Exercise 1 – Rasterization & Ray Tracing

a) ***Describe briefly the process of ray tracing*** in form of a high-level pseudo code (assume a function *intersect* is given, which takes a triangle and a ray and returns a bool, indicating an intersection, as well as a color, the 3D position and depth of potentially intersected geometry).
b) +Please ***list briefly*** two advantages and disadvantages each of Rasterization and Ray Tracing.
c) *Please write an intersection function for a plane (given by a point in the plane P:=(p1,p2,p3) and two direction vectors u:=(u1,u2,u3), v:=(v1,v2,v3)) and a ray (defined by an origin r:=(r1,r2,r3) and a direction d:=(d1,d2,d3)). That returns an object *Hit*, containing a **bool** to indicate whether there was an intersection, and if there is an intersection, the ***intersection point in 3D*** and ***a distance to the ray origin***.

a) Solution (a similar approach is also acceptable, it does not have to be 100% the same code)

```
For each pixel
        Distance=MAX
        Color=0
        Ray=computeRay(pixel)
        For each triangle
                (CurrColor,CurrDistance)=computeIntersection(Ray, triangle)
                If (CurrDistance<Distance)
                {
                        CurrDistance=Distance
                        Color=CurrColor
                }
```

b) Other benefits and drawbacks could be possible as well.

Benefits of Rasterization:
Fast (for primary effects), easily parallelizable, scales well with resolution…
Drawbacks of Rasterization:
Difficult to produce effects (depth of field, shadows etc.), standard version scales badly with geometry…

Benefits of Ray Tracing:
very flexible, can be made physically correct (and unbiased), easy to incorporate secondary effects, scales better with geometry…
Drawbacks of Ray Tracing: usually slow for medium-scale scenes, overhead for dynamic scenes, complicated structures that can be difficult to implement…

c) Different techniques are possible and there are many alternatives - ALL are accepted if correct.

As the code of the students will vary I will list some abstract pseudo-code here

```
n = normalize(cross(u,v)) // compute the normal of the plane
// there is always an intersection except if d is orthogonal to n
if ( dot(d,n) == 0 ) return noHit // or test for fabs(dot(d,n))<Epsilon
```

```
d = dot(P,n) // distance of plane to origin
// plane equation is dot(p,n) – d = 0
// substitute ray for p, so that
// dot( (r + t*d), n ) – D = 0
// then t equals
t = (D - dot(r,n)) / (dot( d,n ))
distanceToRayOrigin = (t*d) / length(d)
intersectionPoint = r + t*d
hit = true
```

Here is another example:

Solve for t, alpha, beta in R+tD=P + alpha u + beta v.

$R_x+tD_x= a_x+\beta\,(b_x-a_x)+\gamma\,(c_x-a_x)$
$R_y+tD_y= a_y+\beta\,(b_y-a_y)+\gamma\,(c_y-a_y)$
$R_z+tD_z= a_z+\beta\,(b_z-a_z)+\gamma\,(c_z-a_z)$

$$\begin{bmatrix} a_x-b_x & a_x-c_x & D_x \\ a_y-b_y & a_y-c_y & D_y \\ a_z-b_z & a_z-c_z & D_z \end{bmatrix}\begin{bmatrix} \beta \\ \gamma \\ t \end{bmatrix}=\begin{bmatrix} a_x-R_x \\ a_y-R_y \\ a_z-R_z \end{bmatrix}$$

So, using, e.g., Cramer’s rule

$$\det\begin{bmatrix} a_x-b_x & a_x-c_x & a_x-R_x \\ a_y-b_y & a_y-c_y & a_y-R_y \\ a_z-b_z & a_z-c_z & a_z-R_z \end{bmatrix} \quad \text{divided by} \quad \det\begin{bmatrix} a_x-b_x & a_x-c_x & D_x \\ a_y-b_y & a_y-c_y & D_y \\ a_z-b_z & a_z-c_z & D_z \end{bmatrix}$$

If the last determinant is zero, the plane and ray are parallel.

Exercise 2 – Projective Geometry and Homogeneous coordinates

In this exercise, we assume to be in a projective space with 3 coordinates – so a 2 (!) dimensional space.

a) +Give the definition of a projective vector space.

b) Given the following matrix:

$$M = \begin{bmatrix} \mathrm{s}\cos(t) & -\mathrm{s}\sin(t) & 0 \\ \mathrm{s}\sin(t) & \mathrm{s}\cos(t) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

1) What is the influence of the parameters s and t?
2) Rewrite M as the multiplication of ***two matrices*** one containing only s, one only t.
3) What influence does it have if you use a parameter ***x := -t*** instead of t?

c) *Please write down the matrix for a rotation around a point Q(1,1) by an angle $a = \pi$. Simplify the values in the matrix as much as possible.

Solution:

a)

In mathematics, a projective space is the set of lines through the origin of a vector space V.
$\mathbf{P}^n(\mathbf{R}) := (\mathbf{R}^{n+1} \setminus \{\mathbf{0}\}) / \sim$,
where $\sim$ is the equivalence relation "$(x_0, \ldots, x_n) \sim (y_0, \ldots, y_n)$ if there is a non-zero real number λ such that $(x_0, \ldots, x_n) = (\lambda y_0, \ldots, \lambda y_n)$".
Or more simply spoken, to project a vector to the projective space add a 1 as the homogenous coordinate. You can reproject all other values to the Euclidian space by simply dividing by the homogenous coordinate.
If a student does not exclude 0 here: $(\mathbf{R}^{n+1} \setminus \{\mathbf{0}\})$, that is fine, but n+1, mentioning relation, and a non-zero real number λ is important!

b)

1) M is a combination of a scaling Matrix S and a rotation matrix R where M = R*S, i.e. s is the scaling factor and t is the rotation angle.

2)

$$R = \begin{bmatrix} \cos(t) & -\sin(t) & 0 \\ \sin(t) & \cos(t) & 0 \\ 0 & 0 & 1 \end{bmatrix}, S = \begin{bmatrix} \mathrm{s} & 0 & 0 \\ 0 & \mathrm{s} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3) Rotation in the other direction (clockwise)

c)

Create the matrix $M = T_2 R T_1$,

$$T2 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \quad R = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad T1 = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix},$$

$$M = \begin{bmatrix} -1 & 0 & 2 \\ 0 & -1 & 2 \\ 0 & 0 & 1 \end{bmatrix}$$

Exercise 3 – Shading

a) Given one light wavelength, and the following constants:

Surface : Kd (diffuse coefficient) = 0.5,

Ks (specular coefficient) = 0.5, s(shininess)=1

Surface position = (0,1,0), surface normal (0,0,1)

Light : Id (diffuse light) = 1, Is (specular light) = 1

Light position = (0,2,2), viewpoint (0,2,1)

1) +Please compute the value of the diffuse term of the Phong Material Model
2) *Please give the specular term of the Phong Material Model (**no need to compute** the value)

b) Why would a Phong material with Kd = 0.6, Ks = 0.7, s=1 not be physically plausible.

Solution:

a)

1)

Light vector l = light position – surface position = (0,1,2), normalized nl = (0, 1/sqrt(5), 2/sqrt(5))
The diffuse term is Id * Kd * cos(dot(nl, n)) = 1 * 0.5 * cos(2/sqrt(5))= cos(2/sqrt(5))/2
this is roughly: 0,447 – the transformation to a number is NOT(!) needed!

2)

Specular term is ks * Is * cos^s(dot(r,nv))
2 points for filling in the right numbers
View vector v = viewpoint – surface position = (0,1,1)
normalized nv = (0, 1/sqrt(2), 1/sqrt(2))
Ks=0.5

r is the normalized reflected light vector refl, computed by refl = v – 2* dot(n,v) * n
= (0,1,1)-2 n =(0,1,-1), hence, r=(0,1/sqrt(2), -1/sqrt(2)), thus
ks * Is * cos^s(dot(r,nv)) = 0.5 cos (0)=0.5

b) With these coefficients, more light can be reflected than what comes from the source

Exercise 4 – OpenGL

a) *The goal is to see a blue and a red triangle on the screen. ***Make a list of mistakes*** in the code - take your time, there are **several** (!):

```
glBegin( GL_QUADS );
   glVertex2f( 0.0, 0.0 );
   glColor3fv( 1,0,0);
   glVertex2f( 1.0, 0.0 );
   glColor3fv( 1,0,0);
   glVertex2f( 0.0, 1.0 );
   glColor3fv( 1,0,0);
glEnd();
glBegin( GL_QUADS );
   glVertex2f( 0.0, 0.0 );
   glColor3fv( 0,1,0);
   glVertex2f( 1.0, 0.0 );
   glColor3fv( 0,1,0);
   glVertex2f( 0.0, 1.0 );
   glColor3fv( 0,1,0);
glEnd();
```

b) +Make ***two sketches*** that illustrate the resulting triangles when defining TYPE as GL_TRIANGLES or GL_TRIANGLE_STRIP:

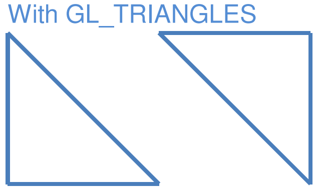
```
glBegin( TYPE );
   glVertex2f( 0.0, 0.0 );
   glVertex2f( 0.0, 1.0 );
   glVertex2f( 1.0, 0.0 );
   glVertex2f( 1.0, 1.0 );
   glVertex2f( 2.0, 0.0 );
   glVertex2f( 2.0, 1.0 );
glEnd();
```

a)

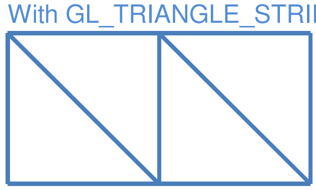
1) GL_QUADS must be GL_TRIANGLES
2) glColor must be called before glVertex
3) glColor3fv must be glColor3f
4) The second triangle is green not blue
5) The second triangle is drawn at the exact same location as the first triangle

b)

With GL_TRIANGLES



With GL_TRIANGLE_STRIP



Exercise 5 – Textures

a) +Explain **briefly** the purpose of Mipmaps (2-3 sentences).

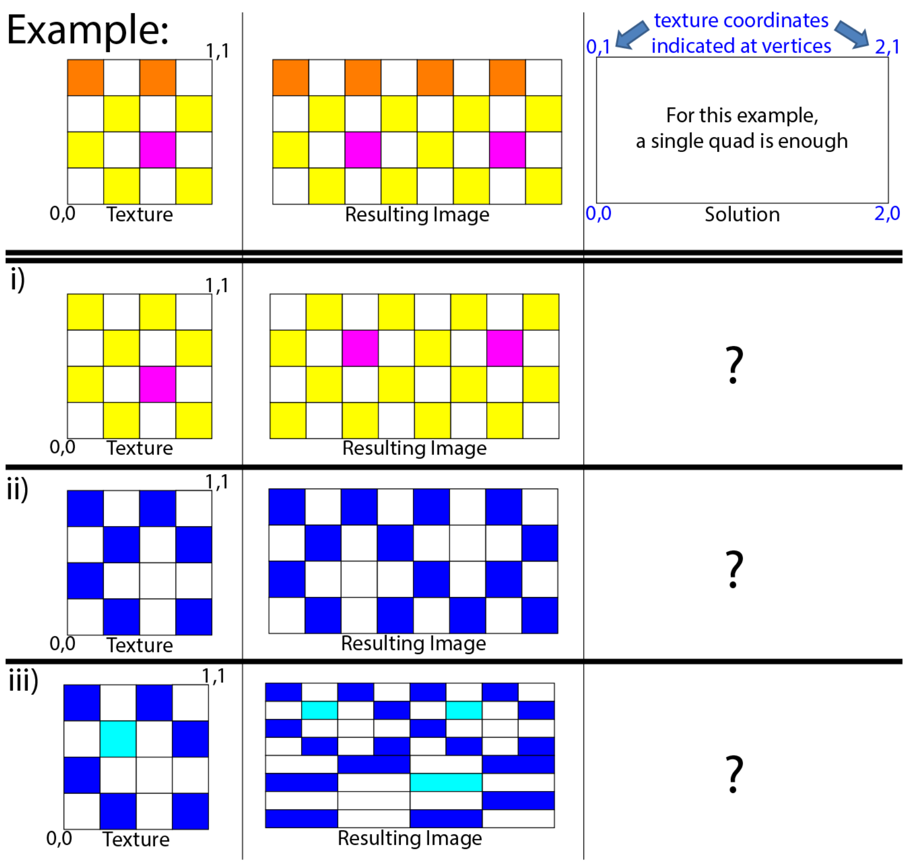
b) Please write down **all Mipmap levels** of the following grayscale 4x4 texture:

$$\begin{bmatrix} 1 & 1 & 0.5 & 1 \\ 1 & 1 & 1 & 0.5 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

c) ***** How much more **memory** do Mipmaps consume compared to the corresponding original texture (level 0) of resolution 2^nx2^n. Please find a factor and give an explanation for it.

d) Given that the highest level in the Mipmap chain is *n*>3, what resolution does level *n*-3 have?

e) Please define ***for the lower three cases*** (the top row is an example!) a projected 2D quad mesh with texture coordinates (and a ***minimal number of quads***), such that using the texture on the left produces the image on the right. Assume that **textures are repeated** when texture coordinates are outside [0,1].



a) A rough answer like this is enough, or even a shorter version: A Mipmap is an image pyramid containing filtered versions of a texture, each level averages 4 neighboring pixels. During rendering levels with pixels closest in size to the footprint(projection) of the screen pixel are chosen for an approximately correct filtering inside the screen pixel.

b) $\begin{bmatrix} 1 & 3/4 \\ 1/2 & 1 \end{bmatrix}$, [13/16]

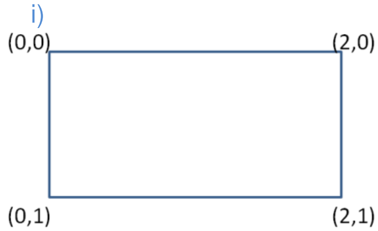
c) The factor is roughly 4/3 (minus a little part, but that is not so important and if someone explained this, the solution is fine). The reason is that the mipmap pyramid describes a geometric series $\sum_i \frac{1}{4^i} T = (\frac{4}{3} - \frac{4^{-n}}{3})$T, where n=levels. There is a geometric drawing solution similar to the anisotropic textures - it also counts:

What you see is an overview of the memory consumption for 3 times the texture (indicated with Red, Green, Blue).
Now, you can see that in the limit, 4 times the memory of the base texture is used for all 3 MipMap chains together - each MipMap level is indicated via a black box. Because 3 MipMap chains lead to 4 times the memory consumption, 4/3 *T is the rough memory consumption of a single MipMap.

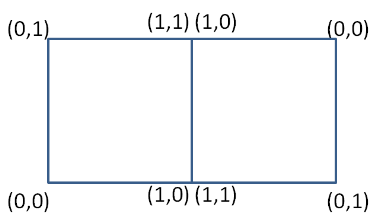
d) 8x8

e) **Several solutions are possible** – all correct ones are fine (especially, many shifted by 1/4 etc):

i)



ii)



iii)