libgltf

Xing Ji

Mar 02, 2021

CONTENTS

This project was generated by glTF 2.0 JSON schema and support to load the glTF 2.0 file to a struct *SGlTF*.

It was used in glTFForUE4.

CHAPTER

ONE

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 some extensions**
  - *KHR_draco_mesh_compression* - Google's Draco
  - *KHR_lights_punctual*
  - *KHR_materials_pbrSpecularGlossiness*
  - *KHR_materials_clearcoat*
  - and more
- **Platforms**
  - **Windows**
    - Win32 (win32)
    - x64 (win64)
  - Linux (linux)
  - macOS (macos)
  - **Android**
    - armeabi-v7a
    - armeabi-v7a-with-neon
    - arm64-v8a
    - x86
    - x86_64
  - **iOS**
    - iOS (iphoneos)
    - watchOS (watchos)

* simulator

CHAPTER

TWO

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 static library `libgltf.(lib/a)`

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

5. Link the static library `libgltf.(lib/a)` in your project.

   You have to link the static library *draco.lib* or *draco.a* with your project, if you want to 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");
}
```

CHAPTER

THREE

USAGE

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

It is 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

Generate the c++11 code:

- You can update the c++11 source code by `jsonschematoc11`.
- You need update the submodule `external/glTF`
- It runs in **Python2**

1. Run `update_parser_by_scheme.bat`
   - For Windows: `cd tools\batch\ && update_parser_by_scheme.bat && cd ..\..\`
   - For Linux/MacOS `cd tools/batch/ && ./update_parser_by_scheme.sh && cd ../../`
2. Build your version by CMake.

3.2.2 Character encoding

- You can set `LIBGLTF_CHARACTOR_ENCODING` in CMake to set the encoding.
- Supports `UTF8`, `UTF16`, `UTF32` or `UNICODE`.
- The default encoding is `UTF8`.

3.2.3 Supports Google's draco

You can update 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 libraries

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.

> In the action page or the release page, libraries was compiled with *MultiThreading* (/MT or /MTd) for **windows**.

CHAPTER

FOUR

DONATION

Please consider donating to sustain my activities

CHAPTER

FIVE

LICENSE

The MIT license.