Google Sheets For Unity Manual

You always can get an up-to-date copy of this document at:
https://docs.google.com/document/d/14cRMeyFXhYYCHYjeNahGFLlsAHlc5HmVWp30ZtdqMkk/edit?usp=sharing

Google Sheets For Unity ("GSFU") allows you to retrieve and send data to Google Spreadsheets from a Unity3D project at runtime or design time, for easy, collaboratively, and simple tweaking and recording of game data.

This can usually be achieved by setting up a complex environment involving oAuth authentication, granular spreadsheets permissions, and developing in Unity the necessary code using the Google API for Spreadsheets.

GSFU simplifies all that, by deploying a web service (in Google's infrastructure, under your account), that will connect to a given Spreadsheet, and update/return data from a desired worksheet, in Json format, and all without involving Google user account credentials from the Unity side.

Then from Unity3D, you can get the info by poking the web service with the standard WWW class, and parsing it into ready to use data objects with the json library of your choice (the demo project includes LitJson).

Some of this may sound complicated if you never worked with Google services before, but will reveal its simplicity as soon as getting hands to the example project.

Also, a major keypoint about GSFU, is that the source code is included, for both, the web service, and the Unity class. This means optimus flexibility, allowing custom and useful practical customizations. For example, running GSFU from a Unity Editor menu to update the game local data before game start...

Installation and Setup

As stated, GSFU consist of two parts the Unity code, and the web service. And of course, the Google spreadsheet containing your data.

Deploying the web service script

First step into setting up GSFU, is the deployment of the web service script.

It is a Google Apps Script, that you can open and modify at will, and have it running as a web app. You need to have your own copy on Google Drive, that you can get here: https://script.google.com/d/13SnqGtbQOUcG1QkUKPQ_xdki58Y3wFZxjp-FWKmL4WtIlr9RMo5wd2o5/newcopy

Then, on to setting up the service:

1. Open your copy of the GSpreadsheetConnector google script.
2. In the script editor, go to "Publish" menu, and click on "Deploy as web app...".
3. You'll now see a dialog with the title "Deploy as Web App". Where you need to set the following values:

    Project version:
    Type "1" in the textbox, and hit "Save New Version".

    Execute the app as:
    Select "Me (your@google.email)".

    Who has access to the app:
    Select "Anyone, even anonymous".

4. Click on "Deploy".
5. A new dialog will show up, take note of the url under the label "Current web app URL".

    This is the Web Service URL, described at the section "UnityDataConnector Parameters".

6. Click "OK".

Finally, Google requires you to authorize the script. For this, open the "Run" menu, and click on the first function, "doGet". An script authorization dialog will appear, where you will be prompted to authorize. You can ignore/dismiss any error produced by this one-time in-editor script run.

You need to do all the previous steps again, if you make changes to your script that want to go

live. For more info, go to https://developers.google.com/apps-script.

You only need to deploy the web service once. It will work with any and all spreadsheets on your Google Drive.

Example Spreadsheet

Complementary to the Unity example project, a Google Spreadsheet is provided, ready for the example Unity project to connect and send/retrieve data.

You can (and should) get a copy of your own of this example spreadsheet following this link: https://docs.google.com/spreadsheet/ccc?key=0Ass-oWeEUz0IdGFLSzRPYmtxU0dDdzJsaUNHSDBiYUE&newcopy

When the time comes for creating your own spreadsheet with data, there are a few considerations, that you should take in mind for GSFU to work properly.

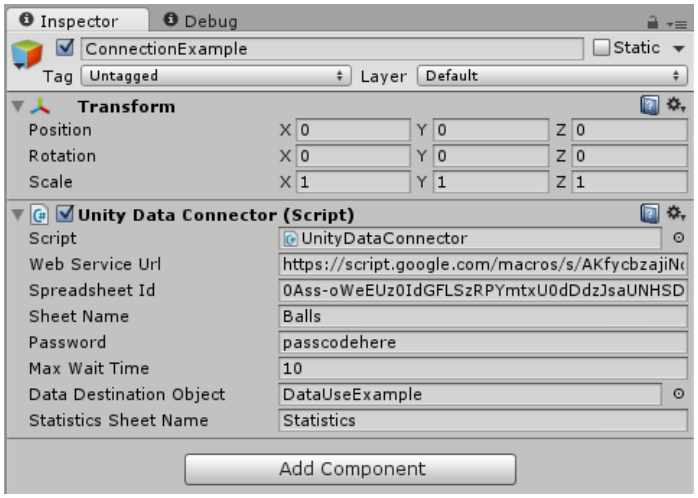
1. Follow the example spreadsheet scheme. First row in a data worksheet is always for the field names.
2. Avoid including extra content in the data worksheets, other than the specific data to be retrieved by Unity.
3. Always have a worksheet called “passcode”. If you want to set a blank password, remove any data from the first top left cell (A1). For more info read the “Security” section of the manual.
4. Google Sheets has some limits that you should take into account when deciding whether or not to use a Spreadsheet to store your script's data. For example, a Spreadsheet is limited to 400,000 total cells across all sheets and 256 columns per sheet. See this help center article for the full listing of limits. https://support.google.com/drive/answer/2505921
5. Formulas can be used to populate data to be retrieved by Unity.

In general terms, you will be safer by using the provided example spreadsheet as template for new spreadsheets.

Example Scene

From Unity, open the “ConnTest” scene. If you hit Play, you will see a working example of GSFU updating the scene parameters at run-time.

To understand how it works, stop the project and select the “ConnectionExample” GameObject in the hierarchy view. You will see this in the Inspector view:



This is the script that you need to include in your Unity project. The UnityDataConnector script, exposes fields with the required parameters for getting the data from your spreadsheet.

UnityDataConnector Parameters

- ❏ **Web Service Url:** Expects the complete url of the web service. For further info look in the “How to deploy the web service script” section.

- ❏ **Spreadsheet Id:** The id of the google spreadsheet containing your data. For more info read the “Getting the Spreadsheet Id” section.

- ❏ **Sheet Name:** Simply the name of the worksheet within your google spreadsheet, that holds the data you want to retrieve. You can easily extend you UnityDataConnector class for getting data from more than one worksheet.

- ❏ **Password:** The string you entered on your “passcode” worksheet. For info, read the “Security” section.

- ❏ **Max Wait Time:** The maximum time to wait before aborting the connection operation. This may vary depending on your internet connection. You can increase it if required by your connectivity.

- ❏ **Data Destination Object:** This is a generic reference to a object that will receive the retrieved data, through a Unity's SendMessage call.

- ❏ **Statistics Sheet Name:** Name of the worksheet within your google spreadsheet, that will receive new rows of data from your game. In the example spreadsheet, the name is "Statistics".

Getting the Spreadsheet Id

Now you need the id of the spreadsheet containing your data. It is highly recommended to use the example spreadsheet the first time.

To get the id, simply open a spreadsheet, and look in the URL. The id in the following URL https://spreadsheets.google.com/a/yourdomain.com/ccc?key=abc1234567 is "abc1234567".

Security

There are a few security considerations and practical matters to have in mind when using GSFU.

- The password for accessing your data from Unity, is set on the spreadsheet itself, on a worksheet called "passcode". This sheet needs to always be present, and not have any other data than the optional password on the first cell (A1).

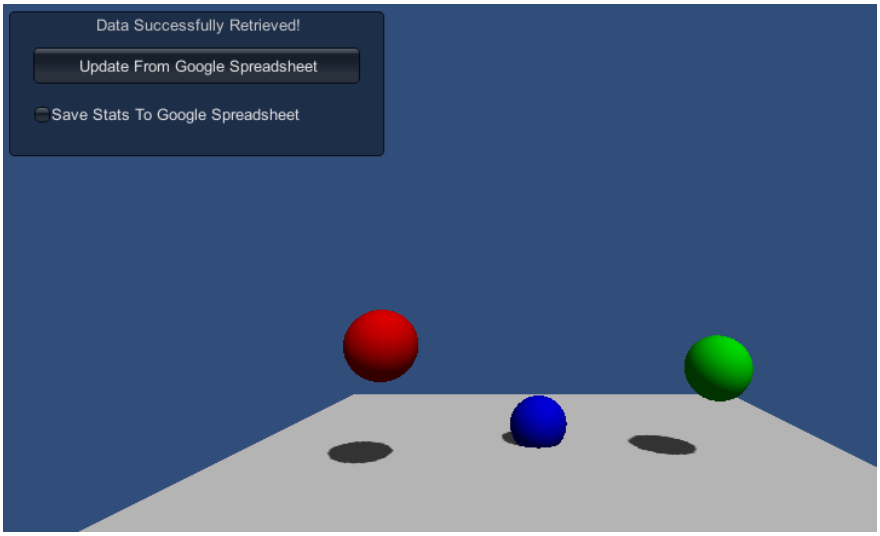
  Is important to note that while is quite practical, this security approach is recommended for development time, but probably not suited for released games. For released versions, it requires some work and careful consideration of security and connectivity aspects.

- If you want a blank password, simply leave the "passcode" worksheet blank.

- The Web Service must be deployed by the same Google user that will hold the Spreadsheet in his Google Drive account.

- The Google Spreadsheet permissions can be set in any way necessary, and won't affect the functionality of GSFU, as long as the Web App owner remains the spreadsheet owner.

The Example Project



The example project after updating from the online spreadsheet.

The included example is pretty much self explicative. When run, you will see three grey balls, bouncing at the same repetitive rate over a plane.

There will be also a GUI box at the top left, which includes three elements:

- ➔ The connection status reference, at the top, that will describe the connection process for the retrieving of data from the spreadsheet. If there are any errors, it will be reported there.

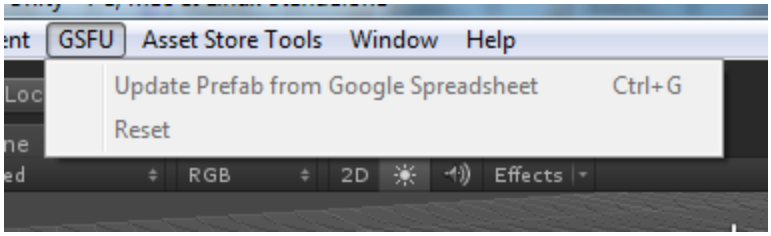
- ➔ The update button, in the middle, that will start the sequence of connecting to the sheet on the cloud, and get the data. It will fetch the ball color and drag coefficient by name.

- ➔ The save stats checkbox, that will start a process of sending to the online sheet, the records of each ball bounce over the plane, including a timestamp, ball name, and bounce magnitude. As opposed to the update button, the status and error handling of this is internal and won't report to the top GUI reference element, nor the Unity console, to avoid clogging; however in code you will find the places where to handle issues depending your case.

Editor Example

Since the latest version of Google Sheets for Unity, a new example is included, in which you can see a use case of the extension capabilities at design time.

For this feature, a new menu in the Unity Editor is created.



The new editor menu

As a practical measure, the menu items will only be enabled if an object named "*DataUseExample*" is currently selected in the scene. This is the scene object handling the data sent or retrieved from the Google Spreadsheets in the base example.

The result of the **"Update from Google Spreadsheet"** menu item action, is the same than in the base runtime example (the bouncing balls color and drag values get updated from the cloud), except that being at design time, you can opt to save the changes permanently in the scene and/or the project, or simply play the scene to test the changes.

The drag value of the balls is affected on the scene, while the color is affected on the hierarchy material asset, meaning the update is beyond the limits of the scene itself, and will persist on the project if desired.

The **"Reset"** menu option will set the color and drag values of the balls back to the original.