Graphics (INFOGR 2012-2013): Final Exam (T2)

Thursday, July 4, 2013, 17:00, EDUC-GAMMA (time for the exam: max. 2 hours)

COMMENTS AND SOLUTIONS

No responsibility is taken for the correctness of the provided information.

Note: the provided solution might not be the only option,
but for some problems others could exist that are correct as well.

Problem 1: Perspective projection

■ **Subproblem 1.1 [6 pts]: Windowing (or viewport) transformation.** Note that for all questions in this subproblem, a short answer, e.g., one phrase or sentence can be sufficient to get full credit.
The following matrix M_{vp} is used in the graphics pipeline to map the orthographic projections in the canonical view volume, i.e. values in the square $[-1,1]^2$, to the actual $n_x \times n_y$-sized screen window:

$$M_{vp} = \begin{pmatrix} \frac{n_x}{2} & 0 & 0 & \frac{n_x}{2} - \frac{1}{2} \\ 0 & \frac{n_y}{2} & 0 & \frac{n_y}{2} - \frac{1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

1. Why do we need to subtract $\frac{1}{2}$ in the last column when we do this mapping?

   ■ **Solution/comments:** Because we want to map to the pixel center

2. Although we map from 2D to 2D, this matrix contains a 3rd row and column for the $z-$value. Why? (Hint: in the lecture, we had two reasons; it is sufficient to list just one of them here.)

   ■ **Solution/comments:** Possible reasons:

   - to carry along the $z-$value for depth calculations / hidden surface removal
   - to combine the matrix with the other matrices used for perspective projection, and these are dealing with 3D data (so we just add a row and column that changes neither the $z-$value nor any other values).

3. Although we map from 2D to 2D, this matrix contains a 4th row and column. How are the values in the last row called, and why do we need them here?

   ■ **Solution/comments:** They are the *homogeneous coordinates*, and we need them because our transformation contains a *translation* (i.e. it's an affine transformation).

■ **Subproblem 1.2 [6 pts]: Multiple choice question.** Mark the correct answer. No explanation required.
There is only one correct answer for each individual question.
Assume four points $\vec{p}_0 = (x_0, y_0, z_0)$, $\vec{p}_1 = (x_1, y_1, z_1)$, $\vec{p}_2 = (x_2, y_2, z_2)$, and $\vec{p}_3 = (x_3, y_3, z_3)$ that are all on a line that goes through our virtual camera. $\vec{p}_0$ is on the near plane n of the view frustum, $\vec{p}_3$ is on its far plane f. $\vec{p}_1$ and $\vec{p}_2$ are both inside the view frustum, and $\vec{p}_1$ is closer to the virtual camera than $\vec{p}_2$. P is the matrix that maps a point $\vec{p}_i = (x_i, y_i, z_i)$ in the view frustum to a corresponding point $\vec{p}_{s_i} = (x_{s_i}, y_{s_i}, z_{s_i})$ in the orthographic view volume.

1. After applying P to $\vec{p}_0, \ldots \vec{p}_3$ the resulting $x-$value x_{s_0} of $\vec{p}_{s_0}$ is ...

   A. x_0 B. n C. nx_0 D. x_0/z_0 E. neither of these

   ■ **Solution/comments:** Points on the near plane remain unchanged. Hence, the correct answer is A.

2. After applying P to $\vec{p}_0, \ldots \vec{p}_3$ the resulting $z-$value z_{s_3} of $\vec{p}_{s_3}$ is ...

   A. unchanged B. $> z_2$ and $< z_3$ C. $> z_3$ D. neither of these

   ■ **Solution/comments:** Points on the far plane move to a different position on the far plane.
   Hence, their $z-$coordinate value remains the same, and the correct answer is A.

3. After applying P to $\vec{p}_0, \ldots \vec{p}_3$ the resulting $z-$value z_{s_2} of $\vec{p}_{s_2}$ is ...

   A. $> z_{s_1}$ and $> z_{s_3}$ B. $> z_{s_1}$ and $< z_{s_3}$ C. $< z_{s_1}$ and $< z_{s_3}$ D. $> z_{s_1}$ and $> z_{s_3}$ E. neither of these

   ■ **Solution/comments:** The order along the line towards the virtual camera (and thus the order of the transformed $z-$coordinates) is preserved. Hence, the correct answer is B.

Make a mark here if you answered parts of these questions on an extra sheet: [ ]

Problem 2: Clipping

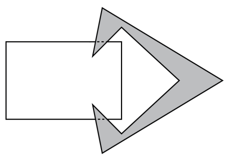
■ **Subproblem 2.1 [4 pts]: Position of clipping in the pipeline.** Complete the following text in a way that creates a correct statement: Clipping at the beginning of the graphics pipeline forces us to deal with rather complex plane equations. Clipping at the end of it can lead to incorrect results. To avoid both issues, . . .

. . . the best position to apply clipping is after the perspective transformation (alternative correct answers: “after the transformation to the orthographic view volume”, “after multiplication with the transformation matrix”, etc.)

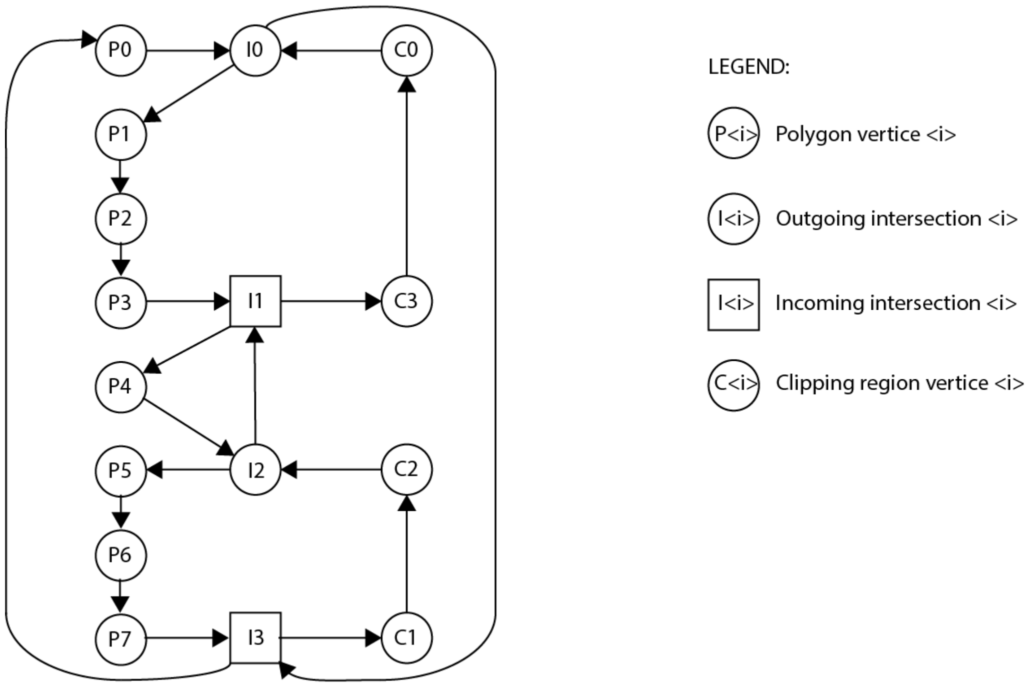
. . . and right before the homogeneous divide (alternative answer: the perspective divide)

■ **Subproblem 2.2 [2 pts]: Sutherland-Hodgman algorithm.** When being applied for clipping arbitrary polygons, the *Sutherland-Hodgman algorithm* can result in *degenerated polygons*. Assume the rectangular clipping region illustrated below. Draw an arbitrary polygon that partly intersects with it and would *not* be clipped correctly by the *Sutherland-Hodgman algorithm*.

■ **Solution/comments:** Any polygon that intersects with the clipping region at at least two different positions would be correct. For example, this one (which btw. would create the graph from the next subproblem when using the Weiler-Atherton algorithm):



■ **Subproblem 2.3 [6 pts]: Weiler-Atherton algorithm.** Assume the graph depicted below was created using the *Weiler-Atherton algorithm* for clipping arbitrary polygons. Write down the resulting clipped polygons that can be extracted from this graph. Note: Make sure to list the vertices of these polygons in the correct order returned by the algorithm, so we can see if you understood how it works.


P0
I0
C0
P1
P2
P3
I1
C3
P4
P5
I2
C2
P6
P7
I3
C1
LEGEND:
P<i> Polygon vertice <i>
I<i> Outgoing intersection <i>
I<i> Incoming intersection <i>
C<i> Clipping region vertice <i>

■ **Solution/comments:** $I0 \rightarrow I3 \rightarrow P0 \rightarrow I0$ and $I2 \rightarrow I1 \rightarrow P4 \rightarrow I2$.

Make a mark here if you answered parts of these questions on an extra sheet: [ ]

Problem 3: Culling

■ **Subproblem 3.1 [4 pts]: Culling techniques.** Complete the following sentences in a way that creates a correct statement by naming the correct technique for removing triangles with the respective characteristic:

1. Triangles outside of the view frustum are removed by a technique called . . .

   frustum (or volume) culling.

2. Triangles within the view frustum that are occluded by others are removed by a technique called . . .

   occlusion culling.

■ **Subproblem 3.2 [6 pts]: Backface culling.** Assume we have two triangles in 3D that define the following planes:

$$f_1(\vec{p}) = \begin{pmatrix}1\\2\\1\end{pmatrix}\left(\vec{p} - \begin{pmatrix}2\\2\\2\end{pmatrix}\right) \quad \text{and} \quad f_2(\vec{p}) = \begin{pmatrix}2\\2\\1\end{pmatrix}\left(\vec{p} - \begin{pmatrix}0\\0\\1\end{pmatrix}\right)$$

Now we want to place our camera at the position $\vec{e} = (1,1,1)$ and check if we have to draw them or if we can apply *backface culling*. For which of the two triangles (if any) will be removed with backface culling? Justify your answer.

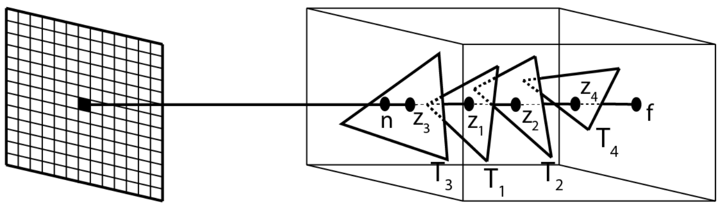
■ **Solution/comments:** We have

$$f_1(\vec{e}) = \begin{pmatrix}1\\2\\1\end{pmatrix}\cdot\begin{pmatrix}-1\\-1\\-1\end{pmatrix} = -1-2-1 = -4 < 0 \qquad \text{and} \qquad f_2(\vec{e}) = \begin{pmatrix}2\\2\\1\end{pmatrix}\cdot\begin{pmatrix}1\\1\\0\end{pmatrix} = 2+2+0 = 4 > 0$$

Hence, the triangle related to f_1 will be removed by backface culling.

Problem 4: Hidden surface removal

■ **[3 pts]: z-Buffer.** Assume the following (simplified) case that illustrates four points from four triangles in 3D that are all mapped to the same pixel position on the 2D screen. What are the values in the *z-Buffer* at the indicated position if we draw the triangles in the order T_1, T_2, T_3, and T_4?



After initialization the value in the z-Buffer is $> f$ (any value larger than f is correct) after drawing T_1 it is z_1

after drawing T_2 it is z_1 after drawing T_3 it is z_3 and after drawing T_4 it is z_3

Make a mark here if you answered parts of these questions on an extra sheet: [ ]

Problem 5: Rasterization

■ **Subproblem 5.1 [3 pts]: Multiple choice question.** No explanation required. There is only one correct answer.

In the lecture, we used the following definition to specify the bounding box of a polygon in 2D that is defined by n vertices (x_i, y_i) with $i = 1, \dots n$:

$$\vec{b_0} = (\min\{x_i\}, \min\{y_i\}) \quad \text{and} \quad \vec{b_1} = (\max\{x_i\}, \max\{y_i\}) \quad \text{for} \quad i = 1, \dots n.$$

Yet, other options exist. Which of the following ones also specify a correct bounding box of such a polygon?

(i) $\vec{b_0} = (\min\{x_i\}, \max\{y_i\})$ and $\vec{b_1} = (\max\{x_i\}, \min\{y_i\})$ for $i = 1, \dots n$
(ii) $\vec{b_0} = (\max\{x_i\}, \max\{y_i\})$ and $\vec{b_1} = (\min\{x_i\}, \min\{y_i\})$ for $i = 1, \dots n$
(iii) $\vec{b_0} = (\max\{x_i\}, \min\{y_i\})$ and $\vec{b_1} = (\min\{x_i\}, \max\{y_i\})$ for $i = 1, \dots n$

A. none B. only (i) C. only (ii) D. only (iii) E. only (i),(ii) F. only (i),(iii) G. only (ii),(iii) H. all

■ **Solution/comments:** The correct answer is *H. all*.

■ **Subproblem 5.2 [2 pts]: Scanline algorithm.** Assume the edge of a 2D triangle that defines the following line given in slope-intercept form:

$$y = \frac{7}{2}x - \frac{5}{2}.$$

Now we want to rasterize this triangle using the scanline algorithm. Calculate the value Δ_x that specifies the increase of the $x-$coordinate along the line when we move from one scanline position to the next.

■ **Solution/comments:** The line has a slope of $\frac{7}{2}$.

In general, the slope of a line is defined by the change in the $y-$coordinate value devided by the related change in the $x-$coordinate value when moving along the line, i.e. by $\frac{\Delta_y}{\Delta_x}$.

When moving from one scanline to the next, the Δ_y changes by $+1$. Hence, we have

$$\frac{\Delta_y}{\Delta_x} = \frac{1}{\Delta_x} = \frac{7}{2}$$

which leads to a Δ_x value of $\frac{2}{7}$.

■ **Subproblem 5.3 [9 pts]: Edge Table and Active Edge Table.** When implementing the scanline algorithm for rasterizing polygons, we commonly use the two data structures *Edge Table* and *Active Edge Table*.

1. Assume the following entry in an *Edge Table*: $2 : (3, 8, \frac{1}{2})$.
   What is the start point and what is the end point of the line segment represented by this entry?

   ■ **Solution/comments:**

   The start point is $(3, 2)$

   The end point is $(6, 8)$

   Explanation (not required):

   The start point follows from how the edge entry is defined: the 1st value, i.e. the scanline index (here 2), indicates the $y-$value of the start point. The 2nd value (here 3) indicates the $x-$value of the start point.

   The 3rd value (here 8) indicates the $y-$value of the line segment's end point. The last one (here $\frac{1}{2}$) indicates the Δ_x that allows us to calculate the $x-$value of the segment's end point:

$$x = 3 + (8 - 2) * \frac{1}{2}$$

(start value of x plus “steps the scanline makes from the start to the end point” times “increase of the $x-$value in each scanline step”).

2. Assume the following entry in an *Active Edge Table*: $6 : (3, 10, \frac{1}{4})$.
   Further assume that we started scanning this edge at scanline number 2.

   What is the start point and what is the end point of the line segment represented by this entry?

   ■ **Solution/comments:**

   The start point is $(2, 2)$

   The end point is $(4, 10)$

   Explanation (not required):

   Again, the 3rd value in the edge entry indicates the $y-$value of the line segment's end point (here 10). To reach that from the current position (here 6), we must make 4 steps (10-6). In these 4 steps, our current $x-$value (here 3) increases by $\Delta_x = \frac{1}{4}$ four times, i.e.

$$x_{end} = 3 + 4 * \frac{1}{4} = 4$$

   If we start scanning the segment at scanline number 2, the $y-$value of the starting point is obviously 2. Because we made 4 steps to get from that position (scanline = 2) to the current one (scanline = 6), the $x-$value increased 4 times by $\Delta_x = \frac{1}{4}$. Hence, the $x-$value of the start point is

$$x_{start} = 3 - 4 * \frac{1}{4} = 2$$

Problem 6: Shading

■ **Subproblem 6.1 [4 pts]: Blinn-Phong shading.** Instead of a reflection vector as in ordinary Phong shading, the Blinn-Phong model uses a *halfway vector* $\vec{h}$ to calculate glossy reflection.

1. How is this halfway vector defined? A short verbal description (1 sentence or phrase) can be sufficient to get full credit; no formulas need to be given here.

   ■ **Solution/comments:** the vector that lies "halfway" between the eye vector $\vec{e}$ and the light vector l

2. What is the major advantage of using the halfway vector instead of the reflection vector? A short verbal description (1 sentence or phrase) can be sufficient to get full credit; no formulas need to be given here.

   ■ **Solution/comments:** The scalar product of the reflection vector and the eye vector can get negative even if they are both on the same side of a surface. This cannot happen with the scalar product of the halfway vector and the normal vector that we use in the Blinn-Phong shading.

■ **Subproblem 6.2 [6 pts]: Radiosity (multiple choice questions).** Mark the correct answer. No explanation required. There is only one correct answer for each individual question.

1. Which of the following characteristics is *not* modelled via form factors?
   - A. the distance r between two patches A_i and A_j
   - B. the relative orientation of two patches A_i and A_j towards each other
   - C. the reflectivity ρ_i of the patch A_i
   - D. the shape of the two patches A_i and A_j

   ■ **Solution/comments:** The correct answer is
   C. the reflectivity ρ_i of the patch A_i.

2. The number of form factors is . . .
   - A. . . . linear in the number of patches,
     because we can calculate them with a linear equation system.
   - B. . . . quadratic in the number of patches,
     because to calculate the form factor for one patch, we have to consider all other patches.
   - C. . . . cubic in the number of patches,
     because the Nusselt Analog says that we can model them via a hemisphere.
   - D. . . . to the power of four in the number of patches,
     because they are symmetric, so we have to calculate them twice.

   ■ **Solution/comments:** The correct answer is
   B. quadratic, because to calculate each patch, we have to consider all other patches.

3. Progressive refinement is . . .
   - A. . . . a method to iteratively approximate the radiosity B_i.
   - B. . . . a method that approximates form factors via a hemisphere
   - C. . . . a method that approximates form factors via a hemicube.
   - D. . . . a method to create an adaptive subdivision of a scene into patches.
   - E. . . . none of the above.

   ■ **Solution/comments:** The correct answer is
   A. a method to iteratively approximate the radiosity B_i.

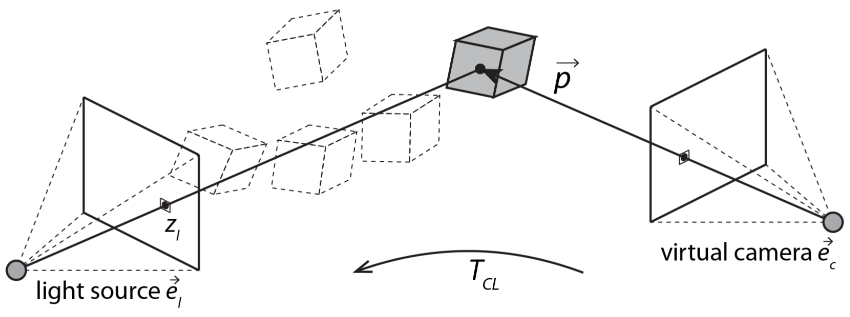
Make a mark here if you answered parts of these questions on an extra sheet: [ ]

Problem 7: Shadows

■ **Subproblem 7.1 [6 pts]: Shadow maps.** Assume the situation illustrated below with a scene represented in camera coordinates with respect to a virtual camera $\vec{e}_c$ and a light source at position $\vec{e}_l$ with a related depth buffer. The transformation from camera coordinates to "light coordinates" can be done via a transformation matrix

$$T_{CL} = \begin{pmatrix} 1 & 0 & 3 & 3 \\ 2 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Notice that it is not clear from the image if objects are blocking the light between $\vec{e}_l$ and the gray cube or not. In both cases, we are looking in the positive $z-$direction.
Consider the point $\vec{p} = (1,2,1)$ in camera coordinates. For this point, the corresponding value in the depth buffer of the light source is $z_l = 2$. Do we draw $\vec{p}$ in the light or in the shadow? Illustrate how you got your answer.



■ **Solution/comments:** When we multiply the point $\vec{p}$ with the transformation matrix T_{CL}, i.e.

$$T_{CL} \cdot (1,2,2,1) = (\cdot,\cdot,3,\cdot)$$

we see that in "light coordinates", the value for z_c is 3, which is larger than the value $z_l = 2$ in the corresponding depth buffer entry. Hence, there is at least one object blocking the light, so we have to draw a shadow at $\vec{p}$.

■ **Subproblem 7.2 [4 pts]: Multiple choice questions.** Mark the correct answer. No explanation required. There is only one correct answer for each individual question.

1. Which of the following problems with shadow volumes is solved by using a depth-fail approach instead of a depth-pass approach?
   - A. Precision problems at the clipping planes.
   - B. Divisions by zero for shadow volumes parallel to the coordinate axes.
   - C. Miscalculations in the counter because the view point lies inside of a shadow volume
   - D. Double blending in case of overlapping shadow volumes
   - E. All of the above
   - F. None of the above

   ■ **Solution/comments:** The correct answer is
   C. Miscalculations in the counter because the view point lies inside of a shadow volume.

2. Which of the following problems with fake shadows is *not* solved by using a stencil buffer?
   - A. Projected shadows have the same depth as shadow receivers.
   - B. Shadows may stick out beyond shadow receivers.
   - C. Multiple occluders may give rise to double blending.
   - D. The stencil buffer solves all of these problems.

   ■ **Solution/comments:** The correct answer is
   A. Projected shadows have the same depth as shadow receivers.

Problem 8: Ray tracing

■ Subproblem 8.1 [8 pts]: Ray calculation.

Assume we want to do ray tracing from the position of a virtual camera given by the vector $\vec{e}$ in world coordinates. Further assume that we have specified a camera coordinate system around $\vec{e}$ with base vectors $\vec{u}, \vec{v}$, and $-\vec{w}$, where $-\vec{w}$ is the vector pointing at the screen. The distance between the screen and our virtual camera is d. Hence, a point on the screen can be expressed as $\vec{s}_{cam} = u\vec{u} + v\vec{v} - d\vec{w}$.

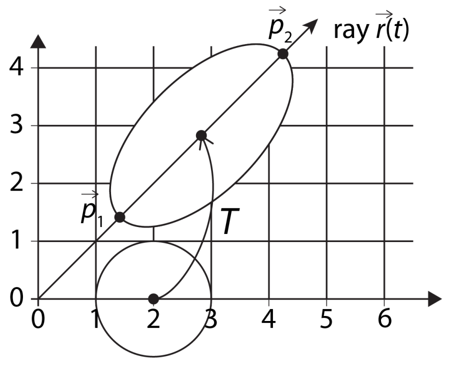
1. Now we want to create a line that can be used as ray when calculating *perspective views* of our scene.
   (a) What would be the support vector of such a line?
   ■ Solution/comments: $\vec{e}$
   (b) What would be the direction vector of such a line?
   ■ Solution/comments: $\vec{s}_{cam} = u\vec{u} + v\vec{v} - d\vec{w}$
2. Now we want to create a line for calculating *parallel (or orthographic) views* of our scene.
   (a) What would be the support vector of such a line?
   ■ Solution/comments: $\vec{e} + u\vec{u} + v\vec{v}$
   (b) What would be the direction vector of such a line?
   ■ Solution/comments: $-\vec{w}$

■ Subproblem 8.2 [5 pts]: Instancing.

The image to the right illustrates a rotated ellipse that was created by multiplying the depicted circle with the transformation matrix T. T realizes a uniform scaling by the factor of 2 in $x-$direction followed by a counterclockwise rotation of 45° about the origin. Calculate the intersection points $\vec{p}_1$ and $\vec{p}_2$ of the ray $\vec{r}(t)$ going through the ellipse. The transformation matrix T is

$$T = \begin{pmatrix} \sqrt{2} & -\frac{\sqrt{2}}{2} \\ \sqrt{2} & \frac{\sqrt{2}}{2} \end{pmatrix}.$$

(Hint: looking at the image can save a lot of calculations)

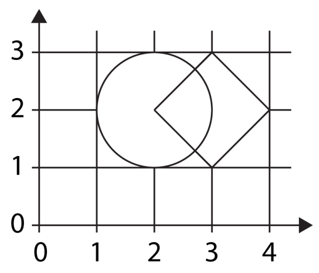


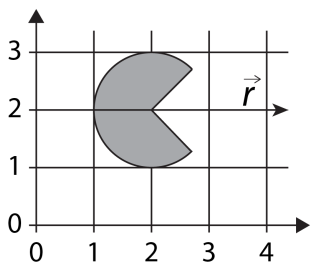
■ Solution/comments: From the image we see that the ray $\vec{r}(t)$ is mapped to the $x-$axis when applying T^{-1}, and that the corresponding intersection points with the circle are $(1,0)$ and $(3,0)$.

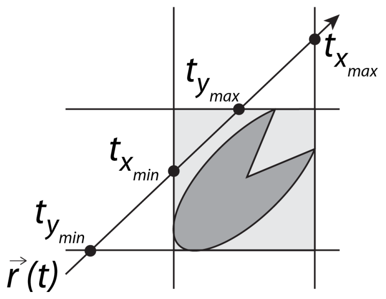
Hence, we have $\vec{p}_1 = T \cdot (1,0) = (\sqrt{2}, \sqrt{2})$ and $\vec{p}_2 = T \cdot (3,0) = (3\sqrt{2}, 3\sqrt{2})$

Make a mark here if you answered parts of these questions on an extra sheet: [ ]

■ **Subproblem 8.3 [5 pts]: Constructive Solid Geometry (CSG).** The two images to the left below illustrate how a more complex object can be created from a circle C and a rotated square S using Constructive Solid Geometry.







1. Write down what kind of set operation we have to apply to the circle C and the square S in order to create the gray shape illustrated in the second image.

   ■ **Solution/comments:** $C - S$ (or "the difference of C and S" or "S excluded from C")

2. To calculate the intersection points of the gray shape in the second image with the depicted ray $\vec{r}$ (i.e. the line $y = 2$), we need to specify the intervals representing the intersections between this ray and the original objects, i.e. the circle C and square S. What are these?

   ■ **Solution/comments:**

   Intersection interval for the circle C: $I_C = [(1,2),(3,2)]$

   Intersection interval for the square S: $I_S = [(2,2),(4,2)]$

3. What kind of set operation do we have to apply to these two intersection intervals and what is the result to this operation? Write down the operation and the resulting interval(s).

   ■ **Solution/comments:**

   Operation: $I_C - I_S$

   Resulting interval(s): [(1,2),(2,2)]

4. We can also use CSG to easily verify if a ray intersects with a bounding box. The image to the right illustrates the intersections of a ray with the four borders of a bounding box in 2D. Using this terminology, write down what kind of intervals we have to compare to check if the ray and the bounding box intersect or not.

   ■ **Solution/comments:** We have to compare $[t_{x_{min}}, t_{x_{max}}]$ and $[t_{y_{min}}, t_{y_{max}}]$

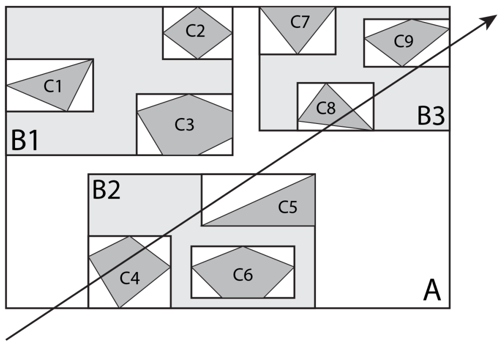
5. Because we just want to know if the ray and the bounding box intersect, but we are not interested in the actual intersection points, we don't have to calculate them. Instead, we just have to check if the intersection of these two intervals is empty or not. What conditions are fulfilled if the ray does *not* intersect with the box?

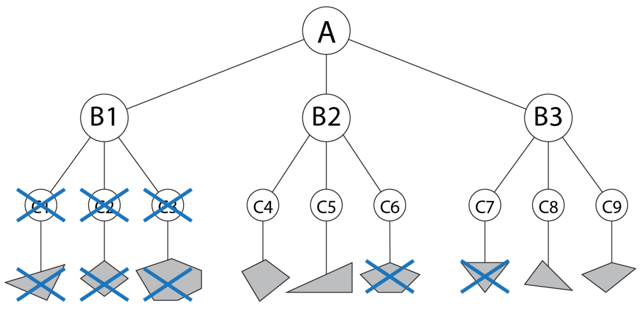
   ■ **Solution/comments:** The ray does not intersect with the box if either $t_{x_{min}} > t_{y_{max}}$ or $t_{y_{min}} > t_{x_{max}}$.

Problem 9: Methods for faster ray tracing

■ **Subproblem 9.1 [5 pts]: Hierarchical bounding boxes.** The image to the left below illustrates a distribution of a 2D space into hierarchical bounding boxes. The one on the right shows a related tree representation. We want to use this tree to speed up our ray intersection tests. Cross out all nodes and leaves in the tree that represent bounding boxes and objects, respectively, that we do *not* have to check for intersection with the ray shown in the image (which only intersects with the objects in bounding boxes C4 and C8).

■ **Solution/comments:**





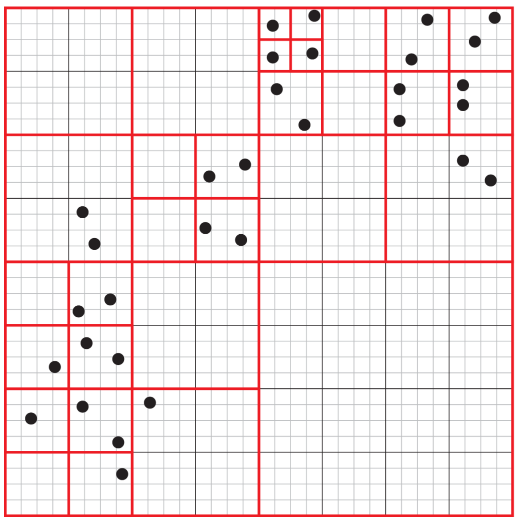
How many *false positives* do we have in the above example?

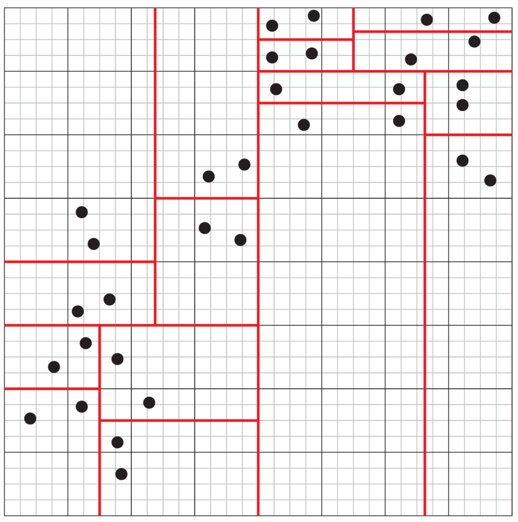
■ **Solution/comments:**

There are 2 false positives.

■ **Subproblem 9.2 [6 pts]: Space partitioning approaches.**
Below are two images of the same 2D space containing 32 objects (= black dots).

■ **Solution/comments:**





For the ***left one***:
Use the ***Octree*** approach to subdivide the space in cells. In the end, each cell should contain maximum two objects. For the ***right one***: Use the ***BSP tree*** approach to subdivide the space in cells. In the end, each cell should contain maximum two objects. Start partitioning the space with a vertical split.

Make a mark here if you answered parts of these questions on an extra sheet: [ ]