Tears of the Kingdom Cooking

A complete, in-depth TotK Cooking System explanation

Introduction

This document attempts to resume and explain the entirety of the Cooking system of *The Legend of Zelda: Tears of the Kingdom*. It *should* be 100% accurate, but I cannot guarantee it is. If you think there's an error somewhere, please let me know on Discord (my username is echocolat). While this explanation uses a lot of data from the game's code and files, it will not always follow the order of operations the game does when calculating everything related to the cooking.

Let's assume we already chose our one to five material(s). I'll go through the cooking of the meal step by step to make it easier to understand.

Resources

- Tears of the Kingdom Cooking Calculator
- Online version of the calculator above
- Tears of the Kingdom Cooking Spreadsheet
- Tears of the Kingdom Datamining server

Getting the recipe

The first step of the algorithm is to find what kind of meal our one to five material(s) create. Firstly, the game checks how many *unique* materials are present in your combination. For example, a meal with two Acorns and one Apple has two unique materials (Apple and Acorn). If that amount is equal to one, it will start searching in the $SingleRecipeList$ list of possible meals (more about how it actually searches in the list in a bit). Else, it will start searching in the $RecipeList$ list of possible meals. However, if the game fails to find a meal in $RecipeList$, and if one of the materials have the $CookSpice$ tag[1], it also searches in $SingleRecipeList$.

Searching in $SingleRecipeList$

The game cycles through the list of SingleRecipe meals, first to last, and does a bunch of checks for each to determine if the materials match the meal's requirements. If they do match it, the meal is selected and the "Getting the recipe" step ends. SingleRecipes only have one $TargetList$, which basically is a requirement that needs to be met for the meal to be selected. Such a requirement can be made of one or multiple elements, such as "Sneaky River Snail" (one element made of a material), or "CookFish" (one element made of a cook Tag), or even "Acorn OR Chickaloo Tree Nut" (two elements made of two materials). If the unique requirement of a SingleRecipe is met, the game selects the meal as the result, and goes to the next step. If the game didn't find a single matching SingleRecipe at the end of the cycle, the result is the failed meal actor (e.g. Dubious Food), its effect is set to None, its effect time and level as well as critical rate are set to 0, its selling price is set to 2, ***and skip to the Conclusion step***.

[1] An actor can have one or multiple Tags that are all found in the Tag.Product RSDB. A bunch of Tags are used throughout the cooking steps, such as CookSpice, CookEnemy, CookFruit, etc. You can check the "primary" cook Tag of materials in /Data/MaterialData.json in the github repository or in the cooking spreadsheet.

Searching in $RecipeList$

Similarly to $SingleRecipeList$, the game cycles through the list of Recipe meals, first to last, and does even more checks to determine if the materials match the meal's requirements. Unlike SingleRecipes, Recipes can have multiple $TargetList$s, which means multiple requirements need to be met for the meal to be selected. A material from your list can only count for one requirement (after which, it's not in consideration for the "Getting the recipe" step anymore). For example, Copious Mushroom Skewers has four requirements, "CookMushroom", "CookMushroom", "CookMushroom" and "CookMushroom", which can be resumed to "CookMushroom AND CookMushroom AND CookMushroom AND CookMushroom". A single mushroom in your material list can only fulfill one of the requirements, after that it can't be used for requirements anymore. The requirements for $RecipeList$ can still have multiple elements, which means the full requirement "expression" can look like "(Raw Meat OR Prime Raw Meat) AND CookFish" (needs either a Raw Meat or a Prime Raw Meat, as well as a fish). If *all* requirements of a Recipe are met, the game selects the meal as the result, and goes to the next step. If the game doesn't find a single matching Recipe at the end of the cycle, and none of the materials have CookSpice, the result is the failed meal actor (e.g. Dubious Food), its effect is set to None, its effect time and level as well as critical rate are set to 0, its selling price is set to 2, ***and skip to the Conclusion step***. If one of the materials has CookSpice, the game searches in $SingleRecipeList$ as stated above.

Getting the effect and its stats

This step generates the eventual effect of the meal and its basic statistics (its duration if it has one and its level). Before proceeding, let's get a quick overview of the different effects and their own properties. Each effect has a $MinLv$ and a $MaxLv$, which act as the lowest and the highest the effect level can be. They also have a $Rate$ (we'll see its use later) and a $SuperSuccessAddVolume$ that we will call $SSAV$ from now, that is the additional level the meal gets in certain conditions (we'll see later too). Effects with a duration also have a $BaseTime$, we'll also see its use later. You can click this link to access a spreadsheet with all effects and their properties. I will be oversimplifying the process to make it easier to understand, if you want a more accurate description (although still far from the actual code) you can check the _effect() function (and below for the next steps) in my calculator.

This step starts with the $CookEnemy$ Spice: each material (not necessarily unique) with a $CookEnemy$ tag adds its $SpiceBoostEffectiveTime$ (which is a property of some materials) value to a bonus time variable which will be used later. This spreadsheet has the cook Tags of each material as well as their $SpiceBoostEffectiveTime$ (called Spice Effect time increase in the sheet).

If two materials of the cooking material list have different effects, the effect is set to None (unless it's an Elixir, in which case the meal result becomes the failed meal, e.g. Dubious Food). If no material in the cooking material list has an effect (unless it's an Elixir, in which case the meal result becomes once again Dubious Food), the game goes to the next step. Else (e.g. the meal has one effect):

- The effect potency and effect duration are initialized at 0
- The bonus time variable talked about earlier is added to the effect duration
- Each material in the cooking material list adds 30 seconds to the effect duration, and if its effect matches the effect of the meal, it does two things:
    - It adds its $CureEffectLevel$ (the potency of the material, Effect potency in the main spreadsheet) to the effect potency
    - It adds the $BaseTime$ (a property of the effect) to the effect time, if it exists
- The effect level becomes $EffectPotency * Rate$ (of the effect). The effect spreadsheet linked above also has equivalences between the effect potency and the effect level (code

wise they are the same variable, but I decided to separate them to make it easier for everyone). If the effect level is bigger than the $MaxLv$ of the effect, it becomes the $MaxLv$ of the effect.

If the resulting recipe is a Fairy Tonic, the effect is set to None, and the effect duration and level are set to 0.

Getting the health recovery

This step generates the basic health recovery amount, in health points (a Heart is four health points). The health recovery is initialized to 0, and each material of the cooking material list adds its $HitPointRecover$ (property of the material, Health recovered in the main spreadsheet) to it (if the $HitPointRecover$ exists). Then, the game multiplies the health recovery with the health recovery rate, which is 2, and moves on to the next step.

In case of a failed meal (Dubious Food or Rock-Hard Food), its effect is set to None, its effect time and level as well as its critical rate are set to 0, its selling price is set to 2, and if it's Rock-Hard Food its health recovery is set to ¼ Heart, if it's the Dubious Food its health recovery is set to 1 Heart. Then, ***skip to the Conclusion step***.

Monster Extract effects

This step covers the effects a Monster Extract can have on a meal, if it's present in it. This step is skipped if the recipe result is Dubious Food or Rock-Hard Food. It only happens once per meal, even if there are multiple Monster Extracts in the meal. If there is a Monster Extract in a meal, the meal cannot receive a Critical hit (***skip to the Non-CookEnemy Spice section***). Unlike Criticals, Monster Extracts can have a negative effect on your meals, so think well before using one. Now, let's move on to the actual effects of the Monster Extract on the meal.

- First of all, if the meal has an effect and an effect duration, the effect duration is sets to either 1 minute, 10 minutes or 30 minutes, randomly (each has a 33.3% chance of happening)
- If the meal has a null health recovery and has an effect OR is Hearty, its effect level is either set to the $MinLv$ of the effect, either gets the $SSAV$[2] (see Getting the effect and its stats section) of the effect added to it (each has a 50% chance of happening)
- Else, if the meal has a null health recovery and has no effect, it gets 3 hearts of health recovery[3]
- Else, if the meal has an effect and a non-zero health recovery, one of the following happens (each has a 25% chance of happening):
    - Health recovery is set to 1 health point (¼ heart)
    - Health recovery is added 3 Hearts
    - Effect level gets set to 1
    - Effect level gets the $SSAV$ of the effect added to it
- Else, if the meal has no effect and a non-zero health recovery, health recovery is either set to 1 health point (¼ heart) or is added 3 Hearts (each has a 50% chance of happening)

[2] Unlike level Critical hits (as we'll see in a minute), Monster Extract positive level effect doesn't always get an additional level on your effect. In the case of the effect being inferior to 1.0 prior to the Monster Extract effect, it will not be enough to reach level 2 even with that boost, and will be brung back to 1 after some steps.
[3] This case (null health recovery and no effect) isn't possible in the game, as there is no meal that has no effect and also has no health recovery at all.

Getting the critical rate

This step generates the chances your meal gets a critical hit, if critical hits are not inhibited (more on that later). When a critical hit happens, the music will be a bit more joyful and Link will be happier about the result. The meal will get a boost in one of the three main statistics (effect duration, effect level, and health recovery), depending on the conditions, more on that later.

The critical rate is initialized at 0, and the highest $SuperSuccessRate$ (property of the material, Added critical chance in the main spreadsheet) of all materials (if it exists) is added. Then the game adds $5 * UniqueMaterialsNum$ where $UniqueMaterialsNum$ is the amount of unique materials in your materials list. Finally, the critical rate is divided by 100 (in order to be an actual rate).

Critical hit effects

This step covers the effect a Critical hit has on a meal. As a reminder, a Critical hit can't happen if there is a Monster Extract in the meal materials. That step is also skipped if the recipe result is Dubious Food or Rock-Hard Food. There are three types of Critical effects: The effect duration gets 5 additional minutes, the effect gets SSAV added to its level, or the health recovery is added 3 Hearts. If a Critical hit is rolled (e.g. if a random number between 0 and 1 is smaller than Critical rate), the following happen:

- If the effect level is lower than 1.0, set it to 1.0
- If the meal has no effect, a health Critical hit is rolled
- Else, if the effect is Hearty, it gets one more Extra Heart (level Critical hit, $SSAV$ is 4)
- Else, if the effect is Stamina Recovery or Extra Stamina, the game checks whether or not the effect level has reached the $MaxLv$ of the effect
    - If the $MaxLv$ has been reached, a health Critical hit is rolled
    - Else, a random Critical hit is rolled between health and level (in the case of a level Critical hit, adds 2 Stamina Segments / Extra Stamina Segments, $SSAV$ is 2)
- Else, if the effect level has reached the $MaxLv$ of the effect, the game checks whether or not the health recovery has reached its $MaxLv$ (160, or 40 Hearts)
    - If the $MaxLv$ of the health recovery has been reached, a time Critical hit is rolled[4]
    - Else, a random Critical hit is rolled between time and health
- Else, the game checks whether or not the health recovery has reached max health
    - If the $MaxLv$ has been reached, a random Critical hit is rolled between level and health[5]
    - Else, a random Critical hit is rolled between level, health and time

Non-$CookEnemy$ Spice

This step handles the eventual health recovery and effect time boosts some materials can give to the meal, after Monster Extract and Critical hits are processed. For each ***unique*** material in the cooking materials list, the game checks if it has the $CookEnemy$ Tag. If it doesn't, it adds the material's $SpiceBoostHitPointRecover$ (Spice Health recovery increase in the main spreadsheet) to the Health recovery, and $SpiceBoostEffectiveTime$ (Spice Effect time increase in the main spreadsheet) to the effect duration (if it exists). The fact that the $CookEnemy$ Spice is separated

[4] A Gloom Recovery meal would get at least 50% chance of getting a time Critical hit, despite being a duration-less meal. That means you can get a Critical hit and not get any positive effect from it.
[5] For some reason, the game rolls between time and health criticals when the health is already maxed, instead of an expected roll between time and level.

from the non-$CookEnemy$ Spice makes it so that if a Monster Extract sets the time to 1:00 or 10:00 and/or the health recovery to ¼ Heart, the $CookEnemy$ Spice will be overridden by it, while the non-$CookEnemy$ Spice will be added on top of the 1:00 / 10:00 time and/or ¼ Heart recovery.

Meal bonuses

This step covers the eventual health recovery change the meal can receive depending on the recipe result. This step is very straightforward, the game adds the $BonusHeart$ (Bonus health in the main spreadsheet) of the recipe (if it exists) to the meal.

Adjusting values

This final step covers the clamping of effect time, effect level and health recovery between minimum and maximum values.

- If the effect time (if it exists) > 1800, it's set to 1800 (can't be above 30:00)
- If the health recovery is > 120 or if the effect is Hearty, it's set to 160 (any Hearty meal or with more than 30 Hearts of recovery gets Full Recovery)
- If the meal has no effect and has a null health recovery, its health recovery gets set to ¼ Heart
- If the effect level (if it exists) > $MaxLv$ for the effect, it's set to the $MaxLv$ (the level can't be higher than the max)
- If the effect level (if it exists) is between 0 (excluded) and 1, it's set to 1
- If the effect is Extra Heart or Gloom Recovery, the effect level gets rounded to the nearest 4, and is set to 4 if it's inferior to 4 (so that only full Extra / Ungloomed Hearts are allowed)
- The effect level is rounded down

Dubious Food's health recovery is set to 1 Heart and its effect is set to None, and Rock-Hard Food's health recovery is set to ¼ Heart and its effect is set to None.

Getting the selling price

Before moving on to the conclusion, let's calculate the selling price of the meal. The selling price is initialized to 0, then each material in the cooking materials list adds its selling price to the selling price of the meal, unless it has the $CookLowPrice$ Tag (see Cook Low Price in the main spreadsheet). In this case, it only adds 1 to the selling price of the meal. After that, the game checks the amount of materials (regardless of if they're unique) and multiplies the selling price of the meal with 1.2 if there's one material, 1.3 if there are two materials, 1.4 if there are three materials, 1.6 if there are four materials, and 1.8 if there are five materials. Then, it rounds the final selling price down. If the final selling price of the meal is lower than 3, it's set to 3. Finally, if the recipe result is a Fairy Tonic, a Dubious Food or a Rock-Hard Food, the final selling price is set to 2.

Conclusion

If you came here because you skipped to the end after a failed meal, you probably messed up somewhere when choosing your materials, but hopefully you still learned some stuff. Otherwise, voilà! The game finished calculating your beautiful meal's statistics and outputs it to you, all pretty and clean. This document ends there! Thanks for reading. If you don't understand something, or think there is an error somewhere, please let me know in Discord (my username is echocolat)!

Credits

- Spreadsheet and document made by Echocolat
- CookingMgr code retrieved and deciphered by dt12345
- TotK Cooking Calculator by Echocolat
- Base code used by the calculator by KingFoo
- Online part of the calculator by Glitchtest
- Additional research by Echocolat, dt12345 and Doge229
- Additional testing by Doge229