# Restaurant

## Customer area

### Tables and Chairs and Front Door

#### Additional tables can be purchased and auto placed. When no free space available, Tables are upgraded to support more Chairs. For simplicity I think each available seat should be treated as it’s own Table, even if it is just one seat on a multi-seat table.

#### Table States:

**Empty**, **Partial** or **Full**, (**Customers** will **Sit** if

**!Full && (Clean || Partial)**

**Clean** or **Dirty**. (**Waiters** will **Clean** if **Dirty && Empty**)

## Kitchen

### Storage (Freezer)

Storesunlimited **Ingredients** caught by the **Divers**. Can be accessed by **Chefs**.

### Cook Station (Upgradable, start with 1)

Where **Sushi** is prepared. Can only be accessed by **Chefs**. Takes an **Ingredient** and returns the finished **Sushi** after **Time to Prepare (TtP)** is complete. Locks in **Chef** during **TtP**.

### Counter

Holds X number of plated **Sushi**. Slots upgradable? Can be accessed by **Chef** and **Waiter**

### Types of Sushi

At first, you can only make the most basic type of Sushi, and better types can be made as better Fish are caught. The most basic Sushi (Onigiri – rice and seaweed) only requires Seaweed, no Fish.

#### Parameters: **Name, Time to Prepare, Sell Value**

#### Parameters are determined by random seed float 0-1 where:

**Name** is selected from a list of ascendingly complex sushi names. 10 types of sushi, 0 to <1f would be the simplest type of sushi (e.g. Onigiri), the next Name would be >1 <2 etc.

**Time to Prepare** would scale with the formula **(Seed+1)^7**

|  |  |
| --- | --- |
| **Seed** | **Time to Prepare** |
| 0 | 1s |
| 0.1 | 1.95s |
| 0.2 | 3.58s |
| 0.3 | 6.27s |
| 0.4 | 10.54s |
| 0.5 | 17.08s |
| 0.6 | 26.84s |
| 0.7 | 41.03s |
| 0.8 | 61.22s |
| 0.9 | 89.39s |
| 1 | 128s |

Game speed/feel can be tweaked (if game is too slow):

**( ( Seed+1 )^7 ) \* speed**

So speed = 0.5f would half all prepare times. There could be a frenzy bar that fills up and when reached, all staff work at double speed for a short period.

**Optional: Frenzy bar** decays over time and fills on any staff completing tasks. When full, force decay which doubles staff speed until empty.

**Sell Value** would scale with the formula **(Seed + 1.1f)^8**

|  |  |
| --- | --- |
| **Seed** | **Sell Value** |
| 0 | $2.14 |
| 0.1 | $4.30 |
| 0.2 | $8.17 |
| 0.3 | $14.76 |
| 0.4 | $25.63 |
| 0.5 | $42.95 |
| 0.6 | $69.76 |
| 0.7 | $110.20 |
| 0.8 | $169.84 |
| 0.9 | $256 |
| 1 | $378.23 |

If scale is too extreme, then **^7.5** should be used. It probably should.

#### X (upgradable) number of Chefs (start with 1)

##### Chef’s tasks:

**Collect** ingredients from **Storage**,

**Prepare** Sushi at **Cook Station**,

**Plate** dish to **Counter** (for **Waiter** to collect),

**Idle** while there are no orders.

If there is a **bottleneck**, **Chefs** should wait at the location they need to **interact** with.

#### Waiters (upgradable, start with 1)

#### Waiter’s tasks:

Take **Customer** **Orders** (instantly sent to **Kitchen**),

**Collect** plated **Sushi** from **Counter** and deliver to correct **Table**,

**Clear Tables** after **Customer** leaves (if table dirty && empty)

#### Customers

**Spawn** based on 0-1f **customerFrequency** that fluctuates throughout the **Day**.

Influenced by **Queue** length and amount in **Storage**.

If **Queue** is long then lowers **customerFrequency**.

If **Queue** is Empty, raises **customerFrequency**.

If **Storage** is low, drastically lower **customerFrequency**.

##### Customer tasks:

**Queue** for seating

If **Table** available, pick one at random and **set** new Table state.

**Flag** **Waiter** after short period (or instantly)

Pass **Order** from available **Menu\*** to **Waiter**

**Wait** for **Sushi**

**Consume Sushi** for random period (5-20 seconds?)

**Set** **Table** to **Dirty**, **Pay** and **Leave**

**\* Menu** is determined by available ingredients in **Storage.**

## Integration with Diver side

**Divers** only need to put their **Catches** into **Storage**; no other interactions are required.