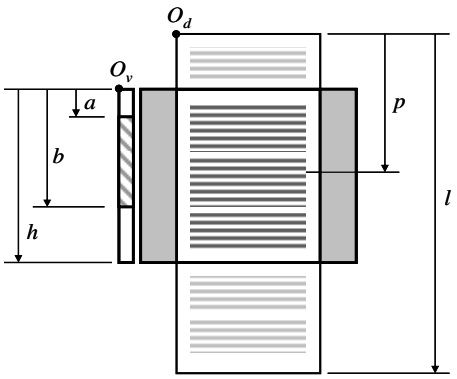
Example Examination 2IV60 – 110412

(translated version 2IV10 11 april 2012, 14:00-17:00)

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 a procedure 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 the display of a document in a viewport. The document has length l cm, the origin O_d is at the upper left corner. It is desired that a line at a distance of p cm from the start of the document is shown vertically centred in the viewport. The viewport has a height of h pixels, the origin O_v is at the upper left corner. The user can set a scale factor s, using as convention that with s=100% the document is shown vertically filled, using larger values the document is shown larger.



a) Give a transformation $y' = f(y)$ for converting a vertical document coordinate y to a pixel coordinate y'.

This is a linear transformation of the form $y' = Ay + B$. We first fix the scale factor A. We normalize the length of the document by division by l, scale it to the height in pixels by multiplication by h, and finally we take the user defined scale factor into account by multiplying by $s/100$. This gives:

$A = sh / (100\ l)$.

Furthermore, we know that for $y = p$ we should get $y'=h/2$. Substitution gives:

$h/2 = shp / (100\ l) + B$, hence

$B = h/2 - shp / (100\ l)$.

The complete result is hence $y' = shy / (100\ l) + h/2 - shp / (100\ l)$. Or, more compact:

$y' = sh(y-p) / (100\ l) + h/2$.

This can also be done using transformation matrices, but because this is one-dimensional, a direct approach works just as well.

b) Next to the viewport a scrollbar is shown. The diagonally hatched rectangle represents the part of the document that is shown. Give a and b in pixel coordinates.

We focus on the vertical center of the rectangle $C=(a+b)/2$ and the height of the rectangle $H=b-a$, and derive a and b from these, because C and H can directly be related to the various variables. To get C, we observe that the fraction p/l represents the relative offset in the document and pixel coordinates can be obtained by scaling with h. Hence, $C = hp/l$.

If the scale factor s is 100, the height of the rectangle should be h; if the scale factor is doubled, the height of the rectangle should be halved, hence H is inversely proportional to $s/100$. Hence for H we get $H = 100h/s$. We furthermore know that $a = C - H/2$ and $b = C + H/2$. Substitution gives:

$a = hp/l - 50h/s$ and $b = hp/l + 50h/s$

c) Suppose that a user moves the hatched rectangle of the scrollbar c pixels downwards with the mouse. What is the new value of p?

We have derived for the center C of the rectangle that $C = hp/l$. In the new situation, $C+c=hp'/l$, where p' is the new value of p. Rewriting the equation for the new situation gives

$C+c = h'/l$, or

$hp/l + c = hp'/l$, or

$p' = p + cl/h$.

It is always a good idea to check the result, to see if it makes sense. The new value can be considered as an increment to the original value of p; the larger the move c, the larger the increment; the increment has the same direction as the move; if the document is large (l large) or if the viewport is small (h small), the effect of a move is strong. Furthermore, the scale factor s has no influence here. This all seems very plausible.

d) For tablets and mobile phones typically the document itself is dragged, by selecting a position and dragging this. Suppose that a user drags the document d pixels down in the viewport. What is now the new value of p?

We have already derived that the base transformation is given by $y' = sh(y-p) / (100\ l) + h/2$. In the new situation, the transformation must have the form

$Y' = sh(y-p') / (100\ l) + h/2$,

where p' is the new value of p, and where Y' represents the new pixel coordinates. Furthermore, it should hold that $Y'=y'+d$: all old pixel coordinates are incremented with d. Substitution gives:

$sh(y-p') / (100\ l) + h/2 = sh(y-p) / (100\ l) + h/2 + d$, or

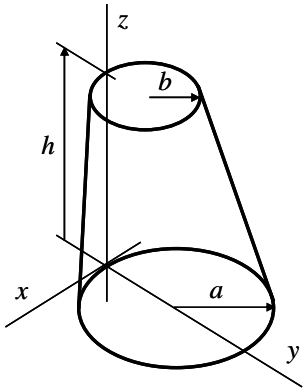
$-shp' / (100\ l) = -shp / (100\ l) + d$, or

$p' = p - 100\ dl / (sh)$

Compared to the scrollbar version, we see that the sign has changed: moving your finger down implies moving up in the document. Also, we see that the scale factor now plays a role: the more zoomed out (the smaller the value of s), the stronger the effect of dragging on the position in the document.

---

2 We consider a part to connect two tubes. The lower and upper edges are circles, with radius a and b, the height is h. Both edges touch the z-axis, are parallel with the xoy-plane and are symmetric with respect to the y-axis. The surface of the part consists of straight lines that connect both edges.



a) Give parametric descriptions $R(u)$ en $S(u)$ of the lower and upper edge. Indicate the domain of u.

Both edges are circles. We use the standard pattern $P(u) = C + r \cos u\ A + r \sin u\ B$ for a circle in space, with centre C, radius r, and two orthogonal unit length vectors A and B in the plane of the circle. For the case given, we get:

$R(u) = (0, a, 0) + a \cos u\ (1, 0, 0) + a \sin u\ (0, 1, 0)$
$= (a \cos u,\ a + a \sin u,\ 0)$
and
$S(u) = (0, b, h) + b \cos u\ (1, 0, 0) + b \sin u\ (0, 1, 0)$
$= (b \cos u,\ b + b \sin u,\ h)$.
In both cases $u \in [0, 2\pi)$.

b) Give a parametric description $P(u, v)$ of the surface.

We get a parametric description of the surface by connecting lower and upper edges by straight lines, and using the v parameter to select a point on the line. In other words,

$P(u, v) = (1-v)R(u) + v\,S(u)$, with $u \in [0, 2\pi)$ and $v \in [0, 1]$.

We can subsitute the results of the previous question, but that does not give a more readable or insightful expression.

c) Give a procedure to determine if a point Q is inside or outside the part (assuming lower and upper sides are closed).

Obviously, if the point Q is above the upper side or below the lower side, it is outside the part. To test for the area in between, we model the curved surface as an implicit surface of the form $f(x, y, z) = 0$, as this allows for easy testing. We observe that cross-sections of the surface with horizontal planes are all circles with a radius $r(z)$ that depends on the distance, and which centers are located at $(0, r(z), z)$. Hence, the function f is given by $f(x, y, z) = x^2 + (y - r(z))^2 - r^2(z) = 0$. If $f(x, y, z) > 0$, then the point (x, y, z) is outside the part. The radius r depends linearly on z as $r = (1-u)a + ub$, with $u = z/h$. Described as a procedure, this gives:

```
function CheckInside(x, y, z): Boolean; // returns true if the point (x, y, z) is inside, else false
{
        if z <= 0 or z >= h then return false;
        r = (1–z / h)*a + (z / h)*b;
        return x*x + (y – r)* (y – r) – r*r  <  0
}
```

d) Given a procedure DrawQuad(A, B, C, D) for drawing a quad with the succeeding vertices A, B, C en D, give a procedure for drawing the part.

There was no request that the wall must be subdivided, so a single loop along the circles suffices here.

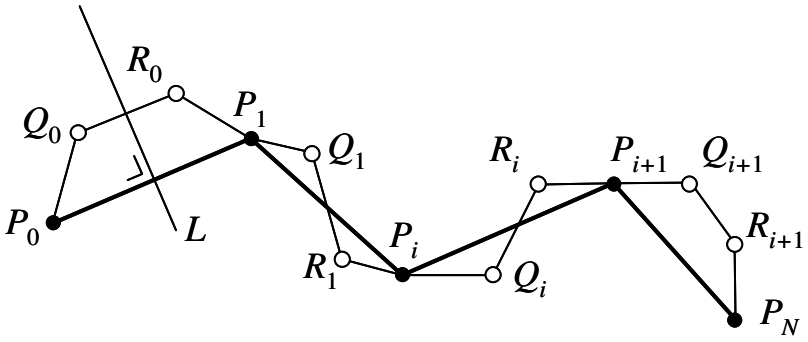
```
procedure DrawPart(a, b, h, N: integer); // Draw a part, using N steps for the circles

        function R(u): Point; { return (a cos u, a + a sin u, 0) };  // get point at bottom
        function S(u): Point; { return (b cos u, b + b sin u, h) };  // get point at top
{
        du = 2*PI/N;  //  increment

        for i = 0 to N–1 do // loop over facets
        {
                u = i*du;           // get current angle
                A = R(u);           // get points of one quad, connecting two points at
                B = R(u + du);      // the bottom (A and B) and two at the top (C and D).
                C = S(u + du);
                D = S(u);
                DrawQuad(A, B, C, D);
        }
}
```

Many opportunities exist for making this more efficient and less readable.

3 We are going to use Bézier cubic splines to interpolate a given sequence of points P_i, with $i = 0,\ldots,N$. To this end we define additional points Q_i and R_i, with $i = 0,\ldots, N-1$, such that a sequence of segments $S_i(u) = (1-t)^3 P_i + 3t(1-t)^2 Q_i + 3t^2(1-t)R_i + t^3 P_{i+1}$ with $t \in [0,1]$ together produce a curve that interpolates the given points.



a) It is required that the curve is smooth for interior points P_i, with $i = 1,\ldots, N-1$, and that the tangent line to the curve at these points P_i is parallel to a line through the points P_{i-1} en P_{i+1}. Give all possible positions of the points Q_i and R_i, with $i = 0,\ldots, N-1$. How many free parameters are left over?

The tangent line of the curve at an interior point P_i is given by $P(t) = P_i + V_i t$, with $V_i = P_{i-1} - P_{i-1}$. The points R_{i-1} and Q_i should be located at this line, with P_i in between. We can achieve this by setting $R_{i-1} = P_i - u_i V_i$ with $u_i > 0$, and $Q_i = P_i + v_i V_i$, with $v_i > 0$. Each of those points has one free parameter, so in total we have $2(N-1)$ free parameters for the interior points, and in total four free parameters for the two free points Q_0 and R_{N-1}.

b) Give a simple recipe for setting the free parameters, such that when the points P_i, met $i = 0,\ldots, N$, are located in order at equal distances on a straight line, this also holds for the sequence of points $R_0, P_1, Q_1, R_1, \ldots, P_{N-1}, Q_{N-1}$.

Suppose that all points are located at unit distances along the x-axis, i.e., suppose that the points P_i are located at positions $(3i, 0)$, the points Q_i are located at positions $(1 + 3i, 0)$ and the points R_i are located at positions $(2 + 3i, 0)$. The length of the vectors V_i is then 6. Selecting $u_i = v_i = 1/6$ gives the desired positions.

c) If we further require that the tangent line at point P_0 is parallel to a line through P_0 and P_1, what are the consequences for the points Q_0 and R_0? Show that this requirement does not give a nice result.

The point Q_0 has to be located on the half line starting from P_0 in the direction to P_1, in other words, $Q_0 = P_0 + v_0 (P_1 - P_0)$. The point R_0 does not influence the tangent line at the start, so this requirement has no consequences for this point. The result is strange: the curve heads of in the direction of P_1, but then makes a turn in one direction followed by a turn to the other direction to make the smooth transition at the next point.

d) Give a recipe for positioning point Q_0 for given P_0, R_0 en P_1, such that the segment $S_0(u)$ is mirror symmetric with respect to a line L (see figure). The line L passes through a point half way the line through P_0 and P_1, and is perpendicular to that line.

This can be solved in various ways. The quickest is the following. To reach the point Q_0, start at P_0 and add the vector $V = R_0 - P_1$. Next, make two steps in the opposite direction of a vector W, which is the projection of V on the vector $U = P_0 - P_1$. We use the properties of dot products, and then we get $W = U\,(V \cdot U / U \cdot U)$. And finally, $Q_0 = P_0 + R_0 - P_1 - 2W$.

---

4 We consider some basic techniques for computer graphics.

a) Give for each of the following scenes which hidden surface algoritme you would use and why: 1) scene with mirroring and transparent balls; 2) scene consisting of a single convex object; 3) scene with meshes of thousands of triangles.

1) ray-tracing is the method of choice: it can easily produce complex optical effects, and spheres can easily dealt with;
2) here simple back-face removal suffices: the only non-visible surfaces are those at the back.
3) The z-buffer or depth buffer algorithm can easily deal with large numbers of primitives, and is directly supported by OpenGL and graphics hardware.

b) Describe a simple algorithm to determine if a point is inside or outside a polygon.

Construct a half-line starting at the point and count the number of intersections with the boundary of the polygon. If this number is odd, the point is inside, otherwise it is outside. If a peculiar situation occurs (line through vertex, for instance), try a different line.

c) Give the model for the perceived color by illumination by a single light source at infinity for a diffuse reflecting surface. Give a sketch of the geometry, give the formula(s), and describe variables used. Does the viewing direction play a role?

Suppose that N is a unit outward normal on the surface, and that L is a unit vector pointing to the light source. Furthermore, the intensity of the light source is (I_R, I_G, I_B) (specified for red, green, and blue), and the reflectance of the surface is given by (k_R, k_G, k_B). Using Lambert's law, we get then for the perceived color

$$(C_R, C_G, C_B) = \max(0, N \cdot L) \, (k_R I_R, k_G I_G, k_B I_B).$$

The reflection is proportional to the cosine of the angle between N and L, which can be conveniently computed with a dot product. If this is negative, the surface is not illuminated, and hence the light does not contribute.
The viewing direction does not play a role, although… Here I assumed that the scene consists of a closed surface which is looked at from the outside. For a generic surface, the normal should be chosen such that it points towards the viewer, and then the viewing direction does play a role.

d) In **1-D** texture mapping an additional parameter value t is specified per vertex. How is this additional information used? Give an application.

The parameter t is used to do a look-up in a 1-dimensional color table. The color found is next used for coloring the surface. A typical application area is visualization, for instance to show temperature distributions using a rainbow color scale.
