Anna Sivera van der Sluijs (s3652254)

Creative programming

Programming Project: Creative Tool

SUSHI!

About the creative tool

Sushi! is a creative tool especially tailored for renowned sushi chefs who are fearless in exploring the sushi cuisine. They are on the frontlines of innovation in their field and are up for the craziest sushi combinations.

The limits of the creations of these sushi chefs are only their imagination, with co-creating tool Sushi! their options are now infinite. With the program's aid, the creative sushi chefs can browse through possible new sushi hits. By clicking on a sushi they like, similar sushi will be shown. In this way the sushi is defined in more detail with each step. The chefs associate through variables like the shape and ingredients until they find the prefect new sushi. They can start with cooking this amazing new sushi right away.

Requirements

1. 4 sushi are generated
2. The sushi consists of elements (rice, seaweed, fish)
3. The elements have properties (size, colour)
4. A sushi can be selected with a click
5. Then, one variable property of the sushi is randomly changed
6. The three new sushi are displayed

Instructions

Use Sushi! as follows:

1. Run the program
2. Click on a sushi you like
3. Keep clicking until you found the perfect sushi
4. Start cooking

Code

The runnable code can be found as **sushi.pde** in the submitted folder.

The code is structured around the three steps of the Genetic Algorithm as described by Daniel Shiffman in his book *The Nature of Code*. The steps are:

SETUP

1. **Creating a population**
   (4) sushi
   - **DNA**
     properties of the sushi like shape and colour

LOOP

2. **Selection**
   interactive selection by clicking on your favourite sushi

3. **Reproduction**
   asexual reproduction with one sushi
   - **inheritance**
     the genes are inherited
   - **mutation**
     one gene mutates for each reproduction (is assigned a random value)

Now there is a new population of sushi, return to step 2.

Example outputs

Maki sushi

Blue regular seaweed

Tuna

Uramaki sushi

Green thick seaweed

Squid

Gunkan sushi

Blue thin seaweed

Fish eggs

Temaki sushi

Green regular seaweed

Salmon

Nigiri sushi

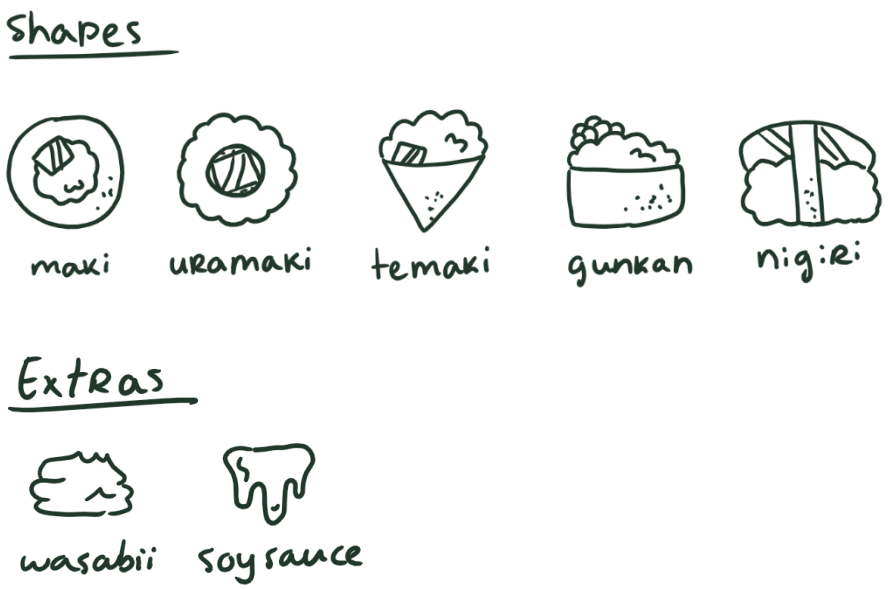
Blue thick seaweed

Squid

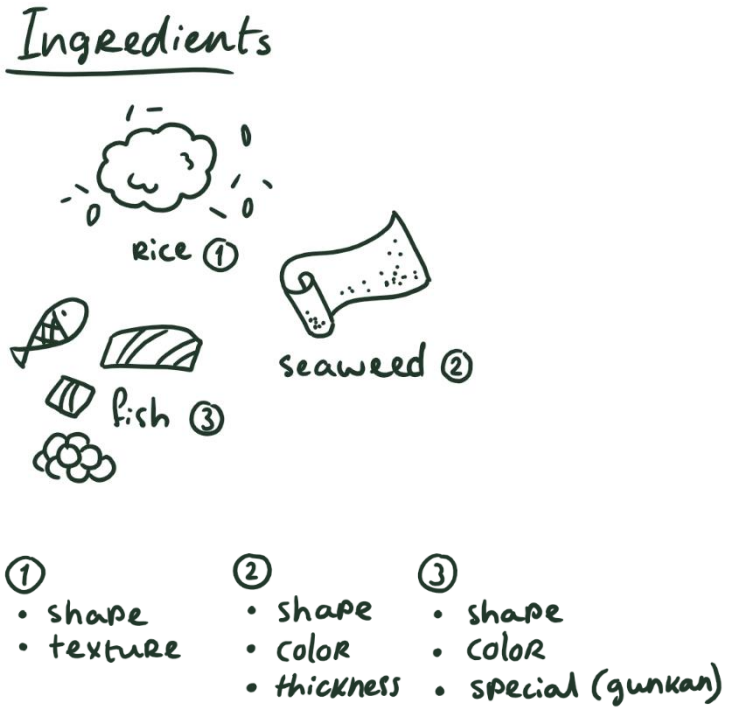
Reflection

This was a big project in which a lot of what I learned during the course is incorporated, from the random function to object oriented programming.

It was difficult to not have a starting point or example code in this project. A revolutionary sushi associator does not exists, so I could only look up specific errors or see if a certain function exists.



Therefore, I had to prepare well before coding. I thought about the structure and abstract working of the code. I sketched and wrote the structure down before I started creating the program. The book *The Nature of Code* by Daniel Shiffman also helped.



Still, the program became big and complex quickly. Steps that seemed simple written down turned out to be big challenges. An example is the reproduction of the chosen sushi, because the random looped variables where not easy to get and use for the other sushi. Wanting to control or mutate specific variables in specific looped sushi was also difficult because of the same or similar nested names. Because of this complex structure, it was also hard to keep track of the scope of variables. What helped was the use of arrays to effectively store values and the use of parameters in functions.

One of the causes of the complexity were the loops in the program. Iterating the sushi with `Sushi[] SushiInstances;` seemed effective and clean at first, but because of the looping I became very confused by what was happening in my code and where or when values were changed. It became nearly impossible to debug my code. Therefore, I at one point decided to restructure my code around separated instances of the sushi class. This caused a lot of double code for each instance (and it took some time), but it helped me to clearly understand how the code blocks interacted with each other. The comprehensive version is also included in the submitted folder as **OLDsushi.pde**. By doing this I was even able to remake the structure of the code back into a loop.

Programming is new for me and this project was definitely a (sometimes frustrating) learning process, but I am happy that I could create what I had in mind in the end. My biggest takeaway would be to structure the code beforehand as much as possible, because this gave me a base to hold on to and kept my eyes on the goal, especially during the confusing and challenging moments.

Mysterious bug?

I have noticed one bug in my code that persisted, whatever I tried. When clicking on a new sushi (not the one previously selected), sometimes the chosen sushi changes as well (it should stay the same). The order of the code or the mutate() function do not seem to have an effect on this. If known, I would like to know the cause of this bug.