2IV10/2IV60 Computer Graphics

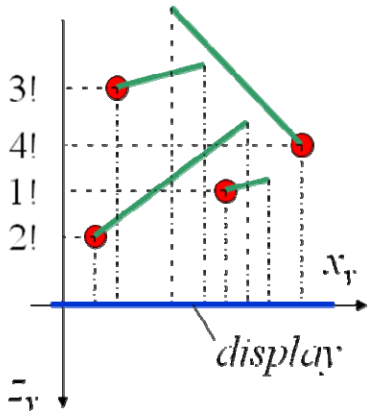
Examination, January 29 2013, 9:00 – 12:00

This examination consist of **four** questions with in total 16 subquestion. 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 from each function and procedure a short description of input and output. The use of the book, copies of slides, notes and other material is not allowed.

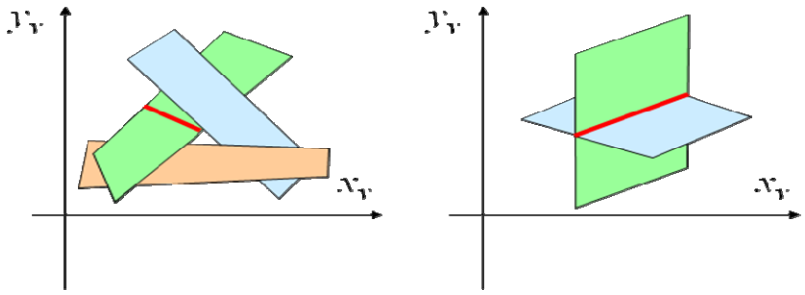
1 We consider some basic techniques for computer graphics.

a) The depth-sort or painter's algorithm seems simple, but to make it work properly requires effort. Give an example where sorting the polygons for the closest vertex does not give the correct result.

Some cases that give problems.



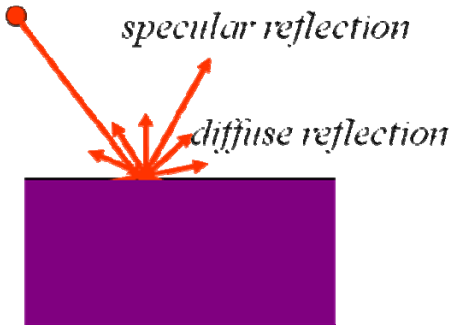
The image above shows that polygons can be in front of other polygons, while the closest vertex is farther away.



Circular overlapping polygons and intersecting polygons also give problems. Here cutting up polygons is the only remedy.

b) What is the characteristic difference between diffuse and specular reflection?

In case of diffuse reflection, incoming light is reflected uniformly in all directions, while with specular reflection the incoming light is reflected more around the direction of mirroring reflection. See the following picture:



c) Give an advantage and a disadvantage of modeling colors using the RGB-model.

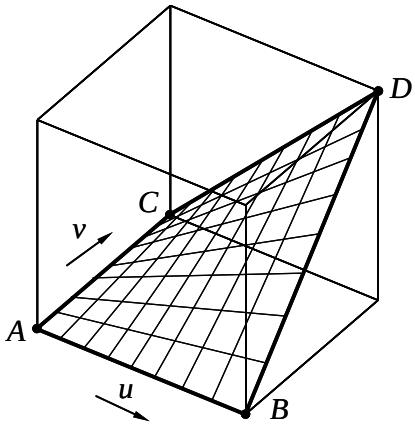
The advantage of using RGB is that it matches directly with graphics hardware, a disadvantage is that specification of colors in RGB is not intuitive.

d) Describe two optical effects that can be handled easily with ray-tracing and not with straightforward methods.

Two of: cast shadows / transparency with refraction / mirroring reflection.

---

2 We are given a quadrilateral *ABCD* (see drawing) and want to fill in the interior with a smooth surface. We use bilinear interpolation for this. On the edges *AB* and *CD* we construct points $P(u)$ and $Q(u)$ with linear interpolation, next we connect these points with straight lines.



a) Give a parametric description $S(u, v)$ of the surface for arbitrary points *A*, *B*, *C*, and *D*.
We obtain $S(u, v)$ by linear interpolation of $P(u)$ and $Q(u)$, using v as parameter:

$$S(u, v) = (1-v)\,P(u) + v\,Q(u).$$

Furthermore, $P(u)$ and $Q(u)$ are obtained by linear interpolation of the original points:

$$P(u) = (1-u)A + uB, \text{ and}$$
$$Q(u) = (1-u)C + uD.$$

Substituting this in the equation for $S(u, v)$ gives:

$$S(u, v) = (1-v)((1-u)A + uB) + v((1-u)C + uD), \text{ or, equivalently:}$$

$$S(u, v) = (1-u)(1-v)A + u(1-v)B + (1-u)vC + uvD.$$

b) Suppose that the four points are located on the vertices of a unit cube, as follows: $A=(0, 0, 0)$, $B=(1, 0, 0)$, $C=(0, 1, 0)$, and $D=(1, 1, 1)$. Give a coordinate-wise parametric description $(x(u, v), y(u, v), z(u, v))$ of the surface.
Substitution in the last formula just found gives:

$$\begin{aligned}(x(u, v), y(u, v), z(u, v)) &= (1-u)(1-v)(0,0,0) + u(1-v)(1, 0, 0) + (1-u)v(0,1,0) + uv(1,1,1)\\ &= (u-uv, 0, 0) + (0, v-uv, 0) + (uv, uv, uv)\\ &= (u, v, uv).\end{aligned}$$

Substitution of the points shows that this is the right answer.

c) Give an implicit representation $f(x, y, z) = 0$ of the surface of the previous question. Hint: eliminate u and v from your previous answer.
We just found $x = u$, $y = v$, and $z = uv$. Hence, u is equal to x, and v is equal to y. Substituting this in the equation for z gives then $z = xy$. Rewriting this in the requested form gives $f(x,y, z) = z - xy$.
.

d) Calculate a normal vector $N(u, v)$ on this surface.
One route is to use $N(x, y, z) = \nabla f = (-y, -x, 1)$. Using the result of question 2b) gives $N(u, v) = (-v, -u, 1)$.

Alternatively, we can use:

$$N(u, v) = \partial S(u, v) / \partial u \times \partial S(u, v) / \partial v$$
$$= (1, 0, v) \times (0, 1, u)$$
$$= (-v, -u, 1).$$

This is not difficult to calculate here, if you know the definition of a normal product.

A unit length normal $N'(u, v)$ is obtained by dividing by the length of $N(u, v)$:

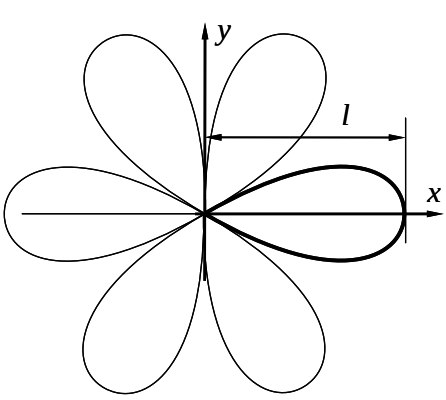
$N'(u, v) = (-v, -u, 1) / \sqrt{(u^2.+v^2 + 1)}$,

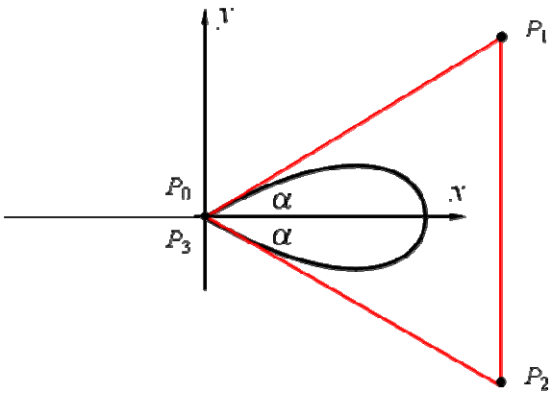
but this was not explicitly requested.

---

3 We are going to use Bézier cubic splines to draw a flower in 2D. A flower has N leaves, each leaf is mirror-symmetric, and the leaves touch each other in the origin (see figure next page). We model each leaf as a single Bézier segment. We focus on one leaf, aligned with the x-axis, indicated bold in the drawing. The curve that describes this leaf is given by

$P(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3$ with $t \in [0,1]$.





a) Give all possible positions of the four control points P_0, P_1, P_2, and P_3, for flowers of arbitrary size and arbitrary N. How many free parameters are left over?

The picture on the right shows a possible configuration of the control points. The curve has to start and end in the origin, hence $P_0 = P_3 = (0, 0)$. We use polar coordinates to specify the two other points. The two angles α should be equal to π/N, such that $2N$ angles α add up to a full rotation. Because of symmetry, all control-points should be at the same distance r from the origin. The parameter r can be freely chosen, and this is the only free parameter here. This gives for the two remaining control points:

$P_1 = (r \cos(\pi/N), r \sin(\pi/N))$ and
$P_2 = (r \cos(\pi/N), -r \sin(\pi/N))$.

For P_2 the y-coordinate is negative, indicating the reflection over the x-axis or (equivalently) the use of a negative angle. Other solutions are possible of course, if we for instance use the x-coordinate of P_1 (say a) as parameter, we get:

$P_1 = (a, a \tan(\pi/N))$ and
$P_2 = (a, -a \tan(\pi/N))$.

b) How to choose the control points such that the leaf has length l ?

This length here is the maximum distance of $P(t)$ to the origin. Looking at the picture and using symmetry, we assume that this maximum is achieved half-way the curve, i.e., for $t = 1/2$. Substitution gives for the x-coordinate $P_x(1/2)$ of the curve:

$$\begin{aligned} P_x(1/2) &= (1/8)P_{0x} + (3/8)P_{1x} + (3/8)P_{2x} + (1/8)P_{3x} \\ &= 0 + (3/8)\, r \cos(\pi/N) + (3/8)\, r \cos(\pi/N) + 0 \\ &= (3/4)\, r \cos(\pi/N)\ . \end{aligned}$$

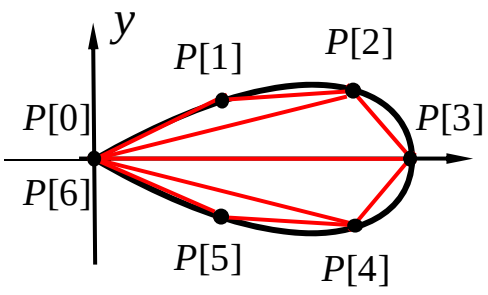
Note that just using the x-coordinate of P_1 or P_2 does not give the proper result. Next, we have to set r (our single free parameter) such that the desired length l is obtained. Setting $l = P_x(1/2)$ next leads to

$$r = 4\, l\ /\ (3 \cos(\pi/N)).$$

c) Write an algorithm *drawLeaf* to draw a filled-in version of the leaf. (See also generic remarks in the header of this examination). Assume that a function *drawTriangle* (A, B, C) is available, where A, B, and C are the vertices of a triangle in clock-wise order. Use M triangles (where M is an even number) to approximate the leaf.

Various solutions are possible. We can position the triangles in a fan-like way, where all triangles have one vertex in the origin, and the other two vertices are succeeding points at the curve using a stepsize of $1/(M+2)$ in t. For compactness, we define a small function to compute these points and to store them in an array:

```
getPoints;
begin
    for i = 0 to M + 2  do P[i] = P(i  /  (M+2) );
end;
```



$M = 4$, fan solution

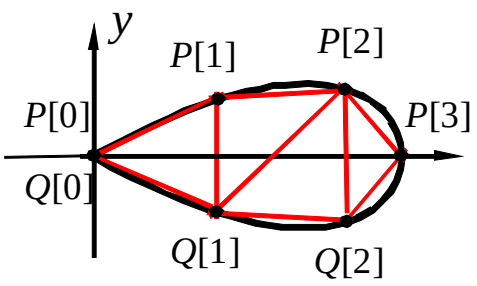
such that the leaf can be drawn with:

```
drawLeaf;
begin
  getPoints;
  for i = 1 to M do drawTriangle (P[0], P[i], P[i+1]);
end;
```

Alternatively, we can use two triangles for the start and end of the leaf, and a zig-zag pattern for the area in between. We define a little function to simplify finding points opposite to a given point:

```
function Q[i]: point; // point opposite to a point P[i], reflection over x-axis
begin
    Q[i] = P[M+2 – i ];
end;

drawLeaf;
begin
  getPoints;
  drawTriangle(P[0], P[1], Q[1]);  // triangle on the left
```



$M = 4$, zig-zag solution

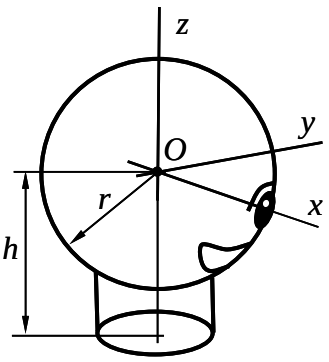
```
  for i = 1 to M/2 – 1 do          // zig-zag pattern
  begin
    drawTriangle(P[i], P[i+1], Q[i]);
    drawTriangle(P[i+1], Q[i+1], Q[i]);
  end;
  drawTriangle(P[M / 2 ], P[M / 2 + 1], Q[M / 2]); // triangle on the right
end;
```

d) Suppose that we want to use this approach to draw flower leaves in 3D. We want the leaves in the origin to be tangent to the $z = 0$ plane, and to bend them up with increasing distance from the origin, to get a tulip-like shape. Is this possible with the single segment, cubic Bézier spline approach used here? Explain your answer.

No, this is not possible. The control-polygon is a triangle, because the begin- and end-point coincide, and this does not allow for non-planar shapes.
Alternatively: suppose that the points P_1 and P_2 have a non-zero z-component. This implies that the tangent vectors at the start and end of the curve also point upward, hence, the leave is not tangent to the plane anymore.

---

4 We are developing a first person action game, where we see everything through the eye of our hero. Our hero is Freddy the Cyclops, a one-eyed giant. We model Freddy's head in a local coordinate system as a sphere with radius r, centered around the origin, his eye is fixed on the sphere on the local x-axis (see figure). The current position and orientation of his head is fixed by a 4×4 homogenous transformation matrix M, such that a position A in global world coordinates is related to a position B in head coordinates via $A=MB$. The matrix M describes a rigid transformation, no scaling or skewing is applied. In the following, matrices do not have to be specified element-wise. It may be assumed that $T(V)$ gives a translation matrix along the vector V, and that $R_X(a)$, $R_Y(a)$, and $R_Z(a)$ give a rotation matrix of a degrees around the x-, y-, and z-axis, respectively.



a) We use a virtual camera model to specify the viewing projection. This model requires a viewpoint P (origin of the camera), a direction V (direction in which the camera points), and an up-vector W (vector that has to be shown vertically), all in world-coordinates. Give formula's to calculate P, V, and W according to the specification of Freddy's head. Hint: what are P, V, and W in local coordinates?

In local coordinates, the eye is located at $(0, 0, r)$; the view direction is aligned with the x-axis, hence $(1, 0, 0)$, and the up-vector with the local z-axis $(0, 0, 1)$. To get these in global coordinates, we add a homogenous coordinate (0 for vectors, 1 for points) and transform with M. This gives:

$P = M\,(r, 0, 0, 1)^T$; $V = M\,(1, 0, 0, 0)^T$; and $W = M\,(0, 0, 1, 0)^T$, where the T-superscript denotes the transpose.

b) Suppose that Freddy is walking, and that the current transformation is given by $M(t)$, with t the time in seconds. We want to make the motion more lively. Assume Freddy is continuously looking around, looking left and right by rotating his head around the local z-axis over angles of maximally d degrees, where one complete cycle takes T seconds. Give an adapted version $M'(t)$, given $M(t)$.

To model the periodic rotation of the head, we use a sine. The current angle $a(t)$ can then be described as $a(t) = d \sin (360\ t / T)$. If we use radians as argument for the sine function, we get $a(t) = d \sin (2\pi t / T)$. The requested rotation can be obtained by rotation over this angle around the local z-axis, hence we use post-multiplication of the given transform $M(t)$. This leads to

$$M'(t) = M(t)\ R_Z(a(t)) = M(t)\ R_Z((d \sin (2\pi t / T)) .$$

c) Freddy is standing still. But suddenly, he gets a blow to the center of his head from his left side. As a result, his head rotates over b degrees around a line. This line is parallel with the local x-axis and goes through a point located at a distance h below the center of his head. Give an adapted version M', given M.

We get the requested rotation by performing the following transformations in local coordinates:

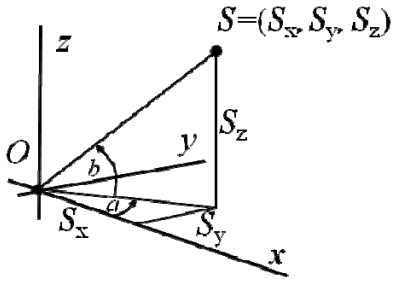
1) move the head down with a translation $T((0, 0, -h))$;
2) rotate the head counterclockwise around the x-axis with $R_x(b)$.
3) move the origin back, translating along the rotated z-axis with a translation $T((0, 0, h))$.

Combined, this gives the transformation $T((0, 0, -h))\ R_x(b)\ T((0, 0, h))$. The adapted total transformation is then:

$$M' = M\ T((0, 0, -h))\ R_x(b)\ T((0, 0, h)).$$

d) Freddy senses that his attacker is located at a position Q in world coordinates. He rotates his head around his local z- and y-axis to focus on him. Again, give an adapted version M', given M. For the calculation of angles, assume a function arctan(y, x) is available, which returns the angle in degrees between the x-axis and a line from the origin to a point (x, y). Hint: Assume that S is the position of Q in local coordinates. How to calculate S?

We want to apply rotations in local coordinates. For this we use the point S, which is Q in local coordinates. We know that $Q = MS$, we can use the inverse transformation to get S for a given Q. Or, more formally, we multiply the left and right side of the equation with M^{-1} to get $S = M^{-1}Q$.



The rotation of the head now has to be such that the transformed x-axis points towards the point S. We do this by rotating around an angle a around the z-axis, followed by a rotation b around the local y-axis. The rotation a can be derived by observing that $\tan(a) = S_y / S_x$. The rotation b can be found by using $\tan(b) = S_z / (S_x^2 + S_y^2)^{1/2}$. Combining this gives:

$$M' = M\ R_z(\arctan(S_y, S_x))\ R_y(\arctan(S_z, (S_x^2 + S_y^2)^{1/2})).$$