Manual: Piglet glTF Importer 1.0.2

- Online documentation: https://awesomesaucelabs.github.io/piglet-manual/
- Support e-mail: awesomesaucelabs@gmail.com

Table of Contents

- Introduction
- Features
- Caveats
- Setup
- Editor Imports
  - Piglet Options Window
- Runtime Imports
  - Runtime Import Tutorial
  - Runtime Import API
- Sample Application: PigletViewer
- Footnotes

Introduction

Piglet is a Unity asset that allows you to import 3D models from glTF files, both in the Editor and at runtime. This provides Unity developers access to a large collection of free textures, materials, and models from sites like Sketchfab and Google Poly[1].

Visit the Web Demo[2] to try Piglet before you buy it.

Features

- import glTF models in the Editor or at runtime
- import glTF models from `.gltf`, `.glb`, or `.zip` files, using file paths or HTTP URLs
- extract textures and materials from glTF models, for use with your own 3D models
- tested with glTF models from Sketchfab, Google Poly[1], and Blender
- supported platforms: Windows, Android, WebGL (Unity 2018.4 or newer)
- full source code provided

Caveats

- **Piglet does not import animations (yet!)**. However, glTF models containing animation data can still be imported as static models. In addition, any skinning data or blendshape data (a.k.a. "morph targets") will be correctly imported alongside the model(s).
- **Runtime imports may stall the main Unity thread**. While I have attempted to minimize interruptions to the main Unity thread during runtime imports, I cannot provide any hard guarantees about this yet. Unity requires certain operations (e.g. mesh creation) to be performed on the main Unity thread, and so it is possible for runtime imports of large/complex models to cause "hiccups" during game execution.

Setup

To set up Piglet in your project, purchase and install Piglet from the Unity Asset Store page. Piglet works with Unity 2018.4 or later, and does not require installation of any third-party dependencies. If you wish to try your own glTF models with Piglet prior to purchasing the asset, please see the Piglet WebGL Demo.

Piglet bundles the following libraries:

Library	Author	License	Path
Json.NET	Newtonsoft	MIT License	`Assets/Piglet/Dependencies/Json.NET`
SharpZipLib	icsharpcode@github	MIT License	`Assets/Piglet/Dependencies/SharpZipLib`
UnityGLTF	Khronos/Sketchfab	MIT License	`Assets/Piglet/Dependencies/UnityGLTF`

Note: If your Unity project already includes one of these libraries (e.g. Json.NET), you may get errors in the Unity Console due to duplicate function/class definitions. In most cases, removing Piglet's copy of the library under `Assets/Piglet/Dependencies` should solve the issue.

Editor Imports

Note: For a video demonstration of Piglet Editor imports, see https://youtu.be/wf26w0gcVcA.

Once you have installed Piglet from the Unity Asset Store, you can import glTF models into your Unity project by dragging-and-dropping `.gltf/.glb/.zip` files from a file browser window (e.g. Windows File Explorer) to a folder inside the Unity Project Browser (Figure 1). Any folder

under `Assets` can be used as the drop target, including the `Assets` directory itself.

Importing a glTF file in the Editor produces a Unity prefab for the model, which can then be dragged into your Unity scenes as desired. Piglet places the generated prefab and any dependent asset files (e.g. textures, materials, meshes) under a newly-created subfolder named after the input `.gltf/.glb/.zip` file.

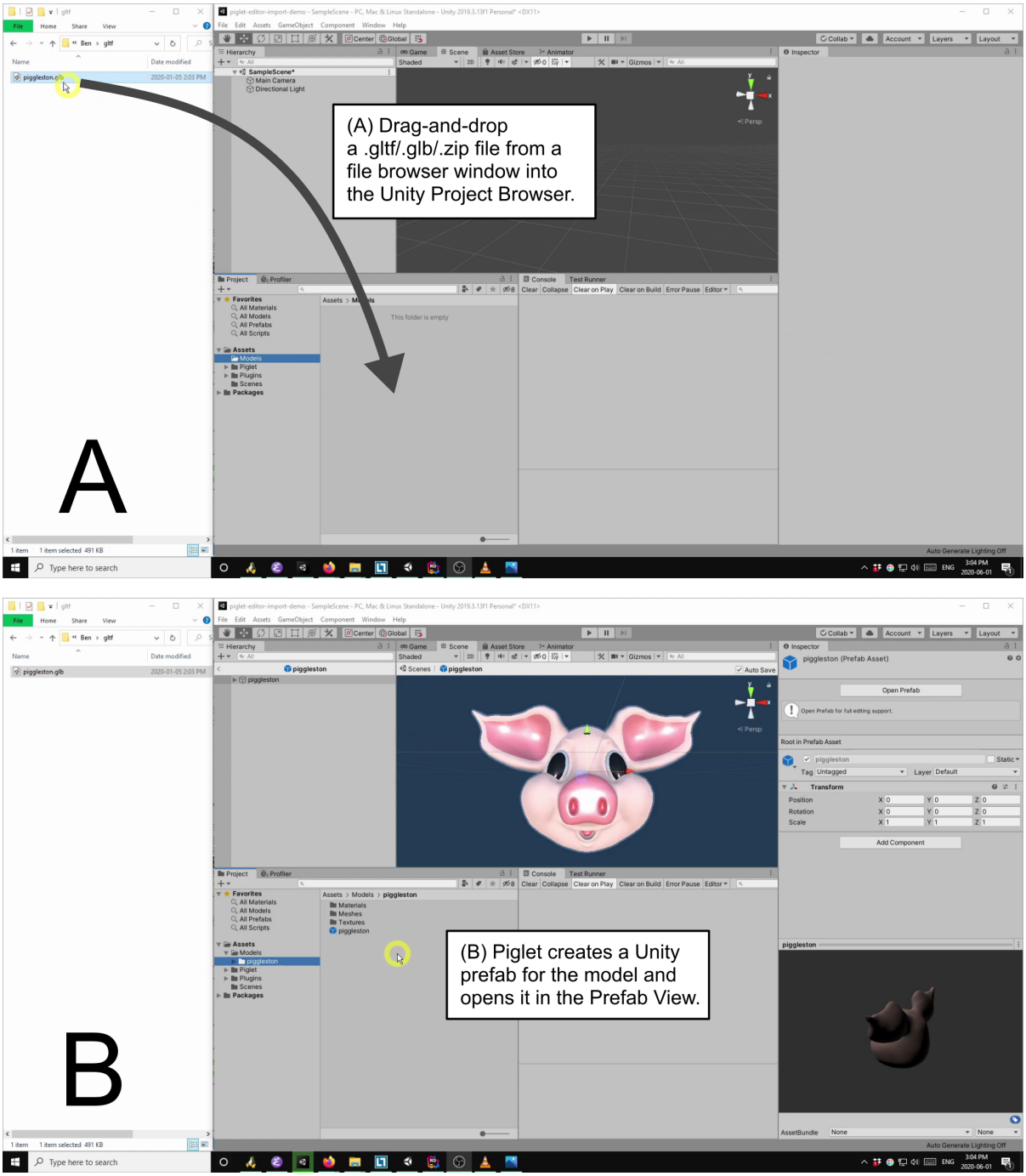


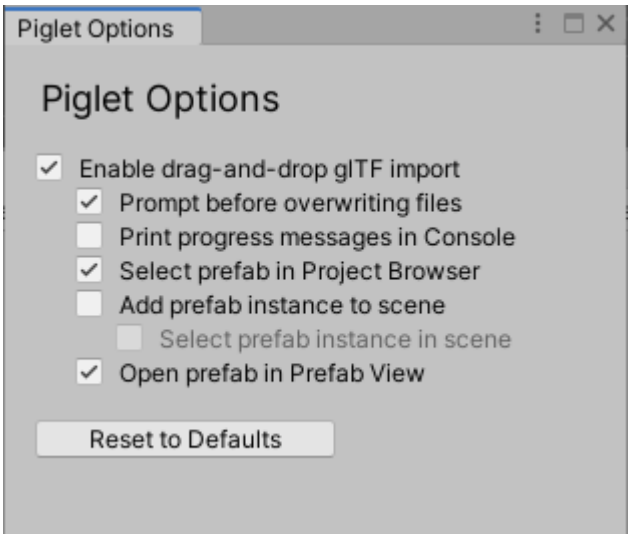
Figure 1: Importing a glTF model in the Editor. (A) The user drags-and-drops a .gltf/.glb/.zip file from Windows File Explorer to the Unity Project Browser window. (B) Piglet creates a Unity prefab for the model and opens it in the Prefab View.

There may be circumstances where you want copy a `.gltf/.glb/.zip` file into your project without automatically converting it to a Unity prefab. You can bypass/disable Piglet's default drag-and-drop behaviour by any of the following methods:

- Hold down the `Control` or `Command` key while dragging-and-dropping the `.gltf/.glb/.zip` into the Unity Project Browser.
- Uncheck `Enable drag-and-drop glTF import` in the Piglet Options window, located under `Window => Piglet Options` in the Unity menu.
- Copy the `.gltf/.glb/.zip` file into the Unity project directory from a file browser window or on the command line. (In other words, copy the file "behind Unity's back".)

Piglet Options Window

A number of options relating Editor imports can be set in the Piglet Options window, located under `Window => Piglet Options` in the Unity menu.



Option	Description
`Enable drag-and-drop glTF import`	Enable/disable automatic glTF imports when dragging .gltf/.glb/.zip files onto the Project Browser window
`Prompt before overwriting files`	Show confirmation prompt if glTF import directory already exists
`Print progress messages in Console`	Log progress messages to Unity Console window during glTF imports (useful for debugging)
`Select prefab in Project Browser`	After a glTF import has completed, select/highlight the generated prefab in the Project Browser window

`Add prefab instance to scene`	After a glTF import has completed, add the generated prefab to the current Unity scene, as a child of the currently selected game object. If no game object is selected in the scene, add the prefab at the root of the scene instead.
`Select prefab instance in scene`	Select/highlight the prefab in the scene hierarchy after adding it to the scene
`Open prefab in Prefab View`	After a glTF import has completed, open the generated prefab in the Prefab View. (This is equivalent to double-clicking the prefab in the Project Browser.)

Runtime Imports

Piglet can import a glTF model at runtime from a file path, an HTTP URL, or the raw byte content of a .gltf/.glb/.zip file (`byte[]`). Runtime imports are performed incrementally, with minimum possible interruption to the main Unity thread.

Runtime Import Tutorial

This section walks through example code for importing a glTF model at runtime. The example code presented in this section is included with Piglet under `Assets/Piglet/Examples/RuntimeImport`, and a video version of this tutorial is available online at https://youtu.be/f66wmgSTPI0.

As our example glTF model, we will use the `.glb` file for Sir Piggleston (the Piglet mascot), which may be downloaded from https://awesomesaucelabs.github.io/piglet-webgl-demo/StreamingAssets/piggleston.glb. The minimal code to import the model at runtime is as follows:

```
using Piglet;
using UnityEngine;

/// <summary>
/// This MonoBehaviour provides a minimal example for using
/// Piglet to import glTF models at runtime.
/// </summary>
public class RuntimeImportBehaviour : MonoBehaviour
{
    /// <summary>
    /// The currently running glTF import task.
    /// </summary>
    private GltfImportTask _task;

    /// <summary>
    /// Unity callback that is invoked before the first frame.
```

```
    /// Create the glTF import task.
    /// </summary>
    void Start()
    {
        _task = RuntimeGltfImporter.GetImportTask(
             "https://awesomesaucelabs.github.io/piglet-webgl-demo/StreamingAssets/piggleston.glb");
    }

    /// <summary>
    /// Unity callback that is invoked after every frame.
    /// Here we call MoveNext() to advance execution
    /// of the glTF import task.
    /// </summary>
    void Update()
    {
        // advance execution of glTF import task
        _task.MoveNext();
    }
}
```

Figure 2: Minimal code to import a glTF file at runtime.

As shown in Figure 2, a runtime glTF import happens in two parts. First, we create an import task by calling `RuntimeGltfImporter.GetImportTask`, passing in the URL of the glTF model as a parameter. Second, we advance the execution of the import task by repeatedly calling `MoveNext()` on the task. A convenient place to call `MoveNext()` is in the `Update()` method, which is called by Unity once per frame. Continuing to call `MoveNext()` after the import has completed does no harm.

Attaching the script from Figure 2 to any game object in your Unity scene is sufficient to import a glTF model at runtime. However, in a real game/application, you will probably want tighter integration between your own code and the importer. For example, you may want to show progress messages while the model is loading, or to attach custom MonoBehaviours to the model once it has loaded. To achieve these types of behaviours, `GltfImportTask` provides callback hooks for: progress messages (`OnProgress`), user cancelation (`OnAborted`), import errors (`OnException`), and successful completion (`OnCompleted`).

As a first example of callback usage, we'll extend the example script from Figure 2 to print progress messages during the glTF import. We can achieve this by assigning a custom method to the `OnProgress` callback for the import task, as shown in Figure 3.

```
using Piglet;
using UnityEngine;

/// <summary>
/// This MonoBehaviour provides a minimal example for using
/// Piglet to import glTF models at runtime.
/// </summary>
```

```
public class RuntimeImportBehaviour : MonoBehaviour
{
    /// <summary>
    /// The currently running glTF import task.
    /// </summary>
    private GltfImportTask _task;

    /// <summary>
    /// Unity callback that is invoked before the first frame.
    /// Create the glTF import task.
    /// </summary>
    void Start()
    {
        _task = RuntimeGltfImporter.GetImportTask(
             "https://awesomesaucelabs.github.io/piglet-webgl-demo/StreamingAssets/piggleston.glb");
        _task.OnProgress = OnProgress;
    }

    /// <summary>
    /// Callback that is invoked by the glTF import task
    /// to report intermediate progress.
    /// </summary>
    /// <param name="step">
    /// The current step of the glTF import process.  Each step imports
    /// a different type of glTF entity (e.g. textures, materials).
    /// </param>
    /// <param name="completed">
    /// The number of glTF entities (e.g. textures, materials) that have been
    /// successfully imported for the current import step.
    /// </param>
    /// <param name="total">
    /// The total number of glTF entities (e.g. textures, materials) that will
    /// be imported for the current import step.
    /// </param>
    private void OnProgress(ImportStep step, int completed, int total)
    {
        Debug.LogFormat("{0}: {1}/{2}", step, completed, total);
    }

    /// <summary>
    /// Unity callback that is invoked after every frame.
    /// Here we call MoveNext() to advance execution
    /// of the glTF import task.
    /// </summary>
    void Update()
    {
        // advance execution of glTF import task
        _task.MoveNext();
    }
}
```

Figure 3: An extension of the runtime import script from Figure 2 that prints progress messages to the Unity console. New lines are highlighted in bold.

Another important use of callbacks is to run custom code after a glTF import has successfully completed. For example, you might want to automatically resize the model, parent the model to another game object, or attach a custom `MonoBehaviour` to the model. These types of tasks can be accomplished using the `OnCompleted` callback. To demonstrate, the example script in Figure 4 uses the `OnCompleted` callback to obtain a reference to the imported model, then uses that reference to continually spin the model about the y-axis as if it were on a record turntable.

The example in Figure 4 marks the end of this tutorial. Good luck and happy coding!

```
using Piglet;
using UnityEngine;

/// <summary>
/// This MonoBehaviour provides a minimal example for using
/// Piglet to import glTF models at runtime.
/// </summary>
public class RuntimeImportBehaviour : MonoBehaviour
{
    /// <summary>
    /// The currently running glTF import task.
    /// </summary>
    private GltfImportTask _task;

    /// <summary>
    /// Root GameObject of the imported glTF model.
    /// </summary>
    private GameObject _model;

    /// <summary>
    /// Unity callback that is invoked before the first frame.
    /// Create the glTF import task.
    /// </summary>
    void Start()
    {
        _task = RuntimeGltfImporter.GetImportTask(
            "https://awesomesaucelabs.github.io/piglet-webgl-demo/StreamingAssets/piggleston.glb");
        _task.OnCompleted = OnComplete;
    }

    /// <summary>
    /// Callback that is invoked by the glTF import task
    /// after it has successfully completed.
    /// </summary>
    /// <param name="importedModel">
    /// the root GameObject of the imported glTF model
```

```
    /// </param>
    private void OnComplete(GameObject importedModel)
    {
        _model = importedModel;
        Debug.Log("Success!");
    }

    /// <summary>
    /// Unity callback that is invoked after every frame.
    /// Here we call MoveNext() to advance execution
    /// of the glTF import task.
    /// </summary>
    void Update()
    {
        // advance execution of glTF import task
        _task.MoveNext();

        // spin model about y-axis
        if (_model != null)
            _model.transform.Rotate(0, 1, 0);
    }
}
```

Figure 4: An extension of the runtime import script from Figure 2 that spins the imported model about the y-axis. New lines are highlighted in bold.

Runtime Import API

In Piglet, a runtime glTF import is accomplished by the following steps:

1. Create a `GltfImportTask` by calling `RuntimeGltfImporter.GetImportTask`, passing in the file path or URL of the input `.gltf`/`.glb`/`.zip` as a parameter.
2. Configure callbacks on the `GltfImportTask` (optional).
3. Call `MoveNext()` on the `GltfImportTask` until the glTF import has completed.

For concrete code examples demonstrating the above steps, see the Runtime Import Tutorial.

`RuntimeGltfImporter` provides the following methods for creating a `GltfImportTask`:

RuntimeGltfImporter Methods

Method	Return Type	Description

`GetImportTask(string uri)`	`GltfImportTask`	Create an import task that imports the glTF model from `uri`, where `uri` is an absolute file path, HTTP(S) URL, or Android content URI that points to a `.gltf/.glb/.zip` file.
`GetImportTask(Uri uri)`	`GltfImportTask`	Create an import task that imports the glTF model from `uri`, where `uri` is an absolute file path, HTTP(S) URL, or Android content URI that points to a `.gltf/.glb/.zip` file.
`GetImportTask(byte[] data)`	`GltfImportTask`	Create an import task that imports from the raw byte content of a `.gltf/.glb/.zip` file (`data`).

`GltfImportTask` provides the following methods for controlling its own execution:

GltfImportTask Methods

Method	Description
`MoveNext()`	Advance execution of the import task by a small increment. This method should be called repeatedly until the import task has completed.
`Abort()`	Abort the import task. This method should typically be called in response to a user action, such as pressing a "Cancel" button.

In addition, `GltfImportTask` provides the following callbacks for integrating custom code with the importer:

GltfImportTask Callbacks

Callback	Description
`OnProgress`	Invoked at regular intervals to report progress of `GltfImportTask`
`OnAborted`	Invoked when `Abort()` is called on `GltfImportTask`
`OnException`	Invoked when `GltfImportTask` throws an exception (e.g. file not found)
`OnCompleted`	Invoked when `GltfImportTask` completes successfully

Sample Application: PigletViewer

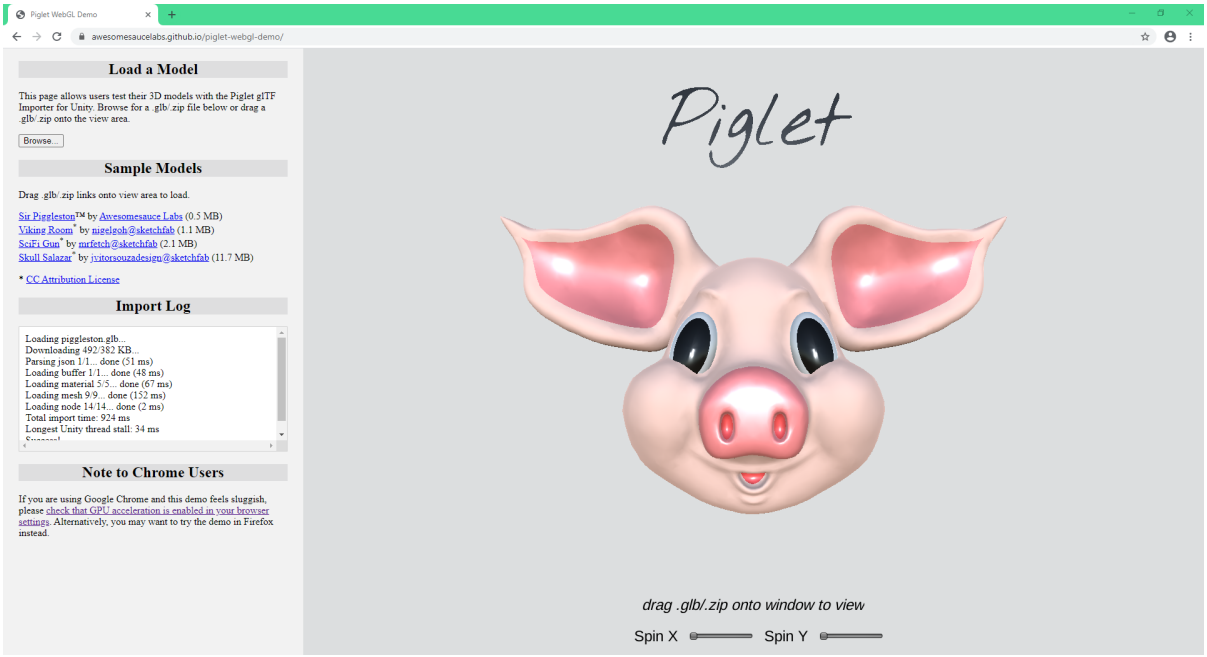


Figure 5: A screenshot of PigletViewer, a sample application which uses Piglet to view 3D models from glTF files.

PigletViewer is sample application which uses Piglet to view 3D models from glTF files (`.gltf`, `.glb`, or `.zip`), and which is used for the Piglet Web Demo. For the benefit of Piglet customers, I have published the source code and documentation for PigletViewer online at https://github.com/AwesomesauceLabs/piglet-viewer, under an MIT license. PigletViewer currently supports builds for Android, WebGL, and Windows, and thus it may be a useful reference for Piglet users developing for those platforms. In general, I recommend looking at the Runtime Import Tutorial before exploring the PigletViewer code, as the tutorial provides a much more succinct introduction to the Piglet API.

Footnotes

1. As of June 2020, Google Poly only provides glTF download links for models made with Google Blocks (as opposed to Tilt Brush). To see only Blocks-generated models on Google Poly, visit https://poly.google.com/blocks.

2. I have tested the Piglet Web Demo with Firefox and Google Chrome on Windows 10 64-bit. If you are using Google Chrome, you can improve performance of the demo by turning on hardware acceleration (i.e. GPU acceleration) in the browser settings. Currently this option is disabled in Chrome by default.