



14. JANUARY 2019

## INTRODUCTION TO COMPUTER GRAPHICS ASSIGNMENT 8

**Submission deadline for the exercises:** 21. January 2019

The paper copies for the theoretical parts of the assignments will be collected at the beginning of the lecture on the due date. The programming parts must instead 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. If it fails to do so on our Windows or Linux computers, we will try to do so on the CIP-pool students' lab as a "fallback".

**To pass the course you need for every assignment at least 50% of the points.**

### 8.1 Transforms in color space (5 + 25 Points)

Many of today's monitors allow to change the color temperature of the image, by moving the white point in the color diagram. .

A monitor corresponding to the standard "ITU-R BT.709" (which is identical to sRGB) uses the following color coordinates  $(x_r, y_r) = (0.640, 0.330)$ ,  $(x_g, y_g) = (0.300, 0.600)$ ,  $(x_b, y_b) = (0.150, 0.060)$ , and a D65 white point  $(x_w, y_w) = (0, 3127, 0.3290)$ .

- While monitors can have their white point changed, its primary colors of the monitor cannot move in the color diagram. With this restriction, can you think of an easy way to perform a movement of the white point on a monitor?
- Calculate the necessary transformation matrix to convert a color from the color space given above into a new one with the same primaries but with the white point at  $(w_x, w_y) = (0.400, 0.330)$ . Explain your solution. Since this transformation is not used to change the brightness, assume a normalized white point. Note that while sRGB colors are gamma corrected with gamma 2.2, in this exercise you are only asked to derive the transformation matrix and ignore the gamma correction.

### 8.2 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.

### 8.3 Bresenham for Parabola (20 Points)\*

Develop a mid-point algorithm to draw the fixed parabola  $y = \frac{1}{12}x^2 + 2$  in the range of 0 to 6. You do not have to implement the algorithm, but to show how to initialize  $x$ ,  $y$ , and the decision variable  $d$  and how to update them in the two cases.

## 8.4 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 occurred.
- 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.

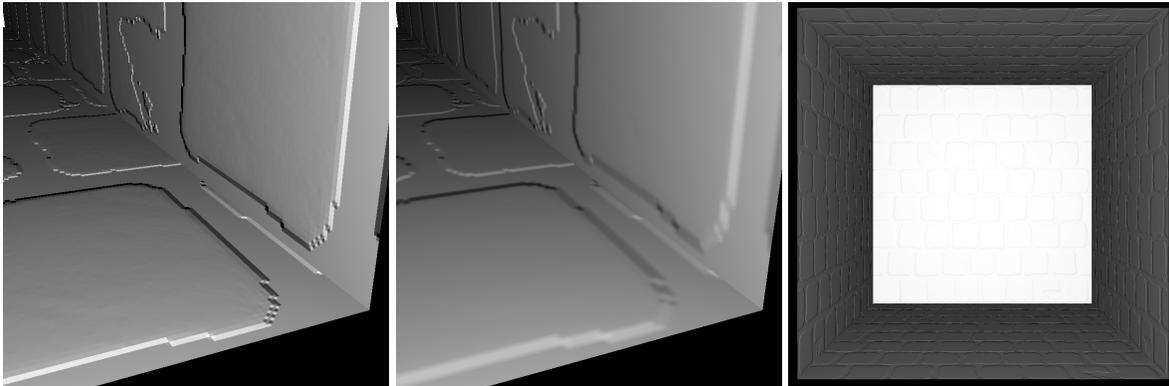


Figure 1: Bump mapping. Close-up, using nearest-neighbour and bilinear interpolation; and full scene.

## 8.5 Volume Rendering (10 + 15 + 15 Points)\*

Extend the `World` to support a pervasive fog.

Implement a new integrator that simulates this fog.

- Assume that the fog is uniform, and lit by an ambient light source. You need to add the attenuation factor for your primary and shadow rays. The primary rays (and only those) should additionally account for the ambient light reflected by the fog.
- Assume that the fog is uniform, but is lit by actual light sources in the world. You will need to perform ray marching along the primary rays, but attenuation of shadow rays can still be done analytically.
- Assume the fog is heterogenous (e.g. defined by some 3D perlin noise texture). Perform the ray marching along the primary and shadow rays.

Pay attention to rays reaching to infinity. If you try to perform ray marching of constant step along such a ray, your program will hang.