Scene Conversion for Physically-based Renderers

Luiza Hagemann
Instituto de Informtica
Universidade Federal do Rio Grande do Sul
Porto Alegre, RS
lahagemann@inf.ufrgs.br

Manuel Menezes de Oliveira Neto
Instituto de Informtica
Universidade Federal do Rio Grande do Sul
Porto Alegre, RS
oliveira@inf.ufrgs.br

***Abstract*—The abstract goes here.**

I. Introduction

II. Related Works

III. System Architecture

Our converting pipeline is subdivided into three main states: the **Import Module**, the **Canonical Scene Representation** and the **Conversion Module**, as illustrated in Figure 1. Given an arbitrary input file format, our converter is able to import the scene and transform it into a generic, canonical representation and then export it to different output formats.

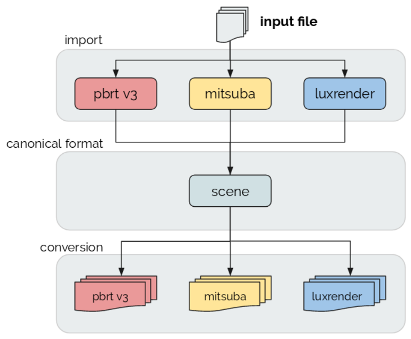

Fig. 1. Illustration of the system pipeline.

Our Proof of Concept encompassed *PBRT* [1], *Mitsuba* [2] and *LuxRender* [3], as these are three of the most popularly used renderers in the community.

A. Import Module

Our import module specializes in reading and interpreting input scene files. The input file is read, parsed and each directive is loaded into our canonical scene representation. Since each renderer has its own proprietary file format, we have three importing modules: one for each renderer.

PBRT and *LuxRender* input file formats are composed of structured text statements which define all scene directives. In order to properly parse these files, a Lex/Yacc parser was considered the best choice. As we intended to keep our system in pure Python, we chose to use PLY [4].

The input file format used in *Mitsuba* consists of a XML file. There are several XML-parsing libraries for Python that can load the hierarchy into a tree data structure; therefore, we did not see the need to create a Lex/Yacc parser. We chose to implement this module using ElementTree [**?**].

B. Canonical Scene Representation

Subsection text here.

C. Conversion Module

Subsection text here.

IV. Results and Analysis

Results section.

V. Conclusion

The conclusion goes here.

Acknowledgment

The authors would like to thank...

References

[1] M. Pharr and G. Humphreys, “Pbrt, version 3,” 2015. [Online]. Available: https://github.com/mmp/pbrt-v3
[2] W. Jakob, “Mitsuba: Physically based renderer,” 2014, http://www.mitsuba-renderer.org/.
[3] J.-P. Grimaldi and T. Vergauwen, “Luxrender v1.6,” 2008, http://www.luxrender.net/.
[4] D. Beazley, “Ply (python lex-yacc),” 2001. [Online]. Available: http://www.dabeaz.com/ply/