UNIVERSITÄT DES SAARLANDES
Prof. Dr.-Ing. Philipp Slusallek
Computer Graphics Group
Hugo Devillers (devillers@cg.uni-saarland.de)
Ömercan Yazici (yazici@cg.uni-saarland.de)

17. January 2022

Introduction to Computer Graphics
Assignment 7

Submission deadline for the exercises: 28. January 2022

The theoretical parts should be uploaded as a single PDF (scan, LaTeX, ...) in the assignment in Microsoft Teams. Please write your name, the name of your partner and the group name (e.g. group42) on top of the sheet. Both of the team members have to upload and submit the same PDF!
For guest students or those who have trouble with Teams, we can exceptionally accept submissions by email, in which caseplease send them by email to Ömercan.
The programming parts must be marked as release before the beginning of the lecture on the due date.
The code submitted for the programming part of the assignments is required to reproduce the provided reference images. The submission ought to tag the respective commit in your group's git repository as well as attach the mandatory generated images to the release. The submission should also contain a creative image show-casing all extra-credit features that have been implemented.
The projects are expected to compile and work out of the box. A successful build by Drone CI is a good indicator.
To pass the course you need for every assignment at least 50% of the points.

7.1 Color Space (10 Points)

Compute the position of the sRGB color (1,0,1) in the CIE-XYZ and CIE-xy color space. Argue why gamma correction in sRGB is irrelevant for the transformation of this particular color.

7.2 Perspective Projection (15 Points)

a) Compute the point where two arbitrary parallel lines seem to intersect after being projected by the perspective projection P. For which parallel lines does no such intersection point exist?

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1 \end{pmatrix}$$

b) Compute the center of projection of the following perspective projection Q.

$$Q = \begin{pmatrix} \frac{3}{4} & 1 & -\frac{1}{4} & -\frac{1}{4} \\ 0 & 0 & \frac{1}{4} & \frac{5}{4} \\ -1 & 0 & 0 & 1 \\ 0 & 0 & \frac{1}{4} & \frac{5}{4} \end{pmatrix}$$

c) Compute the projection plane of projection Q'

$$Q' = \begin{pmatrix} 1 & \frac{1}{4} & -\frac{1}{4} & 0 \\ 0 & \frac{3}{4} & \frac{1}{4} & 0 \\ 0 & -\frac{1}{4} & \frac{5}{4} & 0 \\ 0 & \frac{1}{4} & -\frac{1}{4} & 1 \end{pmatrix}$$

7.3 Clipping (15 Points)

Given an axis aligned clipping window spanned by the two points $(0,0)$ and $(12,6)$

$$C_1 = (0,0), C_2 = (0,6), C_3 = (12,6), C_4 = (12,0)$$

and a polygon with the lines $P_1 - P_2 - P_3 - P_4 - P_1$ with

$$P_1 = (9,8), P_2 = (-6,3), P_3 = (14,20), P_4 = (15,-4).$$

Use the Sutherland-Hodgeman algorithm to perform the first step of the algorithm by clipping the polygon against the top edge of the clipping window only. Describe each step of the algorithm. You do not have to compute the intersection point of the lines with the clipping window, give them names in a figure and use that names instead.

7.4 Hermite Spline (15 Points)

The following cubic polynomial defines a spline curve in 3D:

$$p(t) = at^3 + bt^2 + ct + d \texttt{ with } a,b,c,d \in \mathcal{R}^3$$

a) Compute the coefficients $a, b, c,$ and d of the polynomials such that $p(0) = (0,1,0)^T$, $p(1) = (1,0,1)^T$, $\frac{dp}{dt}(0) = (1,0,1)^T$, $\frac{dp}{dt}(1) = (0,-1,0)^T$. You have to write down the constraints and solve an system of equations.

b) Compute the same coefficients by using the Hermite basis.

7.5 Bump Mapping (20 Points)*

A common technique of increasing model complexity without actually adding to the geometry is to perturb the normal vectors returned when the geometry is intersected. In this exercise your task is to implement *bump mapping* where the normal vector perturbation is defined by a texture which represents an imaginary heightfield. This way, when the surface is lit, it seems to have bumps although it remains a single, flat geometry.
Your task is to implement `BumpMapper`, which holds a triangle solid as its base, and adds a bump texture onto it. The location of the texture is defined by texture coordinates at the triangle vertices. The parameter `vscale` controls the magnitude of the bumps.
When the underlying triangle is intersected, the bump mapper should perturb the normal. You will need to:

- Compute the bump map texture coordinates where the hit occured.
- Compute the gradient at the given texture coordinates. Assume that the image is gray. (`Texture::getColorDX()`, `Texture::getColorDY()`)
- Compute how the texture base vectors e_x, e_y map to world space w_x, w_y.
- Multiply w_x, w_y and the gradient to perturb the normal in world space.

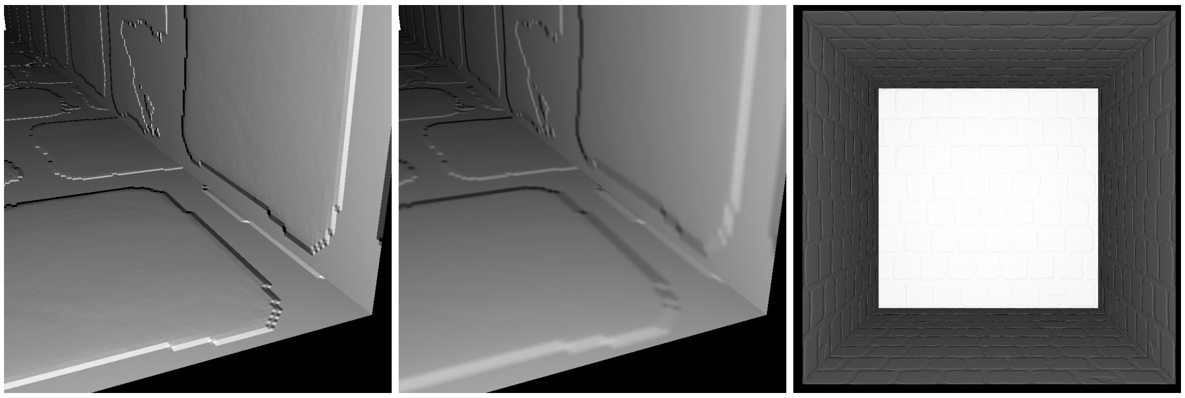

Figure 1: Bump mapping. Close-up, using nearest-neighbour and bilinear interpolation; and full scene.