*Todo list

MSc in Computer Science 2017-18

Project Dissertation

Project Dissertation title: Reflectance Transformation Imaging

Term and year of submission: Trinity Term 2018

Candidate Name: Johannes Bernhard Goslar

Title of Degree the dissertation is being submitted under: MSc in Computer Science

Abstract

Vivamus vehicula leo a justo. Quisque nec augue. Morbi mauris wisi, aliquet vitae, dignissim eget, sollicitudin molestie, ligula. In dictum enim sit amet risus. Curabitur vitae velit eu diam rhoncus hendrerit. Vivamus ut elit. Praesent mattis ipsum quis turpis. Curabitur rhoncus neque eu dui. Etiam vitae magna. Nam ullamcorper. Praesent interdum bibendum magna. Quisque auctor aliquam dolor. Morbi eu lorem et est porttitor fermentum. Nunc egestas arcu at tortor varius viverra. Fusce eu nulla ut nulla interdum consectetuer. Vestibulum gravida. Morbi mattis libero sed est.

Todo Text:
Abstract

Acknowledgements

Todo Text:
Acknowledgements

Contents

List of Figures

List of Tables

1 Requirements and Design

1.1 Requirements

In preparation for the project the author had multiple informal conversations with consenting RTI users from different Oxford departments in relation to their current usage of RTI viewing software and wishes for a modernized software. One common point was the wish for a unification of the used software, most researchers had to export screenshots from the RTI software, to then add annotations and drawings in their graphical editing software. But in their graphical editing software all the advantages of an RTI image were then lost. Based on that and the discussed related work following requirements were arrived at.

The first overall goal is the main focus on web technologies. The new viewer should be available on as many platforms as possible and allow for easy distribution and usage, as well as a comforting developer experience. Based on that the importance for a web supported fileformat came next. No browser is supporting RTI or PTM files natively, so some conversion needs to take place. The requirement of an additional conversion software would be diametrical to the 'everything web based' goal though, so the browsers need to be able to also handle the conversion and then support some kind of export. The easiest way to do so was to develop a newer file format with following features:

1. Support for embedding/consuming the PTM[1] fileformat.
2. Future support for embedding/consuming the RTI[2] fileformat.
3. Extended metadata support.
4. Support for high resolutions.
5. Support for higher bitdepths per pixel than the 8 of PTM/RTI for future HDR applications.
6. Easy exchange between multiple researchers and computers.

For the viewer following features were deemed essential:

7. Compatible with all major operating systems and/or web browsers.
8. Light Controls to change the position of the virtual light source.
9. Multiple rendering modes apart from plain LRGB or RGB.
10. Quick navigation functionality.
11. Annotations to allow for further textual information relating to a part of the viewer object.
12. Paintable layers to further clarify scripture.

In addition these non-functional requirements were arrived at:

13. Free Software, the implementation should be available for everyone to change and distribute. No email-walling, instead embracing jump-in interactions between multiple parties.
14. Suitable of new developers: for scientists for research purposes, or students for educational purposes.
15. Plugin support to allow independent additions or modifications to the core software.
16. Good developer experience.
17. Adequate performance, at least keeping up with current implementations.
18. Easy installation for researchers.
19. Fast responses to user interactions.
20. Reasonable file sizes for instant transfer/viewing.
21. Preservable software and BTF files.

1.2 Fileformat

Though the LoC deems the PTM format “”relatively transparent. It can be viewed in plain text editors. [. . .] A very simple program would present these numbers in a human-readable form.”[1] the author disagrees with this judgment.

1.3 Workflow

Todo Diagramm:
Workflow comparison

json

Todo Text:
File import/export

1.4 Architecture

1.5 State-Driven

Todo Text:
State-Driven

1.6 Plugins

Todo Text:
Plugins

1.7 Rendering Stack

Todo Text:
Rendering Stack

This is the specification of the BTF fileformat, as of version 1.0 on August 27, 2018. It was developed co-supervisor Stefano Gogioso, with input from and extension by the author of the enclosing thesis in relation to the oxrti viewer.

A BTF File Format

This section describes the BTF file format. The aim of this file format is to provide a generic container for BTF data to be specified using a variety of common formats. Files shall have the `.btf.zip` extension.

A.1 File Structure

A BTF file is a ZIP file containing the following:

- A **manifest** file in JSON format, named `manifest.json`. The manifest contains all information about the BRDF/BSDF model being used, including the names for the available **channels** (e.g. `R`, `G` and `B` for the 3-channel RGB), the names of the necessary **coefficients** (e.g. bi-quadratic coefficients) and the **image file format** for each channel.
- A single folder named `data`, with sub-folders having names in 1-to-1 correspondence with the channels specified in the manifest.
- Within each channel folder, greyscale image files having names in 1-to-1 correspondence with the coefficients specified in the manifest, each in the image file format specified in the manifest for the corresponding channel. For example, if one is working with RGB format (3-channels named `R`, `G` and `B`) in the PTM model (five coefficients `a2`, `b2`, `a1`, `b1` and `c`,specifying a bi-quadratic) using 16-bit greyscale bitmaps, the file `/data/B/a2.bmp` is the texture encoding the `a2` coefficient for the blue channel of each point in texture space.
- The datafiles are all in reversed scanline order (meaning from bottom to top), to keep aligned with the original PTM format and allow easier loading into WebGL.

In case of usage with the oxrti viewer, following files can be present in addition to those mentioned above:

A.2 Manifest

The manifest for the BTF file format is a JSON file with root dictionary. The `root` element has two mandatory child elements: one named `data`, and one named `name` with the option of additional child elements (with different names) left open to future extensions of the format.

- The `name` element is a string with a name of the contained object.
- The `data` element has for entries, named `width`, `height`, `channels` and `channel-model`. The `width` and `height` attributes have values in the positive integers describing the dimensions of the BTDF. The `channel-model` attribute has value a non-empty alphanumeric string uniquely identifying the BRDF/BSDF colour model used by the BTF file (see Options section below). The `channels` element has an arbitrary amout of named `channel` entries, according to the `channel-model`.
- Additionally the `data` element has one untyped entry named `formatExtra`, where format implementation specific data can be stored.
- Each `channel` has an `coefficents` child consisting of an arbitrary number of `coefficent` entries, as well as one `coefficient-model` attribute. The `coefficient-model` attribute has value a non-empty alphanumeric string uniquely identifying the BRDF/BSDF approximation model used by the BTF file (see Options section below).
- Each `coefficient` element has one attribute: `format`. The `format` attribute has value a non-empty alphanumeric string uniquely identifying the image file format used to store the channel values (see Options section below).

A.3 Textures

Each image file `/data/CHAN/COEFF.EXT` has the same dimensions specified by the `width` and `height` attributes of the `data` element in the manifest, and is encoded in the greyscale image file format specified by the `format` attribute of the unique `coefficient` element with attribute `name` taking the value `COEFF` (the extension `.EXT` is ignored). The colour value of a pixel `(u,v)` in the image is the value for coefficient `COEFF` of channel `CHAN` in the BRDF/BSDF for point `(u,v)`, according to the model jointly specified by the values of the attribute `model` for element `channels` (colour model) and the attribute `model` for element `coefficients` (approximation model).

A.4 Options

At present, the following values are defined for attribute `channel-model` of element `channels`.

- `RGB`: the 3-channel RGB colour model, with channels named `R`, `G` and `B`. This colour model is currently under implementation.
- `LRGB`: the 4-channel LRGB colour model, with channels named `L`, `R`, `G` and `B`. This colour model is currently under implementation.
- `SPECTRAL`: the spectral radiance model, with an arbitrary non-zero number of channels named either all by wavelength (format `---nm`, with `---` an arbitrary non-zero number) or all by frequency format `---Hz`, with `---` an arbitrary non-zero number. This colour model is planned for future implementation.

At present, the following values are defined for attribute `model` of element coefficients, where the ending character `*` is to be replaced by an arbitrary number greater than or equal to 1.

- `flat`: flat approximation model (no dependence on light position). This approximation model is currently under implementation.
- `RTIpoly*`: order-`*` polynomial approximation model for RTI (single view-point BRDF). This approximation model is currently under implementation.
- `RTIharmonic*`: order-`*` hemispherical harmonic approximation model for RTI (single view-point BRDF). This approximation model is currently under implementation.
- `BRDFpoly*`: order-`*` polynomial approximation model for BRDFs. This approximation model is planned for future implementation.
- `BRDFharmonic*`: order-`*` hemispherical harmonic approximation model BRDFs. This approximation model is planned for future implementation.
- `BSDFpoly*`: order-`*` polynomial approximation model for BSDFs. This approximation model is planned for future implementation.
- `BSDFharmonic*`: order-`*` spherical harmonic approximation model for BSDFs. This approximation model is planned for future implementation.

At present, the following values are defined for attribute `format` of elements tagged `coefficient`, where the ending character `*` is the bit-depth, to be replaced by an allowed positive multiple of 8.

- `BMP*`: greyscale BMP file format with the specified bit-depth (8, 16, 24 or 32). Support for this format is currently under implementation.

- `PNG*`: PNG file format encoding the specified bit-depth (8, 16, 24, 32, 48 or 64). Support for this format is currently under implementation. Different PNG colour options are used to support different bit-depths:
- `Grayscale` with 8-bit/channel to encode 8-bit bit-depth.
- `Grayscale` with 16-bit/channel to encode 16-bit bit-depth.
- `Truecolor` with 8-bit/channel to encode 24-bit bit-depth.
- `Truecolor and alpha` with 8-bit/channel to encode 32-bit bit-depth.
- `Truecolor` with 16-bit/channel to encode 48-bit bit-depth.
- `Truecolor and alpha` with 16-bit/channel to encode 64-bit bit-depth.

References

[1] Library of Congress. *Polynomial Texture Map (PTM) File Format.* June 14, 2018. URL: `https://www.loc.gov/preservation/digital/formats//fdd/fdd000487.shtml` (visited on 08/10/2018).

[2] Library of Congress. *Reflectance Transformation Imaging (RTI) File Format.* June 9, 2018. URL: `https://www.loc.gov/preservation/digital/formats//fdd/fdd000486.shtml#notes` (visited on 08/10/2018).