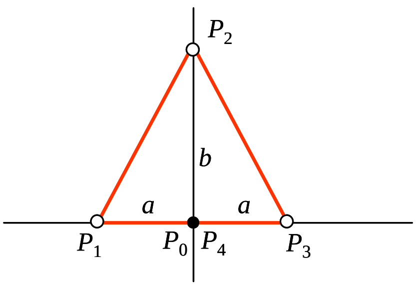
$P_2 = Q_0 = (1-t)P_1 + t\,Q_1 = (at-a, bt)$.

where t is a third control parameter. Symmetry across the y-axis leads to

$Q_2 = R_0 = (1-t)R_1 + t\,Q_1 = (a-at, bt)$.

Again, we have three degrees of freedom, *i.e.*, the parameters a, b, and t.

d) Suppose we use one fourth-degree Bézier segments for the curve. Give the possible positions of the control points. How many degrees of freedom remain?



Such a fourth-degree Bézier segment has five control-points. We position these as follows:

$P_0 = P_4 = (0, 0)$;

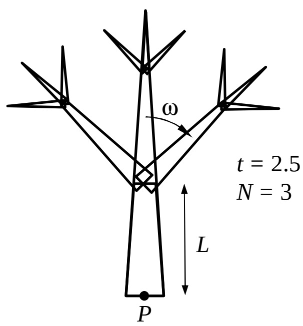
$P_1 = (-a, 0)$ and $P_3 = (a, 0)$; and

$P_2 = (0, b)$.

Here we have two degrees of freedom: a and b.

---

4 We want to draw a growing tree in 2D. The root of the tree is at a point P, the age of the tree is t years. The tree consists of branches, where each branch is modelled as a trapezium (older branches, $t > 1$) or a triangle (young branches, $t < 1$). The length of each branch increases linearly to L cm in its first year. The width of a branch at a position l ($0 \le l \le L$) increases with D cm per year, starting from the time that the branch reaches length l. When a branch is one year old, N subbranches are created. The directions of these subbranches are distributed over angles in the interval $[-\omega, \omega]$, $0 < \omega < \pi / 2$, where an angle of 0 radians is a continuation in the same direction as the original branch.



We want to define a function *DrawTree*(float t) that draws such a tree, and which gives a nice animation when called with small increments of t. We assume that a function *DrawTrapezium*(float a) is available to draw trapeziums, where the coordinates of the four vertices of this trapezium are $(1, 0)$, $(a, 1)$, $(-a, 1)$, and $(-1, 0)$. The use of a=0 gives a triangle. The trapezium is defined in a local coordinate frame; global positions X are derived from local coordinates X' by X=MX', where M is a homogenous transformation matrix, initially equal to the identity matrix. Furthermore, suppose that a standard set of functions to define transformation matrices is available, *i.e.*,

- $T(V)$ gives a translation matrix over a vector V;
- $S(s_x, s_y)$ 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.

An incomplete version of *DrawTree* is given below:

```
Matrix M ;    // current transformation matrix, similar to the model matrix of OpenGL

DrawTree(float t);    // Recursively draw a tree, where the age of the root is t.
{
    Matrix MS;
    MS = M ;                          // Save the current transformation
    w0 = …; w1 = …; b = …;            // Question a)
    M = …;                            // Question b)
    DrawTrapezium(…);                 // Draw the current branch

    if t < 1 then exit;               // A young branch has no subbranches
    for i = 0 to N–1 do               // Draw N subbranches
    {
      M = …. ;                        // Question c).
      DrawTree(t – 1);                // Recursively draw subtree
    }
    M = MS;                           // Restore the current transformation
}
```

Answer the following questions:

a) Calculate the widths w_0 and w_1 at the start and end of the current branch, as well as its length b.
The age at the root is t, and the rate of growth is D, hence we find

$w_0 = Dt$.

For the width at the end and the length, we must make a distinction between the case $t < 1$ (young branch, triangle shaped) and $t \geq 1$ (older branch, trapezium shaped). We find:

$t < 1$: $w_1 = 0$; $b = tL$ and
$t \geq 1$: $w_1 = (t–1)D$; $b = L$.

or, more compactly:

$w_1 = \max(0, (t–1)L$ and
$b = \min(tL, L)$

The age at the end of a branch is 1 year less than at the root, hence the factor $(t–1)$.

b) Give the transformation matrix M needed to draw the first branch of this tree, and the value of the parameter a of *DrawTrapezium.*
For a we use w_1 / w_0 , as the ratio $a/1$ of the top and bottom of the trapezium should match with the ratio w_1 / w_0 of the end and begin width of the branch. A case distinction is not needed here, we solved that already in question a).
We have to scale the trapezium, to get the right size, with $S(w_0 /2, b)$. Finally, we left multiply this matrix with the global transformation M (or M_S), and move everything to position P, giving

$$M = T(P)M\, S(w_0/2 , b).$$

c) Give the transformation matrices M to produce the N subbranches/subtrees.

We first translate the current branch to its end with $T(0, b)$. Next, we rotate the branch around its local origin. Suppose that branch 0 is rotated over $-\omega$, and branch $N-1$ over ω. The increment for each next branch is equal to $2\omega / (N-1)$. Hence, the rotation to be used is $R(2\omega i / (N-1) - \omega)$. Finally, we apply the global transformation M_S. Using M is an error, as this matrix has been scaled non-uniformly in question b). The answer is thus

$$M = M_S\, T(0, b)\, R(2\omega i / (N-1) - \omega) .$$

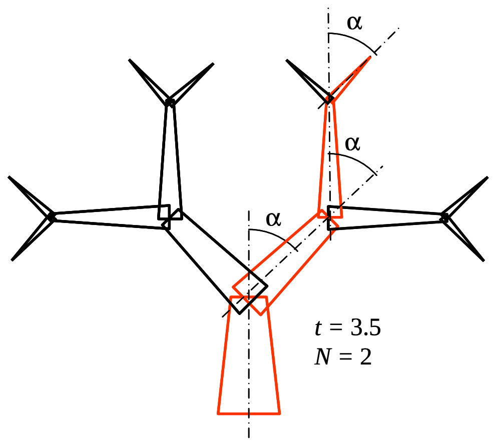
Also, as we can be sure that in this situation $b=L$, we can use

$$M = M_S\, T(0, L)\, R(2\omega i / (N-1) - \omega) .$$

d) What is the height $H(t)$ (the maximum y coordinate for all branches) of a tree of age t ? Take into account whether N is odd or even.

If N is odd, like in the original given drawing, the situation is simple. There is a central branch, which grows in the direction of the positive y-axis, hence here

$$H(t) = Lt.$$



If N is even, the situation is more complex. As an example, we have drawn a case for $N = 2$. We see that we can follow a path of branches, which direction alternates between vertical and with an offset angle α from the vertical direction. This angle α is given by $\omega / (N-1)$: half of the angle between two successive branches, see the previous question.

Let $k = \lfloor t \rfloor$ denote the number of complete years that the tree has grown. [Note: introducing some intermediate variables, like α and k, simplifies the following. Especially if these variables have a clear meaning]. Again, we have to make a case distinction here.

If k is even, then the height is given by

$$H(t) = (k/2)(1 + \cos \alpha)L + (t-k)L.$$

The first term denotes a set of complete two-year cycles, where a branch grows with L and $L \cos \alpha$ in the vertical direction; the last term is the height of the last, vertical branch, which has age $t-k$.

If k is odd, then the height is given by

$$H(t) = ((k-1)/2)(1 + \cos \alpha)L + L + (t-k)L \cos \alpha.$$

Here the number of complete two-year cycles is given by the first term again, followed by term L to denote the last vertical branch, and finally the height of the last branch is given by the last term.