Hochschule Hamm-Lippstadt Wavefront Loader

Specification

Goals

This document describes the OBJLoader Java library. It is dedicated to developers, users and stakeholders of the library.

Scope

The OBJLoader is a Java library which enables loading the contents of OBJ and MTL wavefront files. The library must deliver the data in a format suited for usage in JOGL. The Hochschule Hamm-Lippstadt provides this library to its students in order to help them building their JOGL projects. The library must be compatible with Java version 11. It must not directly depend on any JOGL version, but instead prepare the data for OpenGL in a generic way.

The following transcripts of the wavefront specifications will be used: paulbourke.net/dataformats/obj and paulbourke.net/dataformats/mtl.

Overview

The library must provide functionality to load geometry and material data from wavefront OBJ and MTL files.

Supported geometry includes three-dimensional polygonal vertex positions, three-dimensional polygonal surface normals, two-dimensional polygonal texture coordinates, but none of the free-form curve geometry data. Loading structures from those files, such as vertex objects or vertex groups, must be supported. The geometry must be loaded into arrays of primitives for simple usage in JOGL (Java Binding for the OpenGL API). Material properties must be loaded into objects specifically constructed to hold wavefront material data. Texture pixel data does not have to be loaded, but only the paths to the texture files must be extracted.

Loading a wavefront file should require minimal source code, but should be transparent at the same time. The behaviour of the loader does never depend on the contents of the loaded file, but can always immediately be determined by looking at the configuration of the library inside the source code. This way, the systems behaviour is kept predictable, and does not change across different wavefront files.

The library should be extensible to allow users to implement currently unsupported wavefront features themselves. The library should utilize semantic versioning to convey version information.

Detailed Requirements

Functional Requirements

1. The library must be usable in a project which uses AdoptOpenJDK Version 8 or 11
2. The library must not include the JOGL library in its production dependencies
   (The library may include the JOGL library in its testing dependencies)

3. The library must support the following wavefront OBJ data, as described in the official wavefront OBJ specification:
   - 3.1. Vertex Positions, Vertex Surface Normals, Vertex Texture Coordinates, Polygonal Faces
   - 3.2. Named Objects, Named Groups
   - 3.3. Material Libraries

4. The library must support the following wavefront MTL data, as described in the official wavefront MTL specification:
   - 4.1. Name, Mip Map Anti Aliasing, Opacity Halo
   - 4.2. Colors: Ambient, Diffuse, Specular, Transmission Filter, Emission
   - 4.3. Numerical Values: Opacity, Specular Exponent, Sharpness, Refraction Index, Roughness, Metallic, Sheen, Clearcoat Thickness, Clearcoat Roughness, Anisotropy, Anisotropy Rotation
   - 4.4. Textures: Ambient, Diffuse, Specular, Specular Exponent, Opacity, Decal, Reflection, Bump, Displacement, Normal, Emission, Sheen, Metallic, Roughness

5. The library should ignore the following geometrical wavefront data and (if not disabled) print a warning to the console, if those are encountered while loading:
   - 5.1. Smoothing groups
   - 5.2. Edge Beveling
   - 5.3. Color interpolation configuration
   - 5.4. Dissolve interpolation configuration
   - 5.5. Level Of Detail information
   - 5.6. Shadow Object declarations
   - 5.7. Raytracing Object declarations

6. The library should support any of the following data and raise an Exception if those are encountered while loading:
   - 6.1. Any free-form or curve data, as specified in the wavefront file format, including Basis Matrices, Step Size, and Trimming Loops
   - 6.2. Inofficial non-standard conventions, including alternative naming of properties
   - 6.3. Linked OBJ files, as specified in the wavefront file format
   - 6.4. Executing unix commands, as specified in the wavefront file format

7. The library must be configurable, respecting the following options:
    7.1. If surface normals should be loaded or ignored
    7.2. If texture coordinates should be loaded or ignored
    7.3. If geometry should be compressed using vertex indexing
    or should be put into only a vertex array
    7.4. If materials should be loaded or ignored
    7.5. If all geometry should be merged into a single array
    or divided according to the structures in the wavefront file
    7.6. If wavefront groups should be treated as objects or should be ignored
    7.7. If warnings any should be printed during the loading and conversion process
    7.8. If the third texture coordinate should be ignored

8. The library musst support reading data from any Java InputStream
9. The library must simplify reading data from files
    which have been packed into a jar file of an IntelliJ project

10. The library must prepare data for usage in JOGL
    - 10.1. The library must convert all polygons to triangles
    - 10.2. The library must support loading all geometry data into a single vertex array
    - 10.3. The library must support loading geometry into a vertex array and constructing a separate index array (indexed vertex buffers)
    - 10.4. The library must support loading geometry into the following structure:
        - 10.4.1. An array of floats containing positional data

```
[
    P0.x, P0.y, P0.z,
    P1.x, P1.y, P1.z,
    …
]
```

        - 10.4.2. An array of floats containing positional and normal data

```
[
    P0.x, P0.y, P0.z,   N0.x, N0.y, N0.z,
    P1.x, P1.y, P1.z,   N1.x, N1.y, N1.z,
    …
]
```

        - 10.4.3. An array of floats containing positional and texture data

```
[
    P0.x, P0.y, P0.z,   T0.x, T0.y,
    P1.x, P1.y, P1.z,   T1.x, T1.y,
    …
]
```

        - 10.4.4. An array of floats containing positional, normal, and texture data

```
[
    P0.x, P0.y, P0.z,   T0.x, T0.y,   N0.x, N0.y, N0.z,
    P1.x, P1.y, P1.z,   T1.x, T1.y,   N1.x, N1.y, N1.z,
    …
]
```

Non-Functional Requirements

11. The library must not change behaviour depending on the input file, but be predicable just by looking at the explicit configuration in Java
12. The library should utilize the builder-pattern to enable simple configuration
13. Loading a wavefront file and configuring the loader should not need more than 800 characters (roughly 8 lines) of source code.