
libgltf

Alex Chi

Oct 15, 2020

CONTENTS

- 1 Features 3**
- 2 Getting Started 5**
- 3 Usage 7**
 - 3.1 How to use 7
 - 3.2 Advance 8
- 4 Donation 9**
- 5 License 11**

This project was generated by glTF 2.0 JSON schema and support to load the glTF 2.0 file to a struct *SGITF*.
It was used in [glTFForUE4](#).

FEATURES

- glTF 2.0
- Load the gltf/embedded/glb file
- This is a static library
- Cross platform
- C++11
- Supports the Unicode and UTF8
- Supports the *KHR_draco_mesh_compression* - [Google's Draco](#)
- Supports the *KHR_lights_punctual*
- Platforms * Windows (Win32 and x64) * Linux * macOS * Android (armeabi-v7a, armeabi-v7a-with-neon, arm64-v8a, x86 and x86_64) * iOS (simulator, iOS, tvOS and watchOS)

GETTING STARTED

1. Update the submodule

Run `git submodule update --init`

2. Generate the project by [CMake]

Run `cmake -G "[GENERATOR BY YOUR SYSTEM]" [LIBGLTF FOLDER]`

3. Build the project and generate the the static library *libgltf.lib* or *libgltf.a*

4. Include *libgltf/libgltf.h* in your project.

5. Link the static library *libgltf.lib* or *libgltf.a* in your project.

You have to link the static library *draco.lib* or *draco.a* with your project, if you want support the [Google's Draco](#). And you can find the draco in the external folder.

Code example:

```
std::shared_ptr<libgltf::IGlTFLoader> gltf_loader = libgltf::IGlTFLoader::Create(  
    ↪ *your gltf file*);  
gltf_loader->Execute();  
std::shared_ptr<libgltf::SGlTF> loaded_gltf = gltf_loader->glTF().lock();  
if (!loaded_gltf)  
{  
    printf("failed to load your gltf file");  
}
```


Generate the *makefile* or *ninja* or *visual c++ project* or *xcode project* by [CMake].

For now, just build to a static library - *libgltf.(lib/a)*.

3.1 How to use

3.1.1 Load the glTF file

You can load the glTF file by the function - *libgltf::IglTFLoader::Create*, like this:

```
std::shared_ptr<libgltf::IglTFLoader> gltf_loader = libgltf::IglTFLoader::Create(
    ↪ "Monster.glTF");
std::shared_ptr<libgltf::SGlTF> loaded_gltf = gltf_loader->glTF().lock();
if (!loaded_gltf)
{
    // the glTF file is valid
    return false;
}
```

3.1.2 Load the mesh data

And get the mesh data, like this:

```
// get all indices of the triangle
libgltf::TDimensionVector<1, size_t> triangle_data;
std::shared_ptr<libgltf::TAccessorStream<libgltf::TDimensionVector<1, size_t> > >_
    ↪ triangle_stream = std::make_shared<libgltf::TAccessorStream
    ↪ <libgltf::TDimensionVector<1, size_t> > >(triangle_data);
gltf_loader->GetOrLoadMeshPrimitiveIndicesData(0, 0, triangle_stream);

// get all points of the triangle
libgltf::TDimensionVector<3, float> position_data;
std::shared_ptr<libgltf::TAccessorStream<libgltf::TDimensionVector<3, float> > >_
    ↪ position_stream = std::make_shared<libgltf::TAccessorStream
    ↪ <libgltf::TDimensionVector<3, float> > >(position_data);
gltf_loader->GetOrLoadMeshPrimitiveAttributeData(0, 0, L"position", position_stream);
```

3.1.3 Load the image data

You can get the image (data and type) by `libglTF::IglTFLoader::GetOrLoadImageData`, like this:

```
std::vector<uint8_t> image0_data;  
libglTF::string_t image0_data_type;  
glTF_loader->GetOrLoadImageData(0, image0_data, image0_data_type);
```

3.2 Advance

3.2.1 Regenerate new code by the glTF schema

You can update the c++11 source code by `jsonschematoc11`.

Generate the c++11 code:

1. Run `tools/batch/update_parser_by_scheme.bat` (Windows) or `tools/batch/update_parser_by_scheme.sh` (Unix/Linux/MacOS)
2. Build your version by `CMake`, `Ninja` or `VisualStudio`.

3.2.2 Character encoding

- default using UTF8, char and std::string
- set `LIBGLTF_CHARACTER_ENCODING` in cmake command - UTF8, UTF16, UTF32 or UNICODE

3.2.3 Supports Google's draco

You can update the Google's draco submodule in `external/draco` or pull the draco repo by yourself.

Check the `LIBGLTF_USE_GOOGLE_DRACO` or set `LIBGLTF_USE_GOOGLE_DRACO` is `TRUE`.

- Set the `GOOGLE_DRACO_PATH_INCLUDE`, `GOOGLE_DRACO_PATH_BUILD`, `GOOGLE_DRACO_LIBRARY_DRACODEC_DEBUG`, `GOOGLE_DRACO_LIBRARY_DRACODEC_RELEASE`, `GOOGLE_DRACO_LIBRARY_DRACOENC_DEBUG` and `GOOGLE_DRACO_LIBRARY_DRACOENC_RELEASE`.
- Or enable the `LIBGLTF_USE_GOOGLE_DRACO_SUBMODULE` to compile with the submodule - `external/draco`.

3.2.4 Download the compiled library

This project is compiled by github action, and you can download the compiled library with Google's Draco from the [action page](#) or the [release page](#).

DONATION

Please consider donating to sustain my activities

CHAPTER
FIVE

LICENSE

The MIT license.