

Gacha for StreamAvatars

Made by @MatsuTheBear

Hello! I'm Matsu! And I'm the creator of Gacha for StreamAvatars. This project started as a way to enhance my streams and allow people to experience what gacha hell is. English is not my main language, so if I make mistakes here and there, let me know!

What does it do?

GachaForStreamAvatars is a Python program that allows people to simulate a gacha, enhancing StreamAvatars. Users are able to pull for avatars and gears, ranking from R to SSS, using channel points, donations or anything that can trigger an event, and then use them during your streams. The streamer has total control of what rarity avatars and gears have and their % of being pulled. If the same avatar or gear is pulled, rerolls are done. After N rerolls (number chosen by the streamer) the program will instead give back coins/points, used by StreamAvatars (you can enable avatars to be bought with points). It's a fun gimmick for members that have a lot of channel points.

Is it working as intended?

Yes! But bugs and glitches can happen. So as a safe measure, the program will register every avatar pulled for each user, so in case there is a problem with I/O speeds, OBS or whatever the reason is, you can always check who pulled what.

Where can I find you and where can I support you?

If you need to ask me anything, you can do it with a DM on Twitter @MatsuTheBear or on [Discord](#) (the link will direct you to my Discord server). I'll be more than happy to help you.

If you want to support me, you can do it by following me on Twitter and [Twitch](#). If you want to support me in other ways, I have a [Ko-Fi](#) page where you can also commission me emotes

Remember that this software is **free to use**. Also, if you want to ask me for help, I don't charge money for tech support :P Don't be afraid to contact me for any question or to ask for tech help, I will be more than happy to help you! Keep in mind that living in Italy, our time zones may be different, so I might not respond immediately.

Updates and more

My main goal is to convert everything as a plugin for StreamAvatars. Updates will be done through Github and Twitter, as well on my Discord server. For any bug you can notify me on my socials.

Setup

Requirements

- **Python 3.10** or higher version (available at <https://www.python.org/>)
- **StreamAvatars**
- **Macro Program** that allows to launch shell programs and Python files with arguments passed through Twitch APIs (**Touch Portal was used in this tutorial**)
- **OBS** with **Palakis WebSocket** installed (OBS 28 and above have already it installed)

How to install Python on Windows and macOS

1) **Download Python** from the official site (<https://www.python.org/downloads/>). Select the version you need. Remember to check the version shown. Every version above 3.10.0 is good, as long as is not below 3.10 (with 3.9 the code won't work, for example)

2) During the installation process, **be sure to select ADD TO PATH** if the selection is available (or add to environment variables if defined differently). Feel free to select other options based on your preferences.

- If you forgot to select PATH, *you don't need to uninstall Python*. You can still re-launch the installer after the main setup, and select "Modify". You should find the option available to be selected
- If you are more comfortable with your OS system, you can add Python directly in your environment variables.

3) In order to test if Python was installed correctly, **open Command** on Windows or **Terminal** on MacOS, **type "python3"** (without quotation marks) and **press enter**. If everything works, you should see something like this:

```
Python 3.10.7 (v3.10.7:6cc6b13308, Sep 5 2022, 14:02:52) [Clang 13.0.0 (clang-1300.0.29.30)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

How to setup StreamAvatars

Before I start explaining how you have to set up StreamAvatars, I highly suggest checking out the [StreamAvatars documentation](#) to learn how to import and set up characters, set up gears for different avatars and more. I'm assuming you already have avatars available and verified these requirements:

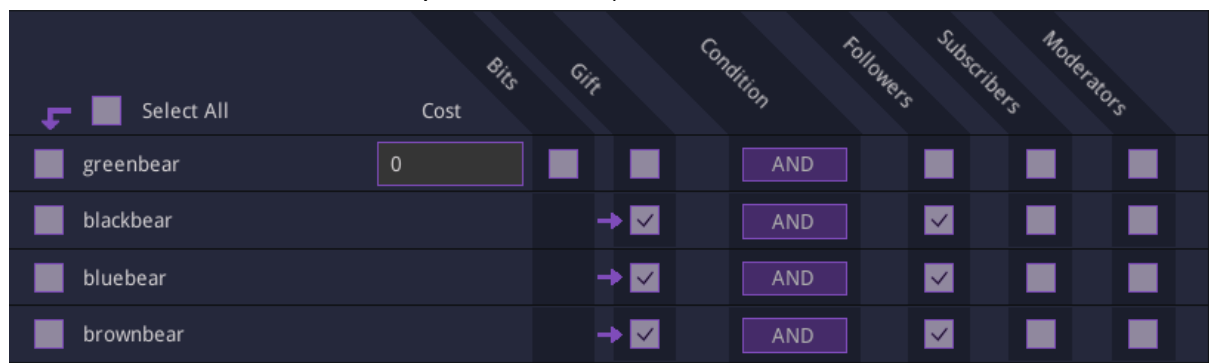
- At least one avatar per rank, and at least one free avatar (so min 4 avatars in total)
- If you are implementing also gears, at least one gear per rank

After these requirements, we can move forward.

0.5) Add avatars. This, of course, it's a must. You can make custom ones or import them through Steam Workshop.

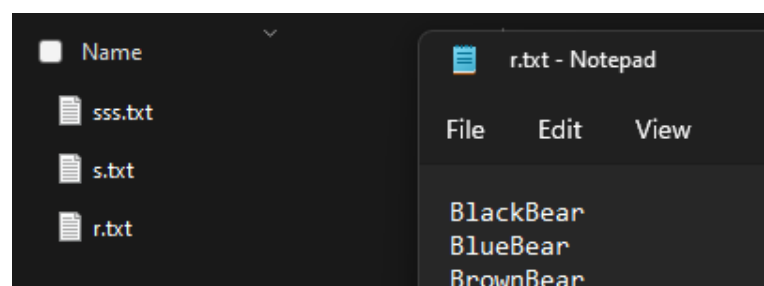
1) Go to **Shop Editing**, and for each character you want to be included in the gacha select **Gift**. You can make them also available only for followers or subscribers if you like. Remember to leave at least one character free to use. StreamAvatars doesn't allow having no free characters to choose, and will always choose one randomly to have a free one.

In my case, I have two of them are free to use (shown is greenbear, free to use, and black/blue/brown bears that are part of Rank R)



	Bits	Gift	Condition	Followers	Subscribers	Moderators
<input type="checkbox"/> Select All	Cost					
<input type="checkbox"/> greenbear	0	<input type="checkbox"/>	AND	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> blackbear		<input checked="" type="checkbox"/>	AND	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> bluebear		<input checked="" type="checkbox"/>	AND	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> brownbear		<input checked="" type="checkbox"/>	AND	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

2) Go to the folder of avatars or gear, select the rank file you like the avatar to be and write it down on a new row. So if you want an avatar to be R rank, go to **avatars/r.txt** and write **exactly** the name of the avatar. *CamelCase is ignored*, but you have to write the exact name of the avatar, or else it will not work.



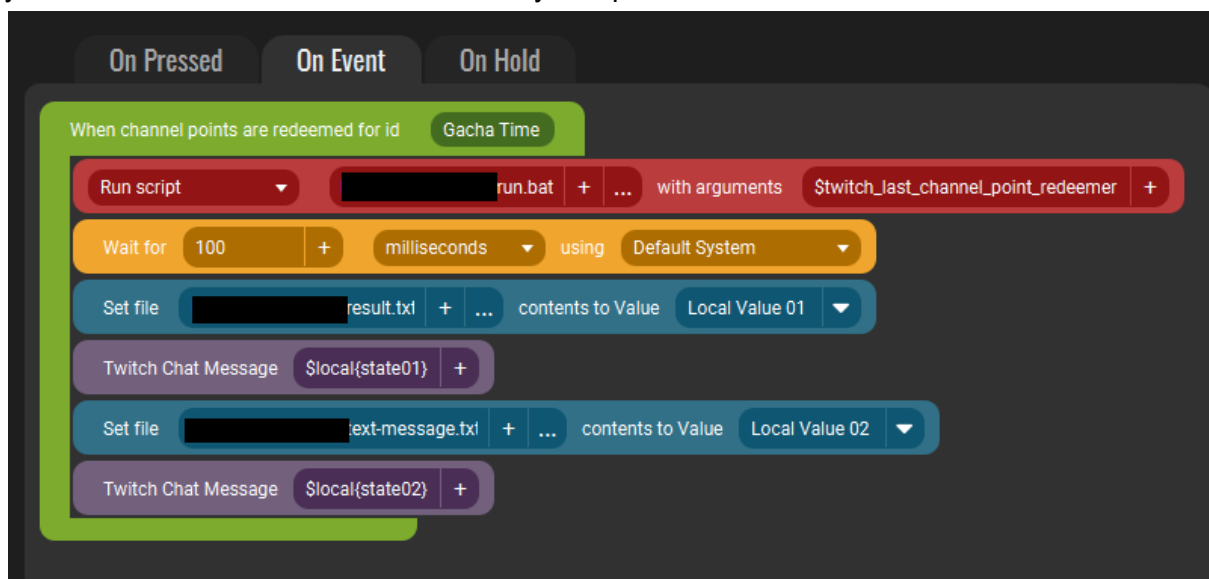
As you can see in the images, I have (example) **bluebear** available on StreamAvatars, and I chose it to be a Rank R bear. So after setting it up correctly on StreamAvatars, I wrote it down on a new row on **r.txt**, text file inside **avatars**. This is needed because the program will read the data from these files. If no avatar is available (if the file is just empty), you will get an error.

How to setup TouchPortal

This setup can work with any macro software that allows people to get data from Twitch APIs, after an event, and can connect of course to OBS. If you are interested in how everything works, this is a "simplified" version of the process:

User redeemes the channel reward → OBS + Macro Software get the trigger → Macro Software sends the username of who redeemed the reward to the python file → Python file returns data based on the input → Macro Software sends a message on OBS

- 1) **Download [Touch Portal](#)**, and select the version based on your OS. Complete all the steps to install it correctly. Highly suggested to restart your PC.
- 2) Open Touch Portal, go to **Settings** (Settings Icon top right → Settings) and select **Twitch**. Complete the setup.
- 3) To be absolutely sure that Touch Portal is communicating correctly with OBS, download [Palakis Websocket for OBS](#). If you have OBS 28 and above, you *don't* have to download it, because it's already installed.
- 4) If you have the free version of TouchPortal, you have to do this step on a button. If not, you can do it in Global Events. Here is my setup.



If you don't know how to do that, here are listed all the things you have to search on the searchbar and the values you need to include

On Channel Points Redeemed

- Write the **name of the channel point redeem** (in my case "Gacha Time"). It must be the same, it's CamelCase sensitive

Run Batch File

- If you are using Windows
 - Select **windows.bat** file inside the folder **run-files**
 - If you are using macOS
 - Select **macos.sh** file inside the folder **run-files**
 - The option "Using" must be set to **/bin/bash**
 - Just to be sure (you never know with Apple), open **Terminal**, locate the **macos.sh** file and type "**chmod u+x macos.sh**". This allows TouchPortal to launch the file with all the permissions granted.
 - For the Twitch argument, select +, then Twitch → Channel Points → Twitch Last Channel Point Redeemer. You should have `$twitch_last_channel_point_redeemer` for now.
 - Next, hit space and write **avatar** or **gear** based on what you are setting up the event for. So basically
 - **Avatar Gacha:** `$twitch_last_channel_point_redeemer avatar`
 - **Gear Gacha:** `$twitch_last_channel_point_redeemer gear`
- You can copy the string and paste inside the argument section.

Wait for Timer

- It's just a safe option for I/O speeds. You can put whatever you want (or remove it)

Files Contents to Value

- Select **result.txt** inside the folder **files** and **Local Value 01**

Send Chat Message

- Select +, then **Local States** → **Local Values** → **Local Value 01**
- You should have `$local{state01}` shown

Do the same for Value02 if you want the "Congrats" message

Keep in mind that based on multiple requests, the program may fail and give errors. Imagine two people redeeming the same reward at the same time and let's assume the program takes 1 second to elaborate and give back the result (worst case scenario). The first person request will be overwritten by the second and only the second one will have the reward. You can fix this with two simple methods:

- 1) If your macro software allows it, make all the request queueable, so even if you have 100+ redeems, the program will work as intended and no ghost/dirty data will appear
- 2) On Twitch, you can set a timer for the redeem. So if someone redeems it, no-one else can before X time. It's the safest option, but not recommended if you have a larger community.

Script File Setup

There is not a lot to change here. The only thing you have to do is go inside the **run-files** folder and modify **windows.bat** if you are using Windows, or **macos.sh** if you are using macOS. Inside those files, replace YOURPATHHERE with the path of the python file.

- On Windows, you can do it by right-clicking the python file, and select **Copy as Path**.
- On macOS, left-click the file and then press Option+⌘. It will show the path on the bottom of the Finder screen, and you can then press right click on the python name → **Copy gacha.py as PathName**

Path consists of the **directory of the file** and the **file itself**, so examples of paths are

- macOS: **/Users/Matsu/GachaGithub/gacha.py**
- Windows: **F:/GachaGitHub/gacha.py**

Errors

Here are some errors that can happen, with possible solutions. If you get a different error not included here, feel free to contact me

ERROR | Main: reward is None, expected Coins or Reward. Check getReward(): an error occurred in getReward, so nothing was given as an output. Check getReward error for further info.

ERROR | getReward(): Cannot choose from an empty sequence: this means that the file it's trying to get data from is empty. So, check the files "r.txt", "s.txt" and "sss.txt" both in **avatar** and **gear**

ERROR | getReward(): [Errno 2] No such file or directory: the file specified does not exist. This can be caused by missing files from import or deleted files. the code without arguments (from cmd/terminal or from Python IDLE) or create the corresponding missing folders

ERROR | getReward(): cannot access local variable 'rewards' where it is not associated with a value: the variable is empty because the function was not able to read data. This can be caused by missing or empty files. Be sure that they exist and they have values inside

ERROR | getReward(): TypeReward is not avatar or gear, returning None...: the argument passed is not equal to "avatar" or "gear". Be sure to have written the argument correctly

ERROR | setRecovery(): user does not exist: the user who redeemed the recovery function does not have any reward obtained/recorded. Be sure that the user redeemed something, or the user did not change their username. If so, just change the name of the corresponding txt file in **users/**

TroubleShooting

In order to troubleshoot the code, the easiest way to do so is to launch the program in cmd/terminal and put this code here. Remember to replace yourpath as shown before.

python3 yourpath/gacha.py test avatar

You should have something similar below. From here, you can check if there are any error happening.

```
OK | Main: got arguments: matsuthebear avatar
OK | getRarity(): function called
OK | getRarity() returned value: 41, function closed
OK | getReward(): function called
OK | getReward(): strings obtained: typeReward avatar and user matsuthebear
OK | getReward(): no rewards found, giving coins instead, function closed
OK | setFilesToEmpty(): function called
OK | setFilesToEmpty(): function closed
OK | Main: got coins, starting SetCoins...
OK | setCoins(): function called
OK | setCoins(): function ended
```

Code

If you are here, there is only one reason: you want to understand how the [redacted] code works. Worry not! I'll explain here below. First, let's explain how the simplified steps work

If the arguments are **username** and **avatar** or **gear**:

- 1) The program will get a random rank, from R to SSS
- 2) Based on the rank, it will return a reward. If after X rerolls it can't find a new reward, it will give back coins instead
- 3) It writes the result into a txt file, ready to be read by the macro software

Else, if they are **username** and **recovery**

- 1) It will return a file containing all the "!gift username avatar/gear name" commands, so that the user can get back all the rewards already obtained.

If no arguments are passed

- 1) It will launch a setUp function, where all the folders/files needed for the program are created. If they already exist, it will skip this process.

Now for the fun part. I'll explain for each function what they do and what the return type is. Keep in mind this is written based on the order of the functions

setFilesToEmpty(): simple function that literally sets some files to empty. This allows the code to write clean data and for the macro software to not read dirty data.

setGacha(): function that creates folders and files that are missing, required for the program to work. It reads the "setup.txt" file inside **files** folder, and each line represents a file or a directory. It first checks if the file/dir already exists. If not, if there is "txt" inside the name, that means is a file, if not is a folder.

setRecovery(username, typeReward): In input, it gets the username and the type of reward, that can be avatar or gear. Based on the type, the function will open the user's rewards file and read all the contents. The function will write this into a file named "recovery.txt", adding to the string obtained a gift command. This function is mostly used for emergencies

setReward(username, typeReward, reward): in input, it gets the username, the type of reward (avatar/gear) and the reward itself, a value that is obtained through the function getReward. It basically opens the reward and message files, writes inside the corresponding strings (!gift or !currency add if the rerolls were not effective) and then closes them.

setCoins(username, rarity): similar version of setReward, though this time based on the rank (aka rarity). As said before, if the rerolls are not effective, coins are given instead. This function writes in the reward txt file the "!currency add" command based on the rank (so if rank R 500 coins, S 2500 coins and SSS 10000 coins. These values can be modified in the "Constants" section

getRarity(chanceR, chanceS): this function gets the rarity of the reward based on the chances setted up. So if the chance of R rank is 70% (3 stars), and chance of S rank is 25% (4 stars), I know that, being the sum equal to 100, the chance of chanceSSS must be 5. The variable is still in the code but commented because it's not used.

getReward(username, rarity, typeReward): this function first sets up the directory corresponding on the type of reward ("avatar" or "gear"), then based on the rarity it opens the linked file ("r.txt", "s.txt" or "sss.txt"). From here, reads all the content of the file, cleans all strings, and then opens up the user file and repeats the process, having at the end two arrays. From here, there are different roads:

- 1) Gets a random reward from the pull
- 2) Checks if the reward is already obtained by the user
 - a) If so, increments the variable i in the cycle (starting from 0, can reach as max value the value of Rerolls) and goes back to point 1
 - b) If not, returns the reward
- 3) If the rerolls are not effective, the function will return "Coins"

Based on the path, the function will return the corresponding result

main(): This is the core of the program. It first checks if the arguments passed are more or equal than 3 (Python file, username, type of Reward). If so, it checks if the third argument passed is "avatar", "gear" or "recovery". If is avatar or gear, it will then call, in order:

getRarity → setFilesToEmpty → getReward → setReward (if pull successful) / setCoins (if pull unsuccessful)

In case of recovery, it will launch setRecovery. If there are no arguments passed, it will launch setGacha.

