glTF-Tutorials

Table of Contents | Next: Basic glTF Structure

Introduction to glTF using WebGL

An increasing number of applications and services are based on 3D content. Online shops offer product configurators with a 3D preview. Museums digitize their artifacts with 3D scans and allow visitors to explore their collections in virtual galleries. City planners use 3D city models for planning and information visualization. Educators create interactive, animated 3D models of the human body. Many of these applications run directly in the web browser, which is possible because all modern browsers support efficient rendering with WebGL.

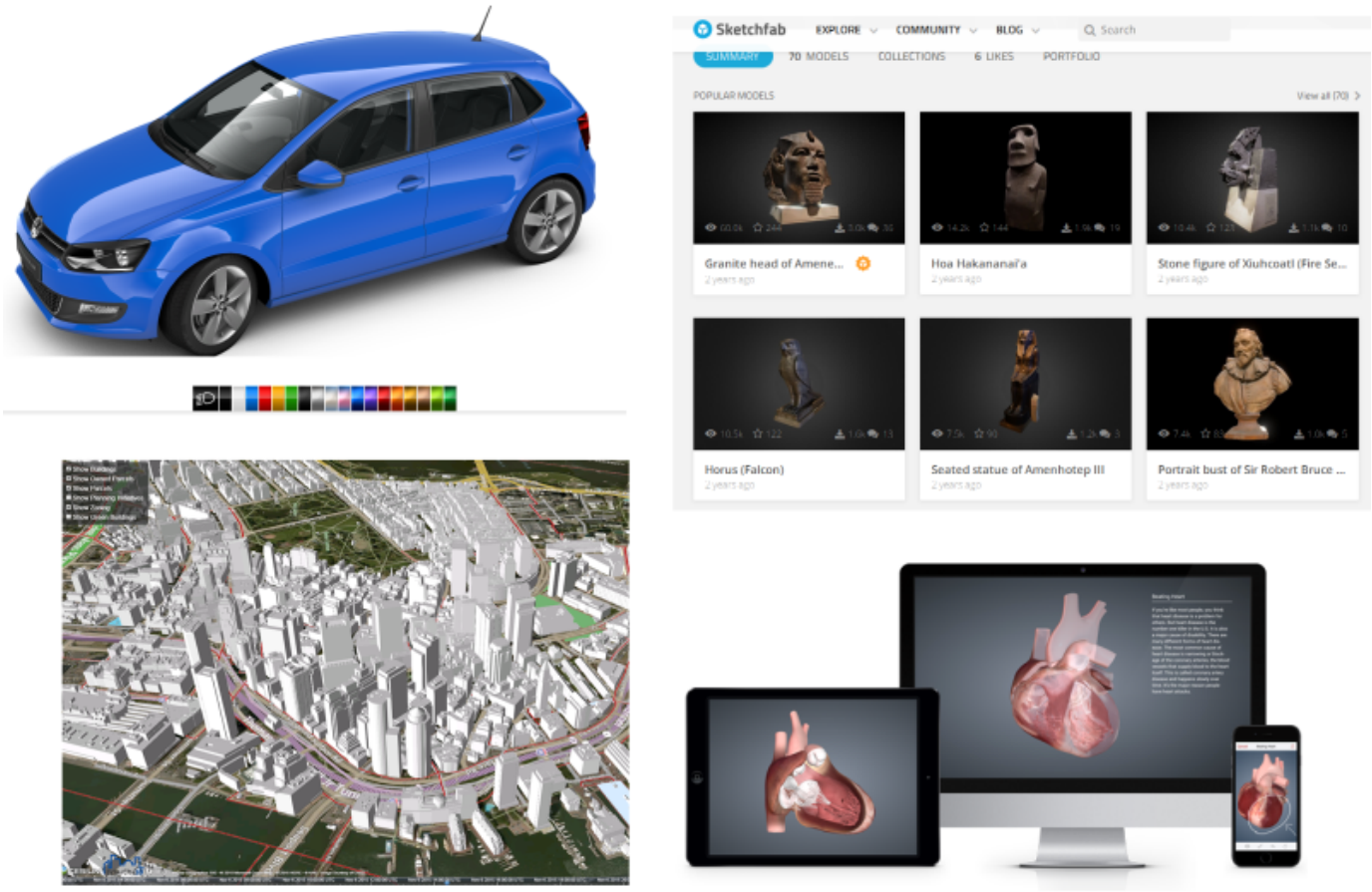

Image 1a: Screenshots of various websites and applications showing 3D models.

Demand for 3D content in various applications is constantly increasing. In many cases, the 3D content has to be transferred over the web, and it has to be rendered efficiently on the client side. But until now, there has been a gap between the 3D content creation and efficient rendering of that 3D content in the runtime applications.

3D content pipelines

3D content that is rendered in client applications comes from different sources and is stored in different file formats. The list of 3D graphics file formats on Wikipedia shows an overwhelming number, with more than 70 different file formats for 3D data, serving different purposes and application cases.

For example, raw 3D data may be obtained with a 3D scanner. These scanners usually provide the geometry data of a single object, which is stored in OBJ, PLY, or STL files. These file formats do not contain information about the scene structure or how the objects should be rendered.

More sophisticated 3D scenes can be created with authoring tools. These tools allow one to edit the structure of the scene, the light setup, cameras, animations, and, of course, the 3D geometry of the objects that appear in the scene. Applications store this information in their own, custom file formats. For example, Blender stores the scenes in `.blend` files, LightWave3D uses the `.lws` file format, 3ds Max uses the `.max` file format, and Maya uses `.ma` files.

In order to render such 3D content, the runtime application must be able to read different input file formats. The scene structure has to be parsed, and the 3D geometry data has to be converted into the format required by the graphics API. The 3D data has to be transferred to the graphics card memory, and then the rendering process can be described with sequences of graphics API calls. Thus, each runtime application has to create importers, loaders, or converters for all file formats that it will support, as shown in Image 1b.

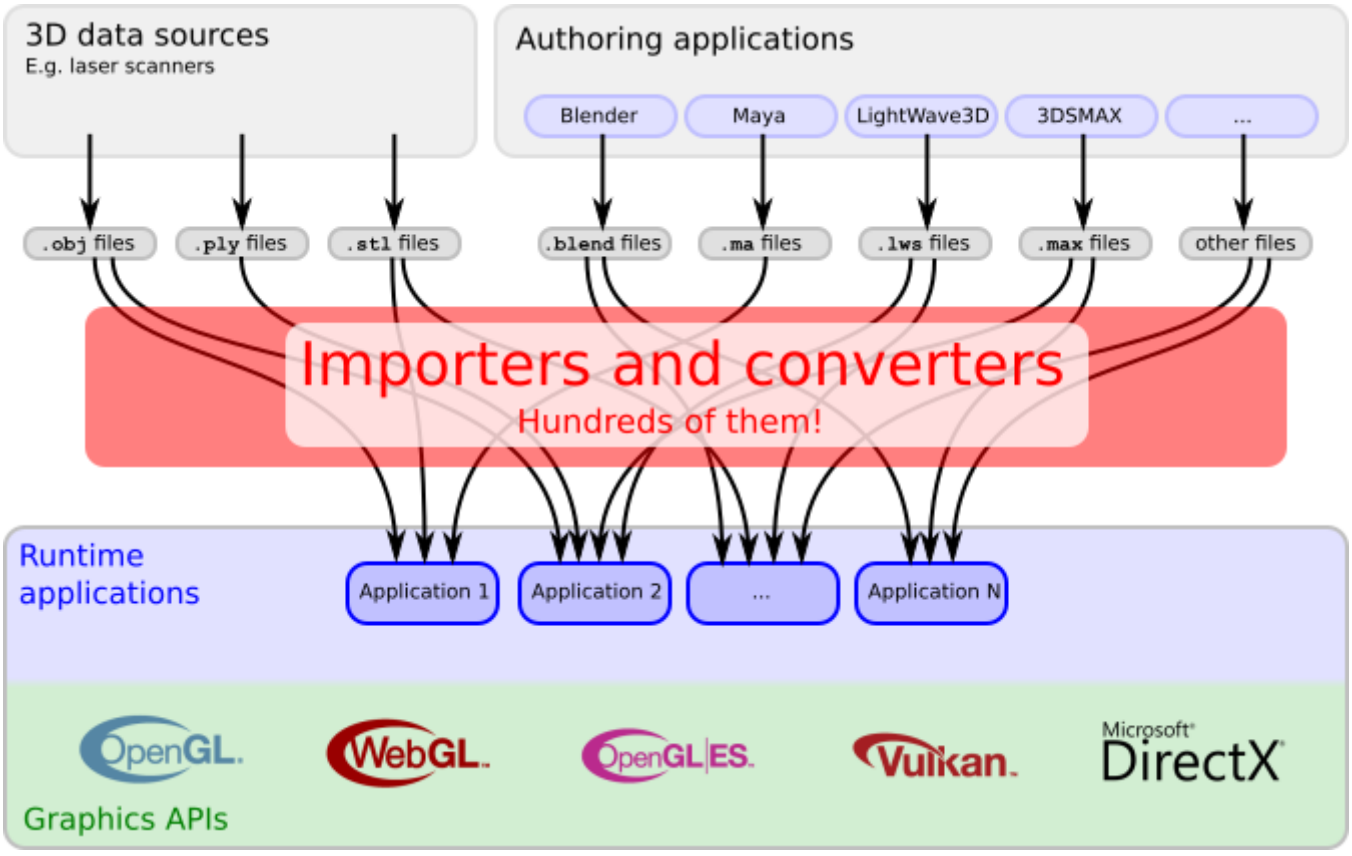

Image 1b: The 3D content pipeline today.

glTF: A transmission format for 3D scenes

The goal of glTF is to define a standard for representing 3D content, in a form that is suitable for use in runtime applications. The existing file formats are not appropriate for this use case: some of do not contain any scene information, but only geometry data; others have been designed for exchanging data between authoring applications, and their main goal is to retain as much information about the 3D scene as possible, resulting in files that are usually large, complex, and hard to parse. Additionally, the geometry data may have to be preprocessed so that it can be rendered with the client application.

None of the existing file formats were designed for the use case of efficiently transferring 3D scenes over the web and rendering them as efficiently as possible. But glTF is not "yet another file format." It is the definition of a *transmission* format for 3D scenes:

- The scene structure is described with JSON, which is very compact and can easily be parsed.
- The 3D data of the objects are stored in a form that can be directly used by the common graphics APIs, so there is no overhead for decoding or pre-processing the 3D data.

Different content creation tools may now provide 3D content in the glTF format. And an increasing number of client applications are able to consume and render glTF. Some of these applications are shown in Image 1a. So glTF may help to bridge the gap between content creation and rendering, as shown in Image 1c.

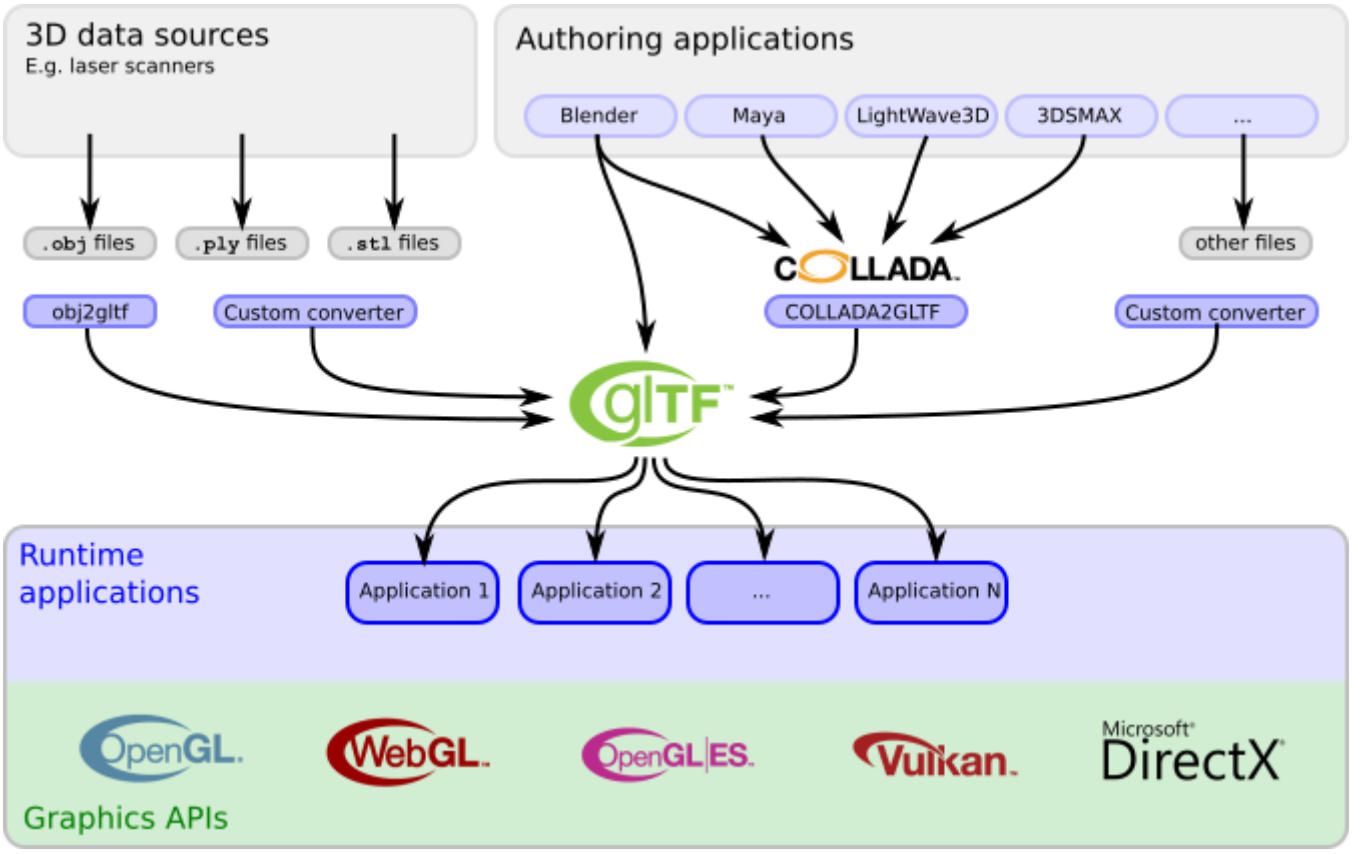

Image 1c: The 3D content pipeline with glTF.

An increasing number of content creation tools provide glTF import and export directly. For example, the Blender Manual documents how to import and export PBR materials using glTF. Alternatively, other file formats can be used to create glTF assets, using one of the open-source conversion utilities listed in the glTF Project Explorer. The output of converters and exporters can be validated using the Khronos glTF Validator.

Table of Contents	Next: Basic glTF Structure
