Prof. G. Zachmann
H. Meißenhelter
T. Hudcovic

University of Bremen
School of Computer Science
CGVR Group
May 7, 2025

Summer Semester 2025

Assignment on Advanced Computer Graphics - Sheet 2

Due Date 22.05.2025

In this assignment we will raytrace Constructive Solid Geometry (CSG). On the website you can find an adapted version of the already known *RaytracingFramework* which now supports scenes containing CSG trees.

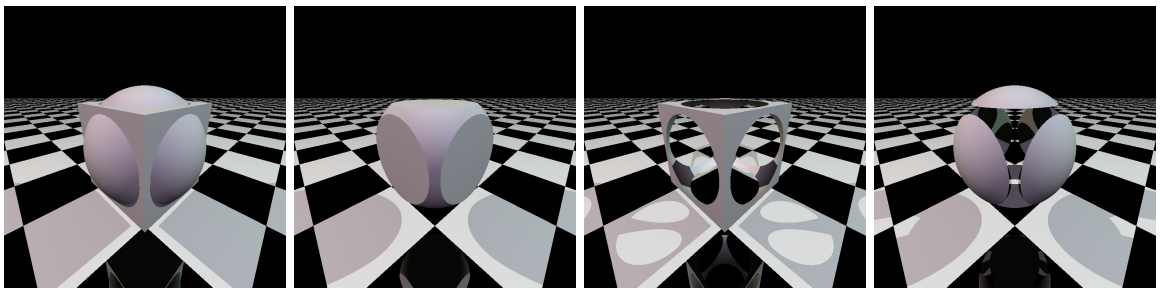

Figure 1: How the four example scenes *CSG Union*, *CSG Intersection*, *CSG Difference A* and *CSG Difference B* should look like after the task is done correctly. For each of these scenes it is necessary to implement only one of the functions (corresponding to its name) to be rendered correctly.

Exercise 1 (Constructive Solid Geometry, *12 Credits*)

In the framework you will find the new files `CSGPrimitives` and `CSGTree`, which contain all classes and structures relevant for this task. A few notes about the classes that are implemented there:

- `CSGTree` implements the tree structure and has a `CSGNode` as root node.
- `CSGNode` implements a node of the CSG tree containing either a `CSGPrimitive` (if the node type is set to `LEAF_NODE`) or two child nodes (when one of the other node types is set).
- `CSGSphere` and `CSGCube` are classes that inherit from `CSGPrimitive` and implement an intersection test that return all intersection points (instead of just the first one). Since these objects are convex, there are at most two intersection points.
- `CSGIntersectionInterval` is a struct that stores both (a) the intersection point when the ray enters the object and (b) the intersection point when it leaves, including the ray scalar t as well as the normal of the surface.

To raytrace CSGTrees correctly, it is not enough to find only the first intersection point. All intervals of entry and exit points along a ray of the CSG Tree must be found. Depending on the operation (union, intersection, difference), these intervals must then be combined in different ways while traversing the tree from bottom to top. In the framework, both the intersection test and the traversal of the tree are given — i.e. you don't have to deal with recursions. You can also assume the intervals are initially being passed sorted. Ideally, your solution keeps the sorting order (if not, resort at the end).

Your only task in this assignment sheet is to implement the merging of the intervals of entry and exit points, which differ depending on the operation (Union, Intersection and Difference), which are declared by the functions `CSGTree::csgUnion`, `CSGTree::csgIntersection` and `CSGTree::csgDifference`. After you implemented all three functions, the *CSG Complex* scene should look like shown in Figure 2. For this exercise you can refer to slides "Constructive Solid Geometry (CSG)", "The CSG Tree by Way of an Example", and "Rendering CSG Objects Using Raytracing" in the lecture on *Non-Polygonal Object Representations*.

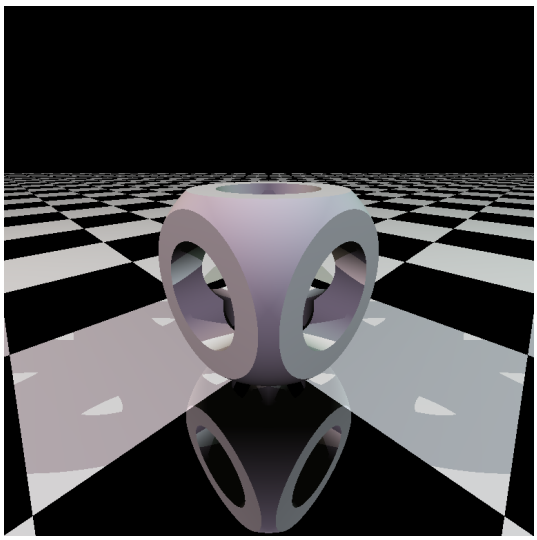

Figure 2: How the scene *CSG Complex* should look like after all three functions have been implemented.

Hint: Make sure to test your solutions in the various scenes, as well as to move the camera in the scene, to spot possible artifacts.