UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

LUIZA DE AZAMBUJA HAGEMANN

A Converter for Physically-Based Renderer Scenes

Work presented in partial fulfillment
of the requirements for the degree of
Bachelor in Computer Science

Advisor: Prof. Dr. Manuel Menezes de Oliveira Neto

Porto Alegre
May 2018

ABSTRACT

Placeholder.

Keywords: Physically-based rendering. scene conversion. PBRT. Mitsuba. LuxRender.

Um conversor de cenas para renderizadores fisicamente realísticos.

RESUMO

Placeholder.

Palavras-chave: Physically-based rendering. PBRT. Mitsuba. LuxRender..

LIST OF FIGURES

LIST OF ABBREVIATIONS AND ACRONYMS

PBR Physically Based Rendering

PB Physically-based

CONTENTS

1 INTRODUCTION

Ever since the development of modern Computer Graphics, one of the goals researchers aspired to was being able to synthesize images indistinguishable from real photographs. In order to produce physically accurate images, the process of image synthesis - also called **rendering** - simmulates the interaction of light with the representation of a three-dimensional scene.

Physically-Based Rendering (PBR) is a complex process that requires thorough knowledge of optics, material properties, geometry and light propagation.

1.1 Physically Based Rendering (PBR)

Over the years, PBR became quite popular and was widely incorporated into the entertainment industries. From movies to videogames, from ads to interior design, PBR made it possible for artists to bring their creations and their vision one step closer to reality. Today, we can say that many - if not most - algorithms used in computer animation, geometric modeling and texturing require that their results be passed through some sort of rendering process.

As PBR popularity grew, a brand new market opened up for physically-based (PB) renderers. Following the creation of *PBRT* and the publishing of *"Physically Based Rendering: From Theory to Implementation"* [**?**], several other research-oriented renderers were created. Among them is *Mitsuba*, one of the renderers chosen for this research, which places strong emphasis on experimental rendering techniques.

Following the lead of Pixar's *Renderman* [**?**], many commercial and performance-oriented renderers appeared on the market. Focused on animation techniques and visual effects for movies, these renderers provide well-established, stable rendering techniques. These renderers, such as *LuxRender* [**?**] and *Octane* [**?**], are state-of-the-art renderers used by the animation and gaming industries.

Even with different applications, the vast majority of modern PB renderers follows the same general guidelines for defining scene directives and world descriptions. Scene directives establish parameters such as which integration and sampling techniques the renderer must use, the view matrix and other camera properties. World descriptions state which objects compose the scene and which materials must be used to render them. This ensemble of descriptions is what, in PBR, is called a **scene**.

1.2 Rendering a Scene

Starting with *PBRT*, most PB renderers have used a similar, traditional structure to describe scenes.

These scenes are usually created by a 3D artist, who will use a modeling software (such as Blender, 3D Max or Maya) to draw the objects, choose their materials and then create the object files. These files will then be instanced in the scene file and interpreted by the renderer.

But even with an artist's expertise, creating scenes is still a complex process. For instance, scenes created for building overviews and interior design often compile hundreds of 3D models and dozens of customized materials and textures, as one can see in Figure 1.1. Each material and texture has to be carefully defined, taking into account the renderer's limitations and particularities.

Figure 1.1: An example of a complex scene created by Laubwerk Plants Kits

After a scene is created and rendered, all the hard work invested by the artist is stored, waiting for a possible future use. However, should the artist choose to change renderers, the scene file they created so diligently would have to be rewritten and/or heavily modified.

Converting a scene file from one renderer format to another is very difficult and time consuming. Aside from adapting material and light properties - which can be hard since sometimes renderers don't provide the same features -, the 3D object formats sup-

ported may not be the same. For instance, *Mitsuba* supports the Object File Wavefront 3D (.obj) format while *PBRT* does not.

Creating free resources in order to help rendering research, Benedikt Bitterli [**?**] manually converted 32 scenes for his renderer *Tungsten* [**?**], later converting them to *PBRT* and *Mitsuba*. According to him, *"this process is time intensive (up to several days per scene)"*.

1.3 Project Definition

Facing the issues presented in the last session, this work proposes an automatic converter between scene files of three popular PB renderers: *Mitsuba*, *PBRT* and *LuxRender*. The converter is able to interpret files from these renderers and store the information in a canonical, intermediate state. From this state, the scene structure is converted into the desired output renderer.

The converter's system architecture was conceived using modules. Each renderer scene file interpreter is a separate module that reads the input file and stores information in a canonical intermediate state. Each converter is also a module that reads from the canonical intermediate state and converts the scene structure into an output renderer file format. Adding new scene interpreters or converters is as simple as having to write a new module.

The converter is only responsible for converting renderer scene files. It cannot convert 3D object files (such as OBJ meshes to PLY meshes) or environment mapping files. If the converter recognizes an unsupported file extension during a conversion, a warning is issued to the user.

1.4 Evaluation and Results

<TODO>

2 RELATED WORKS

(https://github.com/RenderToolbox/RenderToolbox4/wiki/Overview)

3 THEORETICAL FOUNDATIONS

4 PROJECT

5 RESULT ANALYSIS

6 CONCLUSION