using google sheets as a server/db for networked prototypes in Unity

(by ian. please let me know (yt1319 at nyu dot edu) if you find any errors or typos!)

note: works well for asynchronous networked games and turn based games. although real-time is possible, it really depends on what you mean by that
note2: only do this for prototypes bc it'll break the moment you have too many players/more than one session.

TL:DR: use web requests to send data to **google forms**, which will get carried over to **google sheets**, and retrieve data from google sheets with web requests and you'll get json

1. writing data to google sheets through google form

Prep
- create a google form
- in settings, turn off requires sign in (turn off the restrict to xxx org)
- add questions (only tested short text response)

Get response url
- click Send, then open the link in a separate tab
- right click and inspect, ctrl+f and search for "formResponse"
- in that line, the string after "action=" is the post url

Get entry id
- click Send, then open the link in a separate tab
- right click and inspect, ctrl+f and search for "entry."
- all the results should look like "entry.123456789", they are the entry id for each question

In Unity
- in a script, add using UnityEngine.Networking;
- create a coroutine (IEnumerator)
- in the coroutine
    - create a new WWWForm form
    - populate the form with fields like this:
        - form.AddField("entry.xxxxxxxxx", your_value);

- form.AddField("entry.yyyyyyyyy", your_value2);
- ...

- put the response url in a string, called uri
- submit the information using:
  - UnityWebRequest req = UnityWebRequest.Post(uri, form);
  - yield return req.SendWebRequest();

- call the coroutine somewhere, with StartCoroutine()

now check the google form and it should have a new response (your_value, your_value2, ...) every time the coroutine is called

2. Checking/Modifying/Organizing the data in a google sheet

- on your form, click Responses and click the google sheets icon
- choose create a new google sheet
- only modify the data if you know what you're doing

(a few notes here: what i generally do is to create separate sheets (tabs at the bottom) and use some functions like Filter to parse information from the default sheet (Form Response 1) to organize different types of data. This can be for different types of data (game state vs. singular board state) but also incredibly useful for time parsing (entries submitted in the last 10 seconds).

pros:
- rarely need to touch the original data, unless when i'm sure they're no longer needed then i'll delete some rows
- later when we retrieve data, we can selectively retrieve parts of all the data because we've parsed them to a separate sheet, so that we only get the ones we need
- can be tailored in whatever way we want, eg. by category, by time, ordered, etc.

cons:
- functions like Filter can get slow when you have tens of thousands of responses, but so far with over 40k rows it's still pretty fast which is impressive..
- when you do need to edit responses (i.e. someone left inappropriate messages that you wish to remove/edit), you have to find them in the original row in Form Response 1 instead of the parsed sheets, which is a bit annoying since you need to locate the exact row (ctrl+f usually does it in a few seconds tho)

3. Retrieve the data from google sheet in Unity

Setup the google sheet
- click Share and change general access to "Anyone with the link"
- change the default sheet name from "Form Response 1" to something that doesn't have space in it
    - e.g. "FormResponse1"
- click File - Share - Publish to web
    - in Link, select FormResponses1 sheet, and webpage
    - hit publish

(note: you would do this for the other sheets if you're doing the parsing thing)

Get API key
- go to https://console.cloud.google.com/apis/credentials
- log in, create a project
- on the search bar, search "google sheets api"
    - click on the result, and click enable
- go back to credentials, create a new credential, this should create a new api key
- click on the new api key
    - change application restrictions to None, unless you know what you're doing
    - change API restrictions to restrict key, and select Google Sheets API
- copy the api key

In Unity
- in a script, create a coroutine (IEnumerator)
- in the coroutine
    - string uri = "https://sheets.googleapis.com/v4/spreadsheets/{Spreadsheet id}/values/{Sheet name}?key={API key}";
        - {Spreadsheet id} is the weird numbers and letters in the link to your spreadsheet, usually between d/ and /edit
        - {Sheet name} is the name of the sheet you're retrieving data from, i.e. the one that used to be "Form Response 1"
        - {API key} is the api key that you copied from the previous step
    - UnityWebRequest req = UnityWebRequest.Get(uri);
    - yield return req.SendWebRequest();

    - can check whether the request succeeded or failed with req.result

- if succeeded, req.downloadHandler.text will contain the content of the google sheet in json

- call the coroutine with StartCoroutine()
- have fun

note: there seems to be a request limit of 300 per minute for reading the data, and 60 requests per user per minute, according to google. So this thing pretty much easily breaks if I try to call it too often or have a lot of players, which, in either case, should probably look somewhere else.