World State SpreadSheet Manual

By Robert Bailey (@thrownerror)

A guide to using the World State SpreadSheet from thrownerror, as well as code documentation.

Version update - v2!
NEW VERSION v2
Changes: Fixed legacy bug with odd behavior towards end of tables
Added the requested "Unknown" element to faction relationships.
This value will not change and will remain static in it's one-way direction until manually changed to a different value.
This lets factions have one-directional relationships, and can help keep track of edge-case groups, like mercenaries the players don't know about, etc.

Full spreadsheet:
https://drive.google.com/open?id=1NhfZxgdow5snB1d3vd3fGmCE61mnQk3l1BgKASRaL_0
Demo sheet:
https://drive.google.com/open?id=127elyeIZycUOXigkMhvWIEfLPJRIPOMdCgxXQvHTmKQ

Table of Contents

Got Questions, feedback, or concerns?
Please reach out to me. I'm @thrownerror on twitter and reddit, and will try to respond to messages as quickly as I'm able. Feature updates will take longer, and depend on me finding time to fit them in.
If you absolutely want my email, reach me on one of those platforms first.

License

Main Functionality

This project is designed to help facilitate keeping track of factions and groups within a tabletop roleplaying context. “Factions” is used loosely here, and can encompass any size, from individual viewpoints to countries. Factions, named by the user on the “*Faction Information*” sheet, are stored and referenced on the “*Faction Relationships*” page, where a table of viewpoints is listed. This provides a quick overview of the dynamics of the world, and how NPCs view the players.

In addition to managing faction relationships, other utility sheets are presented. A random event table (*Random Events*), faction connection table (*Faction Connections*), and a quest generator (*Quest Generator*). Please feel free to use any or all sheets, as appropriate for your usage.

This tool is designed with improvisational structure in mind, and to remain relatively system and setting agnostic. As such, it provides “prompts” for scenarios, not full plans or descriptors. It’s intent is to help manage world states and large NPC group dynamics quickly and in a manner that allows room for appropriate interpretation. Feel free to combine output from this tool with others, either of your own making or outside creations, as best fits your needs.

All user-editable cells are highlighted in gray. Editing non-gray cells can lead to unpredictable behavior. Buttons are a light blue, and pure output cells are a green.

Note: For the purpose of this manual, “factions” are treated as localized groups, though, again, they can represent any meaningful division of characters.

Disclaimer: Much of the brunt of the tabulated work is done in GoogleAppScript, which will be touched on at the end, rather than traditional Spreadsheet software logic.
Also, some of the contents of this manual may change based on the version of the Spreadsheet, so no word is final (I will add a “recent changes” section when appropriate, and fold it into the appropriate headings after three months).

Faction Information and *Faction Relationships*

These pages are combined into one entry here due to them remaining intertwined.

Use case:

- The "*Faction Information*" sheet **must** be filled out for any other pages to properly function.
    - It lists the names of the factions, and saves the number of factions to use.
- The "factions" sheet is the core of the sheet.
    - It shows how factions feel about each other at the current time, and has a place to show edits to those feelings.
- Example:
    - Running a game inside a city with 12 factions, and wanting them to react to each other in a way that seems meaningful without needing much overhead cost or planning time.

Using *Faction Information* and *Faction Relationships*

- Enter a whole, positive number into Cell B1 (grayed out). This will auto-update the faction list with support for the desired number of factions.
- Enter names for each faction, in Cells B4 on. These will refer to the factions later.
    - Note: It only prompts up to 30 factions. This is only a formatting restraint, and more than 30 factions are supported. However, operation time gets longer.
- Once you are satisfied with the name and number of your factions, press the button labeled "Generate Faction Relationships."
    - This button generates a table on the "*Faction Relationship*" sheet.
- If you ever wish to change the faction names or number of factions, press the button labeled "Clear Relationship Table." Do not change any data before pressing this.
    - This button removes the formatted table on the "*Faction Relationship*" sheet.
        - NOTE: This also removes all data on that sheet.
    - This ensures the new table will be properly generated with proper formatting.
    - To fully delete a list of factions, ensure that this is pressed **before** changing the number of factions
- After pressing "Clear Relationship Table," enter in the new information, including the number in 1B. Press "Generate Faction Relationships" to generate a table with the new faction list.

Faction Relationships

- Ensure a table has generated using the "Generate Faction Relationships" button on the "*Faction Information*" sheet. Both tables start blank.

- The top table represents the current, or active, world state. These relationships are determined by rolling events based on the bottom table.
    - Factions can have the states: Despise, Hate, Tolerate, Neutral, Likes, Admires, Loves" towards other factions or players.
- The bottom table represents the "upcoming" world state.
    - Faction relationships can be set to any of their states.
        - Example: Faction A dislikes Faction B and players after B sent the party on a quest to raid Faction A. They currently "Like" each other, but the recent events warrant a downgrade to "Faction A 'Tolerating' B."
- Both tables are read as how factions in the leftmost column feel about factions in the top row.
    - Factions can have feelings about themselves - imagine it as internal coups, mixed motivations, or internal struggles.
- In addition, each faction has a space for how they feel about the players.
- The "Roll Relationships" button updates the top (current) table using the bottom (upcoming) table. It determines new relationships based off of the upcoming feelings, and then updates the top sheet accordingly.
- A "Save Top to Bottom" button is included to make it so the top table doesn't have to get manually copied down each time. It updates the cells on the bottom to reflect the top.
- If you do not have a set world state in mind at the start of using the tool, maybe do a "Roll Relationships" on the default table to generate a starting situation, and save it for future manipulation.
- If you'd like more of an ingrained history, consider Rolling, Saving, and Rolling repeatedly, keeping track of major changes. This way, you quickly generate a history for your world in only a matter of clicks!

Random Events

Use case:

- This page is designed for random situations and interpretations.
- Examples:What type of item is found, how an enemy reacts, how an abstracted event goes, what type of reinforcements arrive, etc.

Using *Random Events*

- To roll an unmodified result, press “Roll Results.”
    - This will clear the table, make a new roll, and place an “X” in the box the number lines up with. The rolled number is also placed in row 8 on the table.
    - To roll multiple numbers, edit B2 to the desired number of rolls (5 is the default maximum)
- To introduce modifiers, put a positive or negative integer in the gray boxes in row 6. These will directly apply to the roll when it is made.
    - The unmodified result is still visible in row 7 on the table.
- Elements are used as a vague term here, and can be relabeled by modifying the text in the gray boxes in row 5.
- To make a Standard Deviation roll on a Gaussian distribution, press the “Roll Gaussian Results” button.
    - If you would like to modify the distribution, Cell E2 modifies the Standard deviation (1 by default), and F2 modifies the Mean.
    - This might be useful for particular rolls at a set “intensity,” like rolls in the early game mostly being restricted to the first set by setting a mean of 15.
- If you would like to add any elements, then you will need to:
    - Copy columns G and H and paste them to the right, one column over for every element you want to add.
    - Copy column F and paste it to fill the new vacant columns from column G and H’s move.
    - Rename the Element number.
    - Edit the number in B2 to reflect the new number of elements you want to roll.
- Caution: If you roll more elements than there is space for, you will get errors.

Faction Connection Input and *Faction Connections*

Use case:

- Keeping track of meaningful connections between two factions.
- Examples: Trade routes in a pirate campaign, borders, passages between regions, lines of communication

Using *Faction Connections*

- Decide on a rough number of connections, and place that positive integer in cell B2.of *Faction Connections Input* , then click “Generate Connection Fields” or one of the Roll options.
    - Note: It’s easier to overestimate than constantly reset.
    - The fields will be generated in *Faction Connections*
- Then, set the connected factions. Click on the drop down arrow in the faction A and B columns (D and E) and then on a faction name. These names are brought from the factions on the home page.
    - Note: Currently, this relies on the user inserting factions to ensure that the connections are meaningful ones that make sense, something the tool can’t interpret (due to a lack of borders or in-game context).
- If you would ever like to only update the scale and event of connections, then click the button labeled “Roll Connections (Scale/Event Only)” within the *Faction Connections Input* sheet. This will update the table accordingly.
- To clear the table, click “Clear Connections” on *Faction Connections Input*.
- If you would like to modify the available values, refer to Using *Faction Connections Input*.
- As with any rolls, they can be ignored and freely interpreted. If you want to edit the values besides Factions, feel free to click the drop downs and change the result.

Using *Faction Connection Input*

Faction Connection Input manages all the discrete logic of *Faction Connections*. All buttons are in *Faction Connection Input*. If you just want connections and are happy with the default values, you only need to use the buttons and cell B2 in *Faction Connection Input*.

Creating Connections

- If you would like to just generate a number of blank connections, edit B2 to the number of connections you want.
    - Then click “Generate Connection Fields.”
        - The fields will be displayed in *Faction Connections*, and editable there
- To clear the connections, click “Clear Connections.”

    - This will clear the connections in *Faction Connections*. To fully clear, it is recommended to wait to edit B2 until clearing.
- If you would like to make randomized connections, click “Roll Connections (Not Factions)”
    - This will generate a number of connections for *Faction Connections*, and then assign all values but factions.
        - It currently assumes the game master has ideas for which factions can interact and how.
- If you would just like to roll the scale and event of connections, click “Roll Connections (Scale/Event only).
    - This will generate a number of connections for *Faction Connections*, and then assign all values but factions and types.

Modifying Connection options

- To edit how connections are generated, edit the table below the buttons in *Faction Connections Input*. The gray boxes, default categorized as “Connection types, Scale, and Event” are freely modifiable (as are the labels).
    - There are 15 slots for input in each category. This is a maximum.
    - Only three categories are supported.
    - Any blank fields are ignored.
- After modifying the below fields, click “Update Input Fields” to update all future generated connections and the current connections with the proper information.

Quest Generator Input and *Quest Generator*

Use Case:

- Generating quests based on the involved factions. These quests are intentional small prompts, not detailed outlines.
- Example: Generating a bulk of quick quest outlines to present players when they ask for employment, generating outlines to decide between, putting in set outlines and then having an unplanned disturbance.

Using *Quest Generator*

- Open *Quest Generator Input*. Don't worry about all the tables just yet.
- Determine how many quests to generate by changing the number in Cell B2 of *Quest Generator Input*.
- For every quest you want to make, update the required info.
    - Below B2, a table starts with various sections labelled "Quest X info" where X is an integer.
    - Edit how many factions, limitations, and disruptions you want involved on each quest.
        - Factions involved are how many factions are involved in a quest in any capacity.
            - Quests require at least one faction. They are picked from the *Faction Information* sheet.
        - Limitations are some restriction that affects quest completion.
        - Disruptions are wrenches that appear in the plan, possibly unexpected.
    - If you would like more than 12 quests at a time, both *Quest Generator* and *Quest Generator Input* need to have columns added to the end.
- Once you are done, click "Roll New Quests."
    - If you have factions in mind and don't want them randomized, click "Roll new Quests (Set Factions)."
    - Quests will be displayed on *Quest Generator*.
    - Each quest will have it's Info column updated.
        - Factions will be displayed, and picked from those on the *Faction Information* sheet.
        - Scale of the quest will be displayed.
        - Quest Acquisition shows how the players get the quest.
        - The Goal shows what the purpose of the quest is.
        - Afterwards, any Limitations will be generated, if applicable.
        - Finally, Disruptions are generated, if applicable.
- If you would like more or different options in the quests, refer to Using *Quest Generator Input.*

Using *Quest Generator Input*

- All of the fields of quest generation are updatable: Goal, Scale, Quest Acquisition, Limitation, and Disruption.
    - Updating these will be automatically reflected in the drop down the next time Quests are rolled.
    - This allows any form of customization to be more thematic or descriptive of situations.
- To change them, scroll down in *Quest Generator Input* until you see the second table (at rows 11-15). There are five categories.
    - Goal: What the party is meant to do.
    - Scale: Complexity or import of the quest.
    - Acquisition: How the quest is gained.
    - Limitation: Some restriction on the quest.
    - Disruption: Some complication during the quest.
    - The field's labels and intent can be changed, but there can only be five fields. Each field can hold 15 options.
    - To change a label, just edit the label.
        - Note: This does not affect quest output. The category in row 12 will still be generated as "Scale."
    - To change options:
        - Change the values in gray to any string. There are 15 spots per category.
        - Blank spots are skipped over.
        - I recommend keeping the strings short, but there is no character limit.
    - Note: This will not update already generated drop downs, but will affect new quests.

Things that might be coming

These are elements I'm considering adding, and might appear. The timeline for additions is entirely free-floating, and relies a bit on me running a game that needs the new feature, or getting enough demand.

- Expanded random table structure.
    - I like playing with statistics and math, so adding more controls over that.
- Mathier faction table
    - I've been toying with the idea of an additional faction table that exposes more of the math to the end user, and letting them have more control over the rolls and how the machine interprets that. Pretty much relies on me coming up with a neat and clean enough interface for that.
- A discrete "Trade Route" sheet
    - Effectively a variant of Faction Connections. Trade routes are nice, and technically you can already duplicate the "*Faction Connections*" sheet to make a Trade Route one, but I'd like some more control and economic-focused backdrops.
        - Some of that goal has been initially covered in a redesign of *Faction Connections* as currently seen.
        - If you want multiple faction connections that are more static and won't have to be changed later, you can always generate a mass of connections, fill them out, copy the sheet, rename it, and keep the original *Faction Connections* for future use.
            - The duplicated sheet will not be editable by *Faction Connects Input*

Things that I’m not adding

Note: Nothing on here is inherently eliminated from the World State SpreadSheet. If there’s a solution that works for you, by all means add in a new sheet and modify it. If you want to add this sheet to something else, ensure you maintain proper accreditation. These are just features that I’ve deemed out of scope/structure of the project, and will not be investing time towards.

- A map editor
    - It’s possible, but tedious to make it feel “right” and produce reasonable results that remain system agnostic AND looks good. I’d recommend something like http://gmworldmap.com/ or https://donjon.bin.sh/scifi/world/. Sheets/Excel just isn’t the best platform for this, especially within the current context of the project.
- A dungeon editor
    - This is a world management tool, not a dungeon creation tool. Again, I’d look for outside resources for this.
- System Calculators
    - This is meant to remain system agnostic. Including a set title can turn people off using a tool for other systems, and I’d like to avoid that. Technically, it rolls d20s in the background, but it also rolls impossible dice in other situations, so take that with a grain of salt.
- Item generators
    - Never say never, but reasonably do. It’s difficult to make this remain system agnostic. Additionally, I want more than just “Faction A has item X. Faction B also wants it,” or some variant therein. Other tools do random name generation better, and you’ll get better results at those locations. The Quest Generator is already a bit out of scope, but it felt like a fun challenge to make it flexible enough. Random Table also felt out of theme, but it felt weird to create a prompt structure without a random number generator.
- Name generators
    - In general, see Item generators. The sheet does work perfectly fine with Faction A/B/C/etc., so if you can’t decide on names it shouldn’t be a problem.

Changing Code

The majority of logic within the spreadsheet is taken care of within GoogleAppScript, not cell-logic. The language behaves very similarly to JavaScript. This was to allow for comments and personal preference.

Code Structure and General Standard:

- Each sheet has its own .gs code file for the most part. *X Input* and X pairs share a .gs file. This does lead to some method duplication, but it is rare.
- The .gs -> sheet pairings:
    - FactionInfoCode -> *Faction Information*
    - Faction Sheet -> *Faction Relationships*
    - Random Events -> *Random Events*
    - Faction Connections -> *Faction Connections Input* and *Faction Connections*
    - Quest Generator -> *Quest Generator Input* and *Quest Generator*
- Each .gs holds all the necessary variables and functions for it to function alone.
    - Occasionally, they reference variables from other sheets (such as faction names). These variables are then duplicated within the set sheet.
- Readability was prioritized over speed and data efficiency, since it is assumed not all users are familiar with GoogleAppScript. The sheet still maintains a small footprint in the browser cache. Some of this may get cleaned up in future versions as the compromise does annoy me to a certain degree.
- Any changes can be made to code, but changes must retain accreditation according to the MIT license.
- The code is commented, with descriptors at each function.
- Sheet scope variables are declared at the top, followed by methods.
- In general, complicated functionality is structured into other methods.
    - This makes a sheet's "core" function readable
- Functions have capitalized camelcase naming conventions, variables have lowercase camelcase naming conventions.

Code Documentation

I will be making a good faith effort to maintain this. If anything is added that I forgot to update here, please shoot a message my way.Code.gs
Commented out test file.

FactionInfoCode.gs

- MyFunction()
    - Test function designed to allow a testbed.
- UpdateFactionCount()
    - Updates the faction count, can be called anywhere
- MakeDropDown(var x, var y)
    - r: Row in spreadsheet
    - y: Column in spreadsheet
    - Generates a drop down menu with the faction feelings at that x,y location
- ClearDropDown(var x, var y)
    - x: Row in spreadsheet
    - y: Column in spreadsheet
    - Removes data validation at that x,y location
- GenTable()
    - Calls the Table generation functions in Faction Sheet

Faction Sheet.gs

- RollRelationshipTable()
    - Rolls relationship table for the number of factions.
- SaveTopTableToBottom()
    - Save the top table to the bottom.
        - Top table is treated as a “current” state, Bottom table is treated as the “upcoming” feelings.
- GetRowResults(var row, var factionCount)
    - row: the row being looked at
    - factionCount: the number of factions to look at
    - Gets a value for each square of that row of the relationship table.
- RollDieWithModifier(var modifier)
    - modifier: integer modifier for the roll
    - Rolls a die with the modifier of the current cell “relationship,” returns the new relationship to be displayed.

Random Events.gs

- RollSheet()
    - Rolls the sheet for however many columns are needed
- UpdateColumnCount()
    - Checks B2 and updates the amount of columns to be rolled
- RollSheetGaus()
    - Rolls the sheet for however many columns are needed, using Gaussian distribution as restrained by Standard deviation and the Mean.
- PlaceTick(var dieResult, var column)
    - dieResult: location of rolled result
    - column: which column is being rolled
    - Place a “X” in the spot which corresponds to the die roll within the correct column.
- RollDie(var column)
    - column: column the roll is happening in
    - Rolls a die for a set column, and then returns the result after displaying it.
- RollElement(column)
    - column: column the roll is happening in
    - Rolls a die for a column and places a tick in the proper location.
- RollGaussianElement(column)
    - column: column the roll is happening in
    - Rolls a die for a column using Gaussian distribution and places a tick in the proper location.

Faction Connections.gs

- UpdateConnectionCount()
    - Updates the count of connections to generate
- FillFactionArray()
    - Populates the faction array
- GenerateConnTable()
    - Generates a table for the amount of connections needed as told by UpdateConnectionCount. Connections are displayed on *Faction Connections* sheet.
- ClearConnTable()
    - Clears the connection table on *Faction Connections*, removes all data validation, and deletes the data stored in each cell.
- RandomizeConnections()
    - Gives connections that have a random scale, event, and encounter type.
- RandomizeScaleAndEvent()
    - Gives connections that have a random scale and event, but set encounter type.

- UpdateTypeArrays()
    - Updates the arrays for connection generation using the fields in *Faction Connections Input*.

QuestGenerator.gs

- FillFactionArray()
    - Fills the faction array with information from *Faction Information*
- FillLists()
    - Refills the quest generation information shown in the tables on the *Quest Generation Input* sheet.
- GenerateQuests()
    - Generates a quest using information from the *Quest Generation Input* sheet and displays them onto the *Quest Generation* sheet. Relies on user-input information under “Required Info” on *Quest Generation Input*.
- GenerateQuestsSetFaction()
    - Generates a quest using information from the *Quest Generation Input* sheet and displays them onto the *Quest Generation* sheet. Does not populate the faction field with any information besides the drop down. Relies on user-input information under “Required Info” on *Quest Generation Input*.
- RollQuest(var questNum)
    - questNum: number of quest being generated (used for offsetting table)
    - Clears the columns, gets parameters, and then generates and displays the quest.
- RollQuestSetFactions(var questNum)
    - questNum: number of quest being generated (used for offsetting table)
    - Clears the columns, gets parameters, and then generates and displays the quest. Does not populate faction space with anything besides drop downs.
- GetQuestParameters(var column)
    - column: where the information is being gained
    - Gets the required user-input information from the *Quest Generation Input* fields.
- ClearColumn(var column)
    - column: where the quest is being placed
    - Clears all data validation and information from the quest’s column.
- SetupFactionQuest(var count, var column, var detailed)
    - count: how many factions are being used
    - column: where the quest is being placed
    - Detailed: whether or not this quest has set factions
    - Displays the faction information for a quest, and toggles on whether or not factions are picked by code or placed by user depending on “detailed.”
- SetupOtherQuestInfo(var column)
    - column: where the quest is being placed
    - Sets up the rest of the quest info, such as the Goal, Acquisition, Scale, and any number of limitations or disruptions.

Thanks and Credits

I’d like to thank:

Professor David Simkins, from the Rochester Institute of Technology for letting me work on this and giving me tons of feedback, assistance, and advice.
Tadeo Menichelli and Stuart Burton for putting up with my experiments with this for the first chunk of 2018.
Carter Blalack, my roommate who has been more than willing to be a guinea pig and helped show me other DM viewpoints than my own.
The School of Interactive Games and Media and RIT in general, for giving me class credit for this and having a healthy supply of testers.

Copy of the code:

GoogleAppScript makes it difficult to disseminate code within the spreadsheet, so below is the full code as of the latest version.

Code.gs

```
/**
 * @OnlyCurrentDoc
 */
/*var sheet = SpreadsheetApp.getActiveSheet();
//Deprecated

function rollDieWithModifier(modifier){
  //get a random between 0 and 1, round it off. Multiply by 2, and subtract one.
  //Range: -1->1
  var windsOfChange = Math.round(Math.random()) * 2 - 1;

  var result = (Math.floor(Math.random() * 20) + 1 + modifier + windsOfChange * 3);

  //Too many extremes? put winds of change after floor/ceiling here
  if(result > 20) { result = 20}
  if(result < 0) { result = 0}
  return result;
}
*/
```

Random Event.gs

```
/**
 * @OnlyCurrentDoc
 */
var sheet = SpreadsheetApp.getActiveSheet(); //Gets the WorldState randomEvents table
var columnCount = sheet.getRange("B13").getValue();

//Rolls the sheet (default, random roll of a d100 as managed in rollDie);
function RollSheet(){
  //var elementCount = sheet.getRange("B13").getValue(); //location of element count for user
management
  UpdateColumnCount();
```

```
  for(var i = 0; i < columnCount; i++){
    RollElement(i+2);
  }
}

function UpdateColumnCount(){
  columnCount = sheet.getRange("B2").getValue();
}

//Rolls the sheet using a gaussian distribution (managed in rollGaussianElement)
function RollSheetGaus(){
  //var elementCount = sheet.getRange("B13").getValue();
  UpdateColumnCount();
  for(var i = 0; i < columnCount; i++){
    RollGaussianElement(i+2);
  }
}

//Manages placement of visible ticks for users
function PlaceTick(dieResult, column){
  //Clear ticks
  for(var i = 0; i < 20; i++){
    sheet.getRange(i+9,column).setValue("");
  }
  //Place a new one
  var next5Result = 5*(Math.ceil(Math.abs(dieResult/5)));
  var tickLocation = next5Result/5 + 8;
  sheet.getRange(tickLocation, column).setValue("x");
}

//Rolls 1d100, adds any modifiers, and then clamps it between 1-100
function RollDie(column){
  var dieRoll = Math.floor(Math.random() * 100 + 1);
  var modifiedDieRoll = dieRoll + sheet.getRange(6, column).getValue();
  //var dieRoll = Math.floor(Math.random() * 100 + 1 + sheet.getRange(2, column).getValue());
  if(modifiedDieRoll > 100){
    modifiedDieRoll = 100;
  }
  if(modifiedDieRoll < 1){
    modifiedDieRoll = 1;
  }
  sheet.getRange(7,column).setValue(dieRoll); //vanilla roll
  sheet.getRange(8,column).setValue(modifiedDieRoll); //saves the visible roll
```

```
  return modifiedDieRoll;
}

//Gets a die roll and places a tick in the correct spot
function RollElement(column){
  var dieResult = RollDie(column);
  PlaceTick(dieResult, column);
}

//Gets a location, then picks up standard deviaton and mean.
//Then, rolls a die, manipulates it through a general Gaussian distribution, and then places a tick
function RollGaussianElement(column){
  var standardDev = sheet.getRange("E2").getValue();
  var mean = sheet.getRange("F2").getValue();
  //math
  var dieResult = RollDie(column);
  sheet.getRange(7, column).setValue(dieResult);
  dieResult = dieResult / 100;

  var standardNormal = Math.exp(-(Math.PI) * (dieResult * dieResult));

  var genDistribution = (1/standardDev) * standardNormal * ((dieResult - mean)/standardDev);
  genDistribution = Math.round(genDistribution * -1);
  sheet.getRange(8, column).setValue(genDistribution);

  PlaceTick(genDistribution, column);
}
```

Faction Sheet.gs

```
/**
 * @OnlyCurrentDoc
 */
var sheet = SpreadsheetApp.getActiveSheet();


/*Rolls the relationship table with set values */
//Last edited May 4th 2018 by thrownerror
function RollRelationshipTable(){
  for(var i = 0; i < factionCount; i++){
    GetRowResults(i+1, factionCount)
  }
```

```
}

/*Copies the top sheet to the bottom sheet*/
//Last edited May 4th 2018 by thrownerror
function SaveTopTableToBottom(){
  for(var i = 0; i < (factionCount); i++){
    for(var j = 0; j < (factionCount+1); j++){
      var status = sheet.getRange(i+2,j+3).getValue();
      sheet.getRange(factionCount+2+2+i, j+3).setValue(status);
    }
  };
}

/*Gets a value for each square, taking into acount the modifier of the current value*/
//Last edited Feb 17th 2018 by thrownerror
function GetRowResults(row, factionCount){
  for(var j = 0; j <= factionCount; j++){
    var infoCell = sheet.getRange(row+factionCount + 3, j+3);
  //  var testCell = sheet.getRange(2, 3).setValue(0);

    var moodModifier = infoCell.getValue();
    moodModifier = RollDieWithModifier(moodModifier);
    sheet.getRange(row+1, j+3).setValue(moodModifier);
  }
}

/*Rolls a Die with the modifier of the current cell "relationship"*/
//Last edited February 16th, 2019 by thrownerror
function RollDieWithModifier(modifier){
  var numericValue;
//    var arraySet  = ["Despises","Hates","Tolerates","Neutral","Likes","Loves","Admires"];

  switch(modifier){
     case "Unknown" : numericValue = -1;
      break;

    case "Despises": numericValue = 0;
      break;

    case "Hates": numericValue = 5;
      break;

    case "Tolerates" : numericValue = 8;
```

```
    break;

  case "Neutral": numericValue = 10;
    break;

  case "Likes" : numericValue = 12;
    break;

  case "Admires" : numericValue = 15;
    break;

  case "Loves" : numericValue = 20;
    break;

  default : numericValue = 10;
    break;
}

    var testCell = sheet.getRange(10, 10).setValue(modifier);
//get a random between 0 and 1, round it off. Multiply by 2, and subtract one.
//Range: -1->1, adds some variety. Can be easily modifier
if(numericValue != -1){
  var windsOfChange = Math.round(Math.random()) * 2 - 1;

  var result = (Math.floor(Math.random() * 20) + numericValue + windsOfChange * 2);

  //Too many extremes? put winds of change after floor/ceiling here
  if(result > 36){
    return "Loves";
  }
  else if(result <= 36 && result > 28){
    return "Admires";
  }
  else if(result <= 28 && result > 22){
    return "Likes";
  }
  else if(result <= 22 && result > 18){
    return "Neutral";
  }
  else if(result <= 18 && result > 12){
    return "Tolerates";
  }
  else if(result <= 12 && result > 6){
```

```
      return "Hates";
    }
    else if(result <= 6 && result > -1){
      return "Despises";
    }
    else if(result <= -1){
      return "Unknown";
    }
  }
  else{
    return "Unknown";
  }
}
```

FactionInfoCode.gs

```
/**
 * @OnlyCurrentDoc
 */
var sheet = SpreadsheetApp.getActiveSheet(); //Gets the Faction Information sheet
var relationshipSheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Faction
Relationships") //Gets the faction table sheet
var factionCount = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Faction
Information").getRange(1,2).getValue();  //Note - most other sheets rely on information herer
(B2). Change at own risk

/*Helper function to update faction count, and ensure it is always correct when called*/
function UpdateFactionCount(){
  factionCount = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Faction
Information").getRange(1,2).getValue();  //Note - most other sheets rely on information herer
(B2). Change at own risk
}
/*Test function to look at filling a particular range.
In effect, a way to demo custom changes or values in one spot.
By default, goes to B1 on the sheet, and pastes the value there in a 4x4 block starting at D4.*/
//Last edited May 1st, 2018 by thrownerror
function MyFunction() {
  var valueTest = sheet.getRange(1, 2).getValue(); //Note - most other sheets rely on information
herer (B2). Change at own risk
  var selectionRange = sheet.getRange(4, 4, 4, 4);
```

```
  selectionRange.setValue(valueTest);

  //getRange(val1, val2, val3, val4)  -> (row, col, numRows, numColums)
  //getRange(val1, val2) -> (row, col);
  //getRange("A3") -> access a spot through excel notation
  //ex: 2 ,3 , 6, 4
  //2: Starting row - 2
  //3: starting col - 3
  //6: number of rows - 2->8
  //4: number of cols - C->G
}

/*Generates a drop down with the feelings between each faction (Despises, Hates, Tolerates,
Neutral, Likes, Admires ,Lives).
Then sets the cell to include proper drop down validation, enabling later functions to pick one of
these values */
//Last edited Feb 18th by thrownerror
function MakeDropDown(x, y){
  var arraySet  =
["Unknown","Despises","Hates","Tolerates","Neutral","Likes","Admires","Loves"];
  var dropDown = SpreadsheetApp.newDataValidation().requireValueInList(arraySet);
  relationshipSheet.getRange(x,y).setDataValidation(dropDown);
}
/*Clears a cell of any data validation, restoring it to it's original state */
//Last edited May 4th by thrownerror
function ClearDropDown(x,y){
  relationshipSheet.getRange(x,y).setDataValidation(null);
}

/*Generates the faction table*/
//Last edited May 4th by thrownerror
function GenTable(){
  UpdateFactionCount();
  GenerateTopTable(factionCount);
  GenerateBottomTable(factionCount);
}

/*Generates the top half of the faction table (the "resultant" feelings)*/
//Last edited May 4th by thrownerror
function GenerateTopTable(factionCt){
  var tableTopLocation = "B1"; //String determing table top location, where it builds from. Edit this
to change where the table starts.
```

```
  relationshipSheet.getRange(tableTopLocation).setValue("Group/Attitude");

  //Faction table
  for(var i = 0; i < (factionCt); i++){
    var factionName = sheet.getRange(4+i,2).getValue();
    relationshipSheet.getRange(2+i,2).setValue(factionName);
    relationshipSheet.getRange(1,3+i).setValue(factionName);
  };
  //Player column
  relationshipSheet.getRange(1, 3+factionCt).setValue("Players");
}

/*Generates the bottom half of the faction table (the "current" feelings)*/
//Last edited May 4th by thrownerror
function GenerateBottomTable(factionCt){
  //Variables for where the table is constructed
  var tableOffset = factionCt + 2; //Which column
  var tableBotLocation = tableOffset+1; //Where the table starts building. Seperated for purpose
of filling in information
  var tableBotLocationColumn = 2;  //Where the table's starting column is


relationshipSheet.getRange(tableBotLocation,tableBotLocationColumn).setValue("Group/Attitud
e");

  //Faction table
  for(var i = 0; i < (factionCt); i++){
    var factionName = sheet.getRange(4+i,2).getValue();
    relationshipSheet.getRange(2+i+tableOffset,2).setValue(factionName);
    relationshipSheet.getRange(1+tableOffset,3+i).setValue(factionName);
    for(var j = 0; j < (factionCt); j++){
      MakeDropDown(2+i+tableOffset,3+j);
    }
  };

  //Added player column
  relationshipSheet.getRange(1+tableOffset, 3+factionCount).setValue("Players");
  for(var k = 0; k < (factionCount); k++){
      MakeDropDown(2+k+tableOffset,3+factionCount);
  }
}

/*Helper function to reset the sheet. Calls clear top/bottom table*/
```

```
//Last edited May 4th by thrownerror
function ResetSheet(){
  UpdateFactionCount();
  ClearBottomTable(factionCount);
  ClearTopTable(factionCount);
}


/*Clears the bottom table*/
//Last edited May 4th by thrownerror
function ClearBottomTable(factionCt){
  var tableOffset = factionCount + 2;
  var tableBotLocation = tableOffset+1; //Where the table starts building. Seperated for purpose
of filling in information

  //Removes information in each cell in bottom table
  relationshipSheet.getRange(tableBotLocation,2).setValue("");
  for(var i = 0; i < (factionCt ); i++){
    relationshipSheet.getRange(2+i+tableOffset,2).setValue("");
    relationshipSheet.getRange(1+tableOffset,3+i).setValue("");
    for(var j = 0; j <= (factionCt); j++){
      ClearDropDown(2+i+tableOffset, 3+j);
      relationshipSheet.getRange(2+i+tableOffset,3+j).setValue("");
    }
  }
  relationshipSheet.getRange(1+tableOffset, 3+factionCt).setValue("");
}

/*Clears the top table*/
//Last edited May 4th by thrownerror
function ClearTopTable(factionCt){
    var tableTopLocation = "B1"; //String determing table top location, where it builds from. Edit
this to change where the table starts.

  //Removes information in each cell in top table
  relationshipSheet.getRange(tableTopLocation).setValue("");
  for(var i = 0; i < (factionCt ); i++){
    relationshipSheet.getRange(2+i,2).setValue("");
    relationshipSheet.getRange(1,3+i).setValue("");
    for(var j = 0; j <= (factionCt); j++){
      relationshipSheet.getRange(2+i,3+j).setValue("");
    }
  };
```

```
  relationshipSheet.getRange(1, 3+factionCt).setValue("");
}
```

Faction Connections.gs

```
/**
 * @OnlyCurrentDoc
 */
var sheet = SpreadsheetApp.getActiveSheet();
var connectionSheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Faction
Connections");
var connectionInputSheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Faction Connections Input");
var homeSheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Faction
Information");

var connectionCount = connectionInputSheet.getRange(2,2).getValue(); //Default, multiple
functions depend on this. Edit at own peril.

var encounterType = ["Peaceful encounter", "Violent Encounter", "Neutral Encounter", "", "", "",
"", "", "", "","","","","","","",""]; //15
var scaleType = ["Miniscule","Minor","Annoyance","Normal", "Disruptive","Course
Changing","Major", "", "" ,"" ,"","","","","","",""]; //15
var eventType = ["Attack","Authorities","Act of God","Normal resolution",
"Nature","Lost","Character caused", "", "", "", "","","","","","",""]; //15
var encounterInt = 3;
var scaleInt = 6;
var eventInt = 6;

var factionArray = new Array(factionCount);
FillFactionArray();

var dropDown = SpreadsheetApp.newDataValidation().requireValueInList(factionArray);
var typeDrop = SpreadsheetApp.newDataValidation().requireValueInList(encounterType);
var scaleDrop = SpreadsheetApp.newDataValidation().requireValueInList(scaleType);
var eventDrop = SpreadsheetApp.newDataValidation().requireValueInList(eventType);


/*Updates the Connections to be generated*/
//Last edited May 4th by thrownerror.
function UpdateConnectionCount(){
```

```
  connectionCount = connectionInputSheet.getRange(2,2).getValue();
}
/*Fills the faction array. Removed from updateType in case of need of seperation*/
//Last edited May 4th by thrownerror.
function FillFactionArray(){
  for(var i = 0; i < factionCount; i++){
    factionArray[i] = homeSheet.getRange(i+4,2).getValue();
  }
}
/*Generates the table itself on Faction Connections sheet*/
//Last edited May 4th by thrownerror.
function GenerateConnTable(){
  UpdateConnectionCount()
  UpdateTypeArrays();
  for(var j = 0; j < connectionCount; j++){
    var connectionNum = j+1;
    connectionSheet.getRange(j+5, 3).setValue("Connection " + connectionNum);
    connectionSheet.getRange(j+5, 4).setDataValidation(dropDown);
    connectionSheet.getRange(j+5, 5).setDataValidation(dropDown);
    connectionSheet.getRange(j+5, 6).setDataValidation(typeDrop);
    connectionSheet.getRange(j+5, 7).setDataValidation(scaleDrop);
    connectionSheet.getRange(j+5, 8).setDataValidation(eventDrop);
  }
}

/*Clears the table on Faction Connections sheet*/
//Last edited May 4th by thrownerror.
function ClearConnTable(){
    for(var k = 0; k < connectionCount; k++){
      connectionSheet.getRange(k+5, 3).setValue("");
      connectionSheet.getRange(k+5, 4).setDataValidation(null);
      connectionSheet.getRange(k+5, 4).setValue("");

      connectionSheet.getRange(k+5, 5).setDataValidation(null);
      connectionSheet.getRange(k+5, 5).setValue("");

      connectionSheet.getRange(k+5, 6).setDataValidation(null);
      connectionSheet.getRange(k+5, 6).setValue("");

      connectionSheet.getRange(k+5, 7).setDataValidation(null);
      connectionSheet.getRange(k+5, 7).setValue("");

      connectionSheet.getRange(k+5, 8).setDataValidation(null);
```

```
      connectionSheet.getRange(k+5, 8).setValue("");
    }
}

/*Generates fully random connctions on Faction Connections sheet. Does not edit the factions
invovled*/
//Last edited May 4th by thrownerror.
function RandomizeConnections(){
  UpdateConnectionCount()
  UpdateTypeArrays();
  for(var i = 0; i < connectionCount; i++){
    var scaleRoll = (Math.floor(Math.random() * scaleInt));
    var eventRoll = (Math.floor(Math.random() * eventInt));
    var encounterTypeRoll = (Math.floor(Math.random() * encounterInt));
    connectionSheet.getRange(i+5, 6).setValue(encounterType[encounterTypeRoll]);
    connectionSheet.getRange(i+5, 7).setValue(scaleType[scaleRoll]);
    connectionSheet.getRange(i+5, 8).setValue(eventType[eventRoll]);
  }
}
/*Generates only the Scale and Event of connections on the Faction Connections sheet. Does
not edit the factions involved*/
//Last edited May 4th by thrownerror.
function RandomizeScaleAndEvent(){
  UpdateConnectionCount()
  UpdateTypeArrays();
  for(var i = 0; i < connectionCount; i++){
    var scaleRoll = (Math.floor(Math.random() * scaleInt));
    var eventRoll = (Math.floor(Math.random() * eventInt));
    connectionSheet.getRange(i+5, 7).setValue(scaleType[scaleRoll]);
    connectionSheet.getRange(i+5, 8).setValue(eventType[eventRoll]);
  }
}

/*Updates the arrays for the type of connections, based off the fields on the bottom of Faction
Connections Input*/
//Last edited May 4th by thrownerror.
function UpdateTypeArrays(){
  FillFactionArray();
  encounterInt = 0;
  scaleInt = 0;
  eventInt = 0;
  var val = "";
  //gets the information
```

```
  for(var i = 0; i < 10; i++){
    val = connectionInputSheet.getRange(7, i+2).getValue();
    if(val != ""){
      encounterType[i] = val;
      encounterInt = encounterInt + 1;
    }
    val = connectionInputSheet.getRange(8, i+2).getValue();
    if(val != ""){
      scaleType[i] = val;
      scaleInt = scaleInt + 1;
    }
    val = connectionInputSheet.getRange(9, i+2).getValue();
    if(val != ""){
      eventType[i] = val;
      eventInt = eventInt + 1;
    }
  }
  //Updates the drop downs
  dropDown = SpreadsheetApp.newDataValidation().requireValueInList(factionArray);
  typeDrop = SpreadsheetApp.newDataValidation().requireValueInList(encounterType);
  scaleDrop = SpreadsheetApp.newDataValidation().requireValueInList(scaleType);
  eventDrop = SpreadsheetApp.newDataValidation().requireValueInList(eventType);
  for(var j = 0; j < connectionCount; j++){
    //Display of sheet
    connectionSheet.getRange(j+5, 4).setDataValidation(dropDown);
    connectionSheet.getRange(j+5, 5).setDataValidation(dropDown);
    connectionSheet.getRange(j+5, 6).setDataValidation(typeDrop);
    connectionSheet.getRange(j+5, 7).setDataValidation(scaleDrop);
    connectionSheet.getRange(j+5, 8).setDataValidation(eventDrop);
  }
}
```

Quest Generator.cs

```
/**
 * @OnlyCurrentDoc
 */
var sheet = SpreadsheetApp.getActiveSheet();
var homeSheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Faction
Information");
var questInfoSheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Quest
Generator Input");
```

```
var questSheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Quest
Generator");
var questCount = sheet.getRange(7,4).getValue();

var goalList = ["", "", "", "", "", "", "", "", "", "", "", ""];
var scaleList = ["", "", "", "", "", "", "", "", "", "", "", ""];
var acquisitionList = ["", "", "", "", "", "", "", "", "", "", "", ""];
var limitationList = ["", "", "", "", "", "", "", "", "", "", "", ""];
var disruptionList = ["", "", "", "", "", "", "", "", "", "", "", ""];

var goalListCount = 0;
var scaleListCount = 0;
var acquisitionListCount = 0;
var limitationListCount = 0;
var disruptionListCount = 0;



var  limitationDropDown =
SpreadsheetApp.newDataValidation().requireValueInList(limitationList);
var  disruptionDropDown =
SpreadsheetApp.newDataValidation().requireValueInList(disruptionList);
var  scaleDropDown = SpreadsheetApp.newDataValidation().requireValueInList(scaleList);
var  acqDropDown = SpreadsheetApp.newDataValidation().requireValueInList(acquisitionList);
var  goalDropDown = SpreadsheetApp.newDataValidation().requireValueInList(goalList);
var  factionDropDown = SpreadsheetApp.newDataValidation().requireValueInList(factionArray);

var factionArray = new Array(factionCount);

var involvedFactions;
var limitationCount;
var disruptionCount;

var questTableHorizOffset = 4;
var questTableVertOffset = 2;
var questInfoHorizOffset = 3;

/*Fills the faction array with any needed updates*/
//Last edited May 4th 2018 by thrownerror
function FillFactionArray(){
  for(var i = 0; i < factionCount; i++){
    factionArray[i] = homeSheet.getRange(i+4,2).getValue();
  }
  factionDropDown = SpreadsheetApp.newDataValidation().requireValueInList(factionArray);
```

```
}
/*Fills the lists of information (goal, acquisition, scale, limitation, disruption)*/
//Last edited May 4th 2018 by thrownerror
function FillLists(){
  for(var i = 0; i < 12; i++){
    //Fill goal
    val = questInfoSheet.getRange(11, i+2).getValue();
    if(val != ""){
      goalList[i] = val;
      goalListCount = goalListCount + 1;
    }
    val = questInfoSheet.getRange(12, i+2).getValue();
    if(val != ""){
      scaleList[i] = val;
      scaleListCount = scaleListCount + 1;
    }
    val = questInfoSheet.getRange(13, i+2).getValue();
    if(val != ""){
      acquisitionList[i] = val;
      acquisitionListCount = acquisitionListCount + 1;
    }
    val = questInfoSheet.getRange(14, i+2).getValue();
    if(val != ""){
      limitationList[i] = val;
      limitationListCount = limitationListCount + 1;
    }
    val = questInfoSheet.getRange(15, i+2).getValue();
    if(val != ""){
      disruptionList[i] = val;
      disruptionListCount = disruptionListCount + 1;
    }
  }
  limitationDropDown = SpreadsheetApp.newDataValidation().requireValueInList(limitationList);
  disruptionDropDown = SpreadsheetApp.newDataValidation().requireValueInList(disruptionList);
  scaleDropDown = SpreadsheetApp.newDataValidation().requireValueInList(scaleList);
  acqDropDown = SpreadsheetApp.newDataValidation().requireValueInList(acquisitionList);
  goalDropDown = SpreadsheetApp.newDataValidation().requireValueInList(goalList);
}

/*Generates the quests themseleves, helper method*/
//Last edited May 4th 2018 by thrownerror
function GenerateQuests(){
  FillFactionArray();
```

```
  FillLists();
  questCount = questInfoSheet.getRange("B2").getValue();
  for(var i = 0; i < questCount; i++){
    RollQuest(i);
  }
}

/*Generates Quests for a set faction, helper method */
//Last edited May 4th 2018 by thrownerror
function GenerateQuestsSetFaction(){
  FillFactionArray();
  FillLists();
  questCount = questInfoSheet.getRange("B2").getValue();
  for(var i = 0; i < questCount; i++){
    RollQuestSetFactions(i);
  }
}

/*Rolls a discrete individual quest*/
//Last edited May 4th 2018 by thrownerror
function RollQuest(questNum){
  ClearColumn(questTableHorizOffset+(2*questNum));
  GetQuestParameters(questInfoHorizOffset+(2*questNum));
  SetupFactionQuest(involvedFactions, (questTableHorizOffset+(2*questNum)), true);
  SetupOtherQuestInfo(questTableHorizOffset+(2*questNum));
}

/*Rolls a discrete individual quest with set factions */
//Last edited May 4th 2018 by thrownerror
function RollQuestSetFactions(questNum){
  ClearColumn(questTableHorizOffset+(2*questNum));
  GetQuestParameters(questInfoHorizOffset+(2*questNum));
  SetupFactionQuest(involvedFactions, questTableHorizOffset+(2*questNum), false);
  SetupOtherQuestInfo(questTableHorizOffset+(2*questNum));
}

/*Gets parameters for the quest from the required info fields */
//Last edited May 4th 2018 by thrownerror
function GetQuestParameters(column){
  involvedFactions = questInfoSheet.getRange(5, column).getValue();
  limitationCount = questInfoSheet.getRange(6, column).getValue();
  disruptionCount = questInfoSheet.getRange(7, column).getValue();
}
```

```
/*Clears the quest's column for more information*/
//Last edited May 4th 2018 by thrownerror
function ClearColumn(column){
  for(var k = 0; k < 40; k++){
    questSheet.getRange(k+questTableVertOffset, column-1).setValue("");
    questSheet.getRange(k+questTableVertOffset, column).clearDataValidations();
    questSheet.getRange(k+questTableVertOffset, column).setValue("");
  }
}

/*Sets up a quest's faction info with a Count of factions, in a column, and if it's detailed (true
means it picks a faction name, false means leave it blank for user to enter*/
//Last edited May 4th 2018 by thrownerror
function SetupFactionQuest(count, column, detailed){
  for(var i = 0; i < count; i++){
    questSheet.getRange(questTableVertOffset+i, column-1).setValue("Faction " + (i+1));
    if(detailed){
      var factionRoll = (Math.floor(Math.random() * factionCount));
      questSheet.getRange(questTableVertOffset+i, column).setDataValidation(factionDropDown);
      questSheet.getRange(questTableVertOffset+i, column).setValue(factionArray[factionRoll]);
    }
    else{
      questSheet.getRange(questTableVertOffset+i, column).setDataValidation(factionDropDown);
      questSheet.getRange(questTableVertOffset+i, column).setValue("Pick a faction");
    }
  }
}

/*Sets up all other quest info for a given column*/
//Last edited May 4th 2018 by thrownerror
function SetupOtherQuestInfo(column){
  questSheet.getRange(questTableVertOffset+involvedFactions, column-1).setValue("Scale:");
  questSheet.getRange(questTableVertOffset+involvedFactions,
column).setDataValidation(scaleDropDown);
  questSheet.getRange(questTableVertOffset+involvedFactions,
column).setValue(scaleList[(Math.floor(Math.random() * scaleListCount))]);

  questSheet.getRange(questTableVertOffset+involvedFactions, column-1).setValue("Quest
Acquisition:");
  questSheet.getRange(questTableVertOffset+involvedFactions,
column).setDataValidation(acqDropDown);
```

```
  questSheet.getRange(questTableVertOffset+involvedFactions,
column).setValue(acquisitionList[(Math.floor(Math.random() * acquisitionListCount))]);

  questSheet.getRange(questTableVertOffset+involvedFactions, column-1).setValue("Goal:");
  questSheet.getRange(questTableVertOffset+involvedFactions,
column).setDataValidation(goalDropDown);
  questSheet.getRange(questTableVertOffset+involvedFactions,
column).setValue(goalList[(Math.floor(Math.random() * goalListCount))]);

  if(limitationCount >= disruptionCount){
    for(var i = 0; i < limitationCount; i++){
      questSheet.getRange(questTableVertOffset+involvedFactions+3+i,
column-1).setValue("Limitation ");
      questSheet.getRange(questTableVertOffset+involvedFactions+3+i,
column).setDataValidation(limitationDropDown);
      questSheet.getRange(questTableVertOffset+involvedFactions+3+i,
column).setValue(limitationList[(Math.floor(Math.random() * limitationListCount))]);
      if(i < disruptionCount){
        questSheet.getRange(questTableVertOffset+involvedFactions+3+limitationCount+i,
column-1).setValue("Disruption" );
        questSheet.getRange(questTableVertOffset+involvedFactions+3+limitationCount+i,
column).setDataValidation(disruptionDropDown);
        questSheet.getRange(questTableVertOffset+involvedFactions+3+limitationCount+i,
column).setValue(disruptionList[(Math.floor(Math.random() * disruptionListCount))]);
      }
    }
  }
  else{
    for(var j = 0; j < disruptionCount; j++){
      questSheet.getRange(questTableVertOffset+involvedFactions+3+limitationCount+j,
column-1).setValue("Disruption");
      questSheet.getRange(questTableVertOffset+involvedFactions+3+limitationCount+j,
column).setDataValidation(disruptionDropDown);
      questSheet.getRange(questTableVertOffset+involvedFactions+3+limitationCount+j,
column).setValue(disruptionList[(Math.floor(Math.random() * disruptionListCount))]);
      if(i < limitationCount){
        questSheet.getRange(questTableVertOffset+involvedFactions+3+j,
column-1).setValue("Limitation");
        questSheet.getRange(questTableVertOffset+involvedFactions+3+j,
column).setDataValidation(limitationDropDown);
        questSheet.getRange(questTableVertOffset+involvedFactions+3+j,
column).setValue(limitationList[(Math.floor(Math.random() * limitationListCount))]);
      }
```

```
    }
  }
}
```