Visual Raytrace: An Immersive Learning Application

Manfred Brill, Benedict Särota

University of Applied Sciences Kaiserslautern
Amerikastr. 1
66482 Zweibrücken
Tel.: +49 (0)631 / 37 24 - [5382, 5321]
E-Mail: [manfred.brill, benedict.saerota]@hs-kl.de

Abstract: Visual Raytrace is an immersive learning application supporting students in a computer graphics class to create their own version of a working raytracing software. the users can interact with all ingredients of a raytracer like camera model, framebuffer, scene description, sampling, reflections models or texture mapping.

Keywords: Virtual Reality, Education, Immersive Learning, Computer Graphics Classes, Raytraing

1 Introduction

Raytracing, introduced by Turner Whitted [Whi80], is one of the major topics in computer graphics classes. In figure 1 we find on of the famous pictures rendered by Whitted.

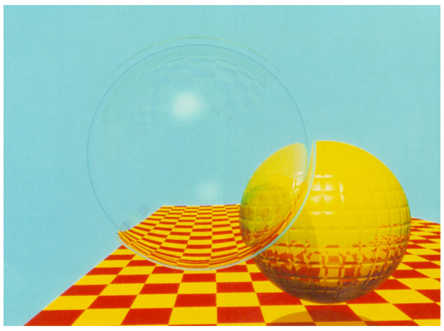

Figure 1: Image rendered by Turner Whitted (Source: `https://de.wikipedia.org/wiki/Datei:Spheres_and_Checkerboard_-_Turner_Whitted.jpg`)

Usually students in a computer graphics class implement their version of such a renderer, after getting the theory in the class. To achieve this they need to understand the basic concepts of computer graphics like coordinate systems, cameras, reflection models or texture mapping. To implement a raytracer or another 3D application the students have to develop a strong visual-spatial ability. Virtual reality applications provide joy of use and support the transfer from 3D space to a programming language and deepen the understanding of the basic concepts of a raytracer.

2 Visual Raytrace

We implemented the immersive learning application **Visual Raytrace** [Sä21a, Sä21b] using the Unity Game Engine and HTC vive Input Utility ([Won21]). Figure 2 shows the main features of the application.

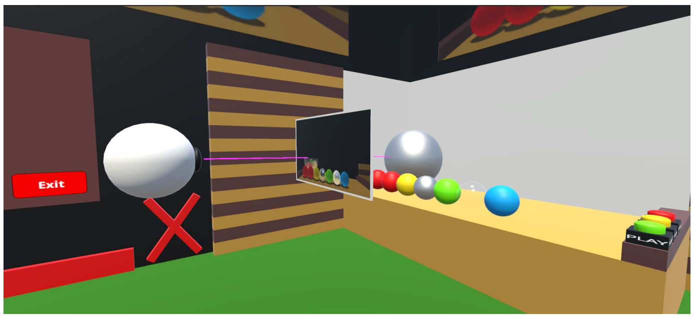

Figure 2: The elements of a raytracer in an immersive environment

The raytracer is implemented in C# und is slowed down to make sure the users can follow the path of the ray and the reflections in the scene until the color of the pixel is finally computed. The raytracing process can be paused or soptted to get an understanding of the rendering algorithm. At the end the computed image can be saved for further investigation.

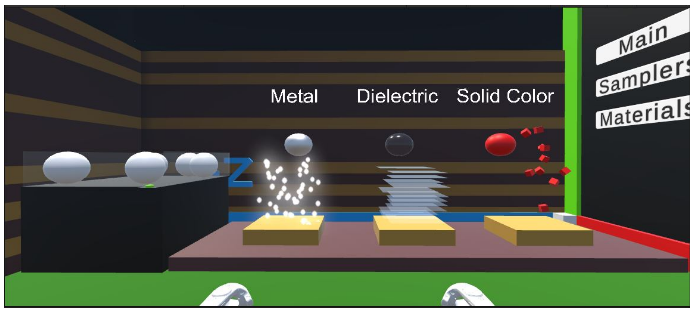

Figure 3: Geometric objects and reflection functions for scene definition

The scene can either be defined in VR or can be read from a scene description file. Figure 3 shows the interface for the interactive scene definition. New objects can be picked and one of the reflection models can be assigned. Afterwards users position this new object in the rendered scene. The options for the raytracer can also be set in the immersive environment.

3 Future Work

We plan to use the immersive learning application in the next computer graphics class in the summer term 2022. The implementation of a working raytracer is the assignment in the class, so we will transfer the C# code for the raytracer to an own repository and link the renderer as a dynamic library. Another possibility would be to change the programming language used in the computer graphics class to C#. Doing this the learning application and the raytrace could be made both public. the assignment could then be changed. This has to be discussed and decided with the lecturers for the next summer term.

It is planned to implement a change of scenes so the users can dive into the 3D world currently being raytraced and back.. This way the students can walk around in the scene currently rendered and have a close look to intersection points, reflections or refractions.

Due to the pandemic situation on our campus we could not evaluate the application with students. Hopefully we will be able to do this in the winter term 2021/22 at our campus. the results of this evaluation will help us to improve the usability and joy of use of the application. Now the software runs in our lab with a HTC Vive Pro and Vive Focus Plus head-mounted displays. The vive Input Utility supports builds for Google Cardboard or Oculus Android. We also work on an OpenXR- and WebXR-based versions to support as many platform as possible to distribute the application on the home equipment of the students.

References

[Sä21a] Benedict Särota. Implementation of a vr application for the visualization of a raytracing process, 2021. Project Work, University of Applied Sciences Kaiserslautern.

[Sä21b] Benedict Särota. Raytracing. `https://github.com/VRLAB-HSKL/RayTracing`, 2021. Last Access: July 22., 2021.

[Whi80] Turner Whitted. An improved illumination model for shaded display. *Communications of the ACM*, 23(6):343 – 349, 1980.

[Won21] Lawrence Wong. Vive input utility for unity. `https://github.com/ViveSoftware/ViveInputUtility-Unity`, 2021. Last Access: July 22., 2021.