Hogra: A Homebrew Graphics Engine

Zoltán Simon
Supervised by: Dr. Balázs Csébfalvi

Department of Control Engineering and Information Technology, Faculty of Electrical Engineering and Informatics,
Budapest University of Technology and Economics, Műegyetem rkp. 3., H-1111 Budapest, Hungary

Motivation

As computer engineering students, we are interested in the inner workings of rendering systems. These applications have a fairly convoluted architecture, challenging the creator's skills and knowledge. Having our implementation allows greater flexibility when adapting the program for different use cases. We could also utilize our system as an educational tool. This motivated our venture to develop our own real-time rendering engine that besides being capable of creating video games has already proved to be useful as the platform for scientific visualization projects.

Related work

Current state-of-the-art game engines include Unity and Unreal Engine. Similar – but simpler compared to previously described ones – engines have been created by individuals, small teams, and open source communities in the past. Godot is a video game engine celebrated by many small developer teams. Hazel is another example of a simplistic engine created by Yan Chernikov and his team. Their intention with the Hazel project is to educate game developers via regularly published video diaries and by making the source code available on the internet. PBRT is an educational engine built to demonstrate the concepts described in the Physically Based Rendering: From Theory To Implementation book.

Our work

Our implementation combines existing techniques. The engine is written in C++. We use OpenGL for rendering. We provide a simple collision and physics engine. When it comes to visualization, we use Physically Based Rendering (PBR). Our current implementation uses deferred shading to lower the cost of PBR shading. We are working on upgrading to forward+ rendering. We have integrated a post-processing pipeline, which supports visual effects such as bloom or different tone mapping techniques. It is also trivial to create new custom post-processing effects. Our architecture supports instanced rendering thus together with deferred shading enables scenes with a large number of objects. Our software has a layer system, which allows combination of different rendering techniques in the same scene. This means that we can create scenes, where forward and deferred shading are used together. Layers can have different post-processing effects applied to them. All of this can be described in a single JSON file enhancing the scene creation process. We support Unicode text rendering with custom fonts but also provide an interface towards ImGui as an easy-to-use windowing library. We have also integrated audio rendering using OpenAL soft. Our future plans include porting to Vulkan API and introducing state-of-the-art multithreading techniques. Owing to the lucidity, modularity, and transparency of our implementation, the presented render engine currently serves as the foundation of multiple research projects, and is also ready to be applied in classrooms for teaching Computer Graphics.

Results

Our project has not arrived at a finish line yet but has served as the platform of our choice for various scientific visualization projects at the university. One of these projects required support for the visualization of volumetric datasets. We utilized the layer system to combine rendering of CT / MRI datasets using half angle slicing and ray tracing methods. A basic forward rendering layer was used to display utility icons such as a light bulb showing the direction of illumination and the bounding box of the volumetric dataset. We used a separate layer to render the actual volume. The user could toggle settings via ImGUI.An ongoing project requires visualization of the so called Chinese magic mirrors. Here we utilize the scene loading from JSON configuration file to make changing settings easy.

Conclusion

We have created a graphics engine, which is capable of simulating and rendering complex 3D scenes. It has a highly customizable render layer system. We have used this framework to create scientific visualization applications.

Acknowledgement

We are thankful for the guidence of
Dr. Balázs Csébfalvi.

This work was supported by
OTKA K-124124.