ANCIENT CAVE STARRING GOEMON - RANDOMIZER MANUAL

Version: 2.1
Date: April 19, 2024
Latest Release: Version 2.1
Contact: Twitter | ko-fi | Discord

Check out my Ko-fi at ko-fi.com/abyssonym for free articles and exclusive membership content!

Original documentation by **abyssonym**, reformatted by **Emerald_Void**. All credit goes to abyssonym. Thank you for creating such an incredible tool that has rekindled my love for one of my all-time favorite N64 games.

I. QUICK START

II. FAQ

III. CONFIG FILE OPTIONS

IV. FLAGS AND COMMAND LINE OPTIONS

V. CREDITS, REFERENCES, & SPECIAL THANKS

I. QUICK START

1. Download the randomizer from https://github.com/abyssonym/mn64rando/releases

1a. On Windows, you have the option to download the file `mn64rando_windows.zip`. This version is a little slower, but does not require a Python installation on your system. If this version does not work for you, see step 1b.

1b. On other platforms, you must download the file `mn64rando_source.zip`. This version requires Python installed on your system. All of the files in this zip archive must be present for the randomizer to work.

Extract the contents of the zip archive to its own folder.

2. Acquire a ROM of either Mystical Ninja Starring Goemon or Ganbare Goemon: Neo Momoyama Bakufu no Odori for N64. If you want to verify, these are the relevant MD5 hashes:

```
EN - 643cce1ab06f97e9590241d27e5c2363
     f2162be647e4aa59254ec5ed7db1e94a (byte-swapped)
     e8d83efd203e533d734933e0a14f1a8c (little-endian)

JP - 9b929e0bf8d63e62820f2fa6257cc5cf
     9704b99f5f86879c0482d7223257dcef (byte-swapped)
     e5e8d1d0cd03bbcf8bd58151bd84e58f (little-endian)
```

Place the ROM in the folder containing the randomizer.

3. This randomizer uses a configuration file for the randomizer settings. Verify that the file `mn64_settings.yaml` is present in your randomizer folder. Open the file in a text editor to check the settings. If you want to make changes to this file, it is recommended that you copy the file to make a backup. For more information on what each of the settings do, consult the "CONFIG FILE OPTIONS" section below.

4. On Windows, run the file `mn64rando.exe`. Otherwise, run the file `randomizer.py` using Python.

4a. The randomizer will ask for ROM filename. Input the name of the file from step 2, or drag-and-drop it over the command line window. To maximize the chances of the randomizer finding the appropriate file, you should ensure that the ROM file is in the same folder as the randomizer. Sometimes, if there are special characters or spaces in the file path, the randomizer will be unable to find the file, so it is best to give the ROM a simple name like `mn64.z64` and input this file name instead of the full file path.

4b. The randomizer will ask for a seed number. If you don't care, it is fine to leave this blank. If you want to share your seed with others (i.e. for a race), note that different systems may produce different results even for the same seed, so it is best not to rely on consistency from the randomizer.

4c. Next, the randomizer will ask for a config file. You can leave this field blank to use the default filename from step 3, `mn64_settings.yaml`, or input the name of a custom config file if you have one. Make sure that your config file is in the same folder as the randomizer.

5. Wait for the randomizer to finish. The randomization is very complex, so this can take around 20 minutes depending on your system specs, your config file, and how many times the randomizer needs to retry the randomization.

6. After completing the randomization, two new files should appear in your randomizer folder.

6a. The first file is the randomized ROM, which will have the seed number in the file name. Open this file in your N64 emulator to play the randomized game.

6b. The second file is the spoiler log, a text file with the seed number and "spoiler" in the file name. Hold on to this file in case you get lost, don't know how to progress, or need to debug something.

After following each of these steps, you should be ready to play!

II. FAQ

Q: What is the objective of the game?

A: Obtain the four Miracle Items, then use the Pemo Pemo Statue at Ugo Stone Circle to travel to space and defeat the final boss. The Miracle Items are all in their original locations, but there is a setting to start with Miracle Snow already, which is enabled by default.

Q: What about cutscenes and events?

A: Almost none of the game's events have been modified. The randomizer's logic is routed around and through these cutscenes.

Q: What has been modified?

A: The Pemo Pemo statue takes you directly to the final boss instead of the final dungeon. Additionally, the Fire Ryo is no longer required to charge the Karakuri Camera or other chargeable weapons. The default warp location is now Oedo Castle. Various small patches are used depending on the options selected.

Q: What about locked doors?

A: Generally, most locked doors have been removed. This is because the vanilla keys are not interchangeable; the keys are just cleverly placed to prevent you from ever encountering two keys or two locks of the same type at any time. This becomes very confusing if the rooms are shuffled, so the old locks are removed, and a new key and lock arrangement is generated instead. This means you may encounter keys and locks in places they would not normally appear.

Q: Where can I find keys?

A: Keys may be placed anywhere there would be a "valuable" pickup in vanilla. This includes the locations of keys, gold dangos, cat dolls, surprise packs, Elly Fants, or Mr. Arrows.

Q: What happens if I sequence break?

A: There are no safeguards against sequence breaking techniques like Ebisumaru Slide, Mermaid Dash, or Ryo Hover. Using these techniques may place you out of logic, which means you may end up stuck in an area with no exit, and the keys that you find won't fit the locks that you find. Maybe someday in the future, someone will write up alternate logic that incorporates sequence breaks, but I don't have the knowledge or the will to do that at this time.

Q: What about other techniques?

A: The randomizer logic may require you to take damage or make precise jumps to progress.

Q: How does the Dragon Flute work?

A: The flute is safe to use; it won't place you outside of logic. However, some changes were implemented to make the warp flute intuitive and efficient. There are 11 unlockable warp locations: Goemon's House, the five town inns, and the five teahouses. The following warp screen locations were repurposed:

Oedo Castle → Goemon's House

Gourmet Sub Castle → Sogen Inn

To unlock a warp location, you **MUST ENTER** the building. That means going inside every inn and teahouse. If the location of the teahouse or inn was changed, the associated warp point will be outside of the building at the new location.

Q: How do I randomize enemies?

A: There is a `randomize_enemies` option in the config file. Note that this option may carry a small risk of game crashes. It only happened to me twice, and I patched the instances that I encountered, but I can't guarantee that it won't happen in some other edge case.

Q: Is there a way to make a "plandomizer" or a custom mod?

A: You can export text dumps of all of the game's actors and event scripts using the `export` code on the command line. Then, you can modify these dumps and import them using the `import` code. Use the `norandom` code to disable randomization. This gives you complete freedom to modify the game, including adding, removing, or changing actors that are not affected by the randomizer. If you only wish to make a plando, though, there is no easy way to do that, but feel free to experiment and edit any of the source files.

Q: What about custom game modes?

A: See **"CONFIG FILE OPTIONS"** below for more comprehensive information about your options. Currently, there are game modes that start you with Camera, Chibi-Ebisu, Mermaid magic, Miracle Snow, or the Flute. However, make note that using Ebisumaru Slide is not incorporated into the randomizer logic, and doing so may place you out of logic.

There are also separate game modes for starting as either Sasuke/Yae or Goemon/Ebisumaru. By default, you will start as Sasuke/Yae.

Q: My submarine disappeared even though I have `fix_bad_maps` enabled. What gives?

A: My somewhat-hacky solution for this was to to tie the submarine's appearance to the "money handling flag", which should always be off unless you interrupt a shopping sequence using the debug menu. So, you should be able to fix this issue by visiting a shop and completing the dialogue normally.

Q: Any tips?

A: Take notes. Remember landmarks and waypoints to important locations, especially relative to dragon warp points. This randomizer will test your navigation skills even with all of the "easy" settings enabled.

III. CONFIG FILE OPTIONS

all_warps
Value: True or False (default False)

Guarantees that every warp point (inns or teahouses) is accessible. This can impact generation performance, so it is off by default.

automash

Value: True or False (default False)

Makes certain long cutscenes progress automatically without player input. The affected cutscenes are any that include a character portrait. This turns cutscenes into ideal moments for short breaks, but beware that your input will still be required to respond to choices such as save prompts.

avoid_softlocks

Value: True or False (default True)

Creates a maze where you can't get stuck (for example, by falling off a ledge into an area where you can't get back up). This option is computationally slow but highly recommended. If you choose not to use it, it is recommended that you play with `enable_debug` enabled.

cluster_bgm

Value: True or False (default True)

Attempts to connect rooms that share a BGM, if the original exit also connected rooms that share a BGM. This option makes the maze somewhat easier to navigate and a great deal easier to listen to.

completionist

Value: True or False (default False)

When enabled, ensures that all key items are accessible in the maze, even if they are not required to reach the goal. This includes weapon upgrades, each character's weapons and magic, the super pass, the triton shell, the achilles heel, the special key, and the cucumber. Note that it does NOT guarantee all silver cat dolls, but it does guarantee that the Turtle Rock cat doll can be obtained. Check the randomizer output for a warning that the cat dolls could not be allocated if you care about that. This setting is disabled by default because it greatly increases seed generation times.

enable_debug

Value: True or False (default True)

Enables the use of Mystical Ninja Starring Goemon's native debug menu by pressing `L` at any time. This will allow you to get out of various softlocks and is recommended while the randomizer is still in its experimental stage. However, it may cause the ending credits to be interrupted.

fg_cache_limit

Value: 0 or more (default 10000)

This setting is used purely to optimize maze generation. The default value 10000 is a sane limit; any benefit from increasing or decreasing this value will be negligible or negative.

fix_bad_maps

Value: True or False (default True)

Changes various maps and events that are prone to softlocks and difficult to route around. These include:

- Entrance to Ghost Toys Castle (requires chain pipe)
- Invisible path in Gourmet Sub (requires camera)
- Disappearing submarine in Japan Sea
- Missable Chain Pipe
- Missable Benkei quest
- Missable Kihachi quest and Special Key

Enabling this option greatly improves seed generation speeds, and is basically required for any large maze with complex conditions.

fixed_singletons

Value: 0 to 1 (default 1)

Connects singleton rooms to their vanilla exits at the specified rate, i.e. a rate of 0.5 will connect roughly 50% of singleton rooms this way. A singleton room is a room with only one exit, for example, a shop or someone's house. Excluding these rooms from the randomization makes the maze MUCH easier to navigate and it is recommended to set this setting to the MAXIMUM value, unless you are specifically looking for a painful experience.

flute_anywhere

Value: True or False (default True)

Allows the flute to be used in any room, including in castles and shops. If `start_flute` is enabled, this will slightly improve seed generation times. If `start_flute` is disabled, it will greatly increase seed generation times. It is recommended to enable both `start_flute` and `flute_anywhere` together.

goal_based_missables

Value: True or False (default True)

This determines how the randomizer handles skippable events. If True, it will only force the player's path through those events if they are required to reach the goal. Previously, it was recommended to turn this option off to optimize key/lock randomization; however, this is no longer recommended, because the potential performance boost is minor compared to optimizing the maze generation phase.

hard_mode

Value: True or False (default False)

Quadruples the damage taken from all sources. As a result, most enemies will do a full 2 hearts of damage, meaning you will be able to survive 2 hits with the starting five hearts. This gives the many restaurants, shops, and cat dolls scattered throughout the maze more of a purpose. Note that helmets and armor will still negate the full damage of a hit, making them much more valuable. A side effect of this option is that any damage taken during the Sweets Minigame is a one-hit kill.

ice_kunai_logic

Value: True or False (default True)

When this setting is enabled, the maze generator will assume that the player cannot cross hot platforms without the ice kunai. It is recommended for inexperienced players to leave this setting enabled.

jp_super_jump_logic

Value: True or False (default False)

When enabled, this adds several super jump checks to logic that are only reachable on the Japanese version. This setting has no effect when randomizing the English version. This setting is experimental; I am not aware of all the locations where the Japanese super jump has an advantage. The setting is off by default to maintain consistency in generated seeds between the English and Japanese versions.

lazy_complex_nodes

Value: True or False (default False)

Lazily evaluates complex nodes during maze generation. Complex nodes are nodes that have strict requirements on how they can be reached, i.e. must be reached from a specific direction or while holding a specific item to prevent softlocks. Enabling this option will speed up maze generation, but will create the potential for softlock scenarios.

logic_filename **- Default value:** router_logic.txt

This name of the file that contains the reachability of every exit in the game. The default value will suffice unless you want to use custom logic such as incorporating sequence-breaking techniques like the Ebisumaru Slide into the maze logic. In those cases, you will need to specify a different filename containing the new logic. More information about how to write these logic files will be provided in a separate article.

map_size

Value: 0 to 1 (default 0.75)

This is a rough percentage of the total number of rooms that will be connected. There should be no issue connecting 100% of the rooms, and if the specified value is too low, the minimum number of rooms to make the maze beatable will be used. A default value of 75% is used to make the maze a bit smaller.

map_strictness

Value: 0 to 1 (default 0.75)

This option determines variance in the number of connected rooms, based on the value set in `map_size`. In other words, `map_size` establishes a rough upper limit, and `map_strictness` is a rough lower limit. If `map_strictness` is 100%, and `map_size` is 75%, then all of the 75% will be connected. But if `map_strictness` is 50%, then somewhere between 50% and 100% of the 75% will be connected. In practice, the result is almost always closer to 100% than 50%, but it is recommended to set this value a little bit below 100% for some flexibility. But it does not make much of a difference, so it is probably best just to ignore it and use the default value.

no_logic

Value: True or False (default False)

Generate a "no logic" seed that does not consider progression in any way, and is unlikely to be completable. This setting is used to quickly generate seeds for testing, and is off by default for obvious reasons.

nodes_filename

Default value: router_nodes.txt

This is the name of the file that contains the names of all of the nodes used in `logic_filename`. There are two types of nodes in this file; the basic node is an exit such as a door that can be connected to other doors. Nodes prefixed by `+` are noteworthy locations such as item pickups.

randomize_enemies

Value: True or False (default True)

Randomize all actors belonging to a group that I've identified as non-boss enemies. There are few restrictions on which enemies can become what, for example, crowds of small enemies can become crowds of giant crabs. This can result in certain maps requiring much more memory than they normally would, so there is a high risk of lag and there may also be a small risk of some maps crashing. This feature is highly experimental, but the crashes seem to be quite rare, so it is enabled by default.

randomize_keys

Value: 0 or more (default 2)

Create a layer of key-and-lock obstacles on top of the randomized maze. The keys will be placed randomly at the locations of "valuable" pickups specified in the `randomize_pickups` section. The specified value is the number of different key/lock configurations to generate, and the best one is chosen at the end. A higher value roughly increases the number of doors that are required to unlock, however, it is quite slow and has diminishing returns, so a value of 1-3 is recommended. A value of 0 disables locked doors altogether.

Note about keys in MN64:

The keys of a certain type (Silver, Gold, Diamond) are NOT interchangeable. The randomizer's key generation feature will attempt to cleverly place keys only in locations such that you will never be able to possess two different keys of the same type, and will never encounter two different locked doors of the same type. However, be warned that if you ever sequence break the randomizer logic using techniques such as Ryo Hover or Ebisumaru Slide, then this is no longer assured, and you may pick up, for example, a Silver Key that does not fit into the next Silver Lock that you find.

randomize_pickups

Value: True or False (default True)

Randomizes "valuable" pickups including keys, gold dangos, cat dolls, surprise packs, Elly Fants, Mr. Arrows, and item pots. Except for item pots, these are also potential locations for newly generated keys mentioned in the `randomize_keys` section. This option will attempt allocate enough cat dolls to reach 100% completion, but if this is not possible, the randomizer will print a warning.

retry_limit

Value: 0 or more (default 750)

This option determines how long the randomizer should try to make a particular seed work before giving up. In general, you should rarely hit this limit unless you also increase `time_limit`. There should not be much reason to increase this limit unless you are trying to solve routing logic that is more complex than the default settings.

etry_limit_close

Value: 0 or more (default 1000)

This limit establishes an "overtime" contingency for seeds that appear close to completion, i.e. all of the victory conditions have been met, but there are still softlock scenarios in the maze layout. It should be a higher number than `retry_limit`.

ryo_hover_logic

Value: True or False (default False)

When this setting is enabled, the maze generator will assume that the player is capable of using Ryo Hover to cross large gaps. This logic is still experimental, so it may not be correct for all cases where Ryo Hover might be used. Furthermore, some of the Ryo Hover checks are quite precise, so it is recommended for inexperienced players to leave this setting disabled.

Specifically, this setting adds the path to Oedo Castle to logic, which is technically reachable via Ryo Hover but is also reachable using easier tricks. This setting does NOT add the Gourmet Sub Camera Room Ryo Hover to logic, because of inconsistent lag in this room and because I wasn't able to pull it off even once on the English version in an emulator.

sasuke_mode

Value: True or False (default True)

Begin the game as Sasuke and Yae instead of as Goemon and Ebisumaru, with the maze logic updated accordingly. None of the cutscenes have been altered, but you can recruit Ebisumaru by viewing Yae's cutscene and Goemon by viewing Sasuke's. This option is enabled by default to give Sasuke some time in the spotlight.

skip_complex_nodes

Value: 0 to 1 (default 0.5)

Removes complex nodes from the maze generation pool at the specified rate (in other words, a value of 0.5 will remove a random 50% of complex nodes from the pool). Complex nodes are nodes that have strict requirements on how they can be reached, i.e. must be reached from a specific direction or while holding a specific item to prevent softlocks. Increasing this value will speed up maze generaton on average, but will result in certain rooms being excluded from seeds at a high rate.

start_camera

Value: True or False (default False)

Begin the game with Ebisumaru's Karakuri Camera already in your possession, with the maze logic updated accordingly. The Karakuri Camera is a required item for several rooms with softlock potential, so enabling this option can speed up maze generation slightly. However, this improvement is negligible over the default settings.

start_flute

Value: True or False (default True)

Begin the game with Yae already recruited and Yae's flute already in your possession, with the maze logic updated accordingly. This option is recommended because there can be a frustrating degree of backtracking in many seeds, and the flute is not guaranteed to be obtainable otherwise. Note that Castles are no longer warp locations; you can only warp to Goemon's house, the teahouses, and the inns.

This setting drastically improves seed generation times, so it is highly recommended to leave this enabled for large or complex maze settings.

start_mermaid

Value: True or False (default False)

Begin the game with Yae already recruited and the Mermaid Magic already learned, with the maze logic updated accordingly. Not much to say about this option, I just added it to test stuff.

start_minimaru

Value: True or False (default False)

Begin the game with Ebisumaru's Chibi-Ebisu magic already learned, with the maze logic updated accordingly. Note that the maze logic does NOT reflect the use of the Ebisumaru Slide, so using it may result in sequence breaks or softlocks, but the option is there if you want it.

start_snow

Value: True or False (default True)

Start with the item "Miracle Snow" already in your possession, with the maze logic updated accordingly. This can potentially skip a long fetch quest at the end of the game, which can be quite tedious while navigating a randomized maze. But if you want the "full" experience, feel free to turn this option off.

time_limit

Value: 0 or more (default 600)

This is a hard limit, in seconds, on how long to spend trying to make a particular seed work. Four minutes is usually long enough for reasonable seeds to complete, but if your machine is slower than mine, you may need to increase this limit.

IV. FLAGS AND COMMAND LINE OPTIONS

Command line arguments are specified in the following format:

python randomizer.py <ROM_FILENAME> <CODES> <SEED NUMBER>`

For example, to obtain a clean dump of the ROM, you would use the command:

python randomizer.py mn64.z64 export.norandom 0`

To import a modified dump to a clean rom, you would use the command:

python randomizer.py mn64.z64 import.norandom.debugmenu 0`

The available command line codes are as follows:

debugmenu - Has the same effect as `enable_debug` in the config file. Enables the use of Mystical Ninja Starring Goemon's native debug menu by pressing `L` at any time, even when not generating a randomized maze.

enemizer - Has the same effect as `randomize_enemies` in the config file. Allows enemy randomization even when not generating a randomized maze.

export - Export two text dumps after randomization: first, a dump of all the actors on every map. Second, a dump of all the event scripts. The output filename will be in the format `<ROM_FILENAME>.export.txt`. If there are no other modifications to the ROM via `enemizer`, `import`, or by generating a randomized maze, then the dump will be clean with all of the original pointers intact. This dump can then be modified and used with the `import` option.

import - Modify the input ROM by importing text dumps in the format produced by `export`, updating the actors and event scripts according to the specifications in the dumps. These modifications take place before randomization, if randomization is enabled.

norandom - Do not generate a randomized maze. When using this code, the config file is ignored and doors are not randomized.

V. CREDITS, REFERENCES, & SPECIAL THANKS

Project 64

- Being a Nintendo 64 emulator with debug features that works in Wine

(even if it crashes whenever I so much as look at it funny)

- revision 6247-dd7ec63 was four months ago so maybe it's more stable now

github.com/Fluvian

- Information about Konami compression scheme (lzkn64)

- Information about room metadata

Sterbenz, Parasyte, & spinout

- Information about N64 checksum scheme (n64crc)

tcrf.net/Mystical_Ninja_Starring_Goemon_(Nintendo_64)

- Various info about game data structures
- Debug code

twitch.tv/claude

- Concept
- Background noise
- Playtesting

twitch.tv/Stingchameleon

- Uploaded a youtube video 10 years ago with some useful RAM addresses

twitch.tv/zar6565

- Playtesting

VR43XX 64-Bit Microprocessor User's Manual

- I speak 65816 I don't understand any of this
- Why does no one list opcodes, I'm working with binary data over here

Abyssonym- because I deserve it. This was really frickin hard