mod()

an experimental game by maxi

major fix

I think the project has evolved a lot if we compare it at the stage of the prototype and now at the stage of the progress report. I am very happy with how everything is developing as most of my ideas feel doable in a visually interesting way. I rarely block on error messages or things that doesn't display properly, so everything feels like it's smooth sailing! The one big thing that made my program crash is now fixed, and I feel unstoppable! Honestly the only that hinders my progress is my other classes and a general lack of time.

Let's start with the crashes, and how I fixed it. I was calculating the density of atoms for each part using a bounding box approach (see figure 1). When the body parts are left unmodified, the surface area of the bounding box and the actual surface area of the polygon was kind of similar. There were situations created where some vertices would go super far from their original point, rendering the actual surface area way smaller than the bounding box area (see figure 2). In addition to that problem, I was checking if the atoms were overlapping, and if they were, they were repositioned. This method was fine with a bigger area, but when the area got super stretched and way smaller, it made my program crash.

The solution I found to remedy this set of problems is to run the overlap check around 90% of the time. That means we sometimes get atoms overlapping, but the overall separated and spread look I really wanted is kept almost intact (see figure 3). I really tried to make the program crash after implementing this fix, but I cannot anymore!

fig. 1.

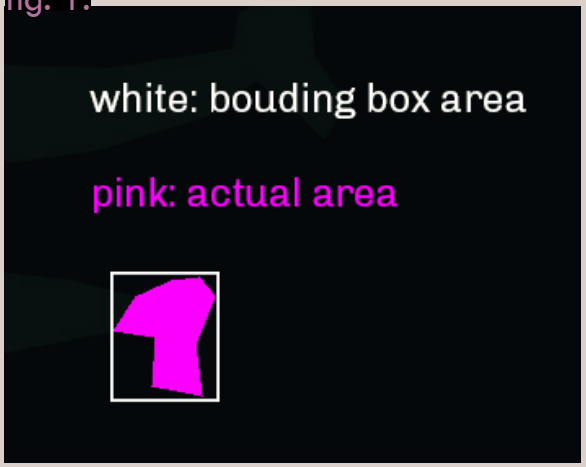


fig. 2.

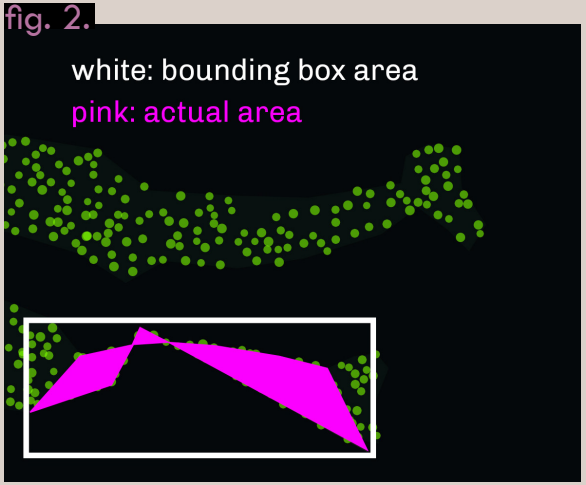
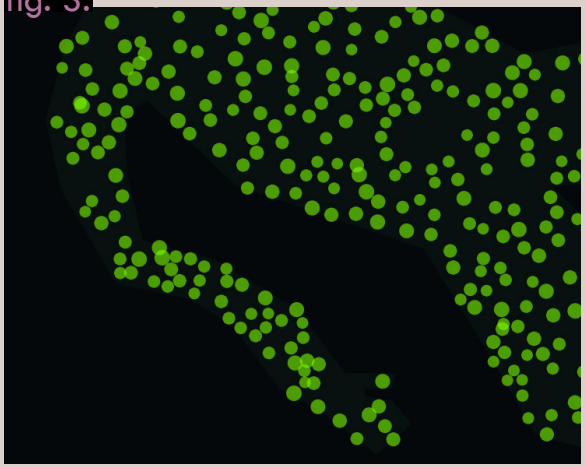


fig. 3.



additions

Other than that, I worked a lot on visuals (colors, ui, ui animation, background animation, atom animation) and I created the rest of the body parts according to the vertex plan made for the project's proposal (see figure 4).

Another big thing that I implemented is the reusability of the stretch algorithm. I started by cleaning up the code, making in more easily readable and making so it would not go through unnecessary loops using series of conditional statements. I then changed the code so you can pass a body part as a parameter. The method now checks which body part it is and know which vertices (and how many) can be modified, all because of the way I created each part (see figure 5).

With that done, I implemented a method to select and deselect the body parts. When you press the 'S' key, you select the head first, then the torso, then the left shoulder, etc. It goes through the array in order, and even though it is a bit annoying when you go too fast and 'miss' the part you wanted, I think it adds a little charm. Besides, there's only 13 parts to go through, so it doesn't take a lot of time to get back to the beginning of the loop. When you press the 'D' key, you deselect the body part. In addition to the last point, this method was a great milestone in the project, because that meant the user could control which body part would get the modification. It also made possible a 'highlight' animation using a sine wave possible, and to my eyes, it is very satisfying (see figure 6).

To finish this part of the report, I will try to list everything that has been added since the prototype:

- All 13 body parts are on the canvas and are populated by a semi-regular density.
- The atoms are now redrawn every third and eight frames, making a glitchy effect.
- Select/Deselect method.
- Selected body part animation.
- Background animation.
- Ui with instructions to discover.
- Ui animation (red lines blinking every time the algorithm is activated).
- Big crash fix.

figures on next page.

fig. 4.

fig. 6.

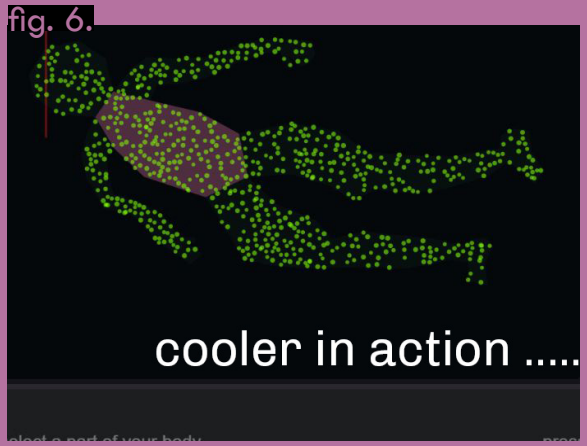


fig. 5.

```
} else {
  // define the number of vertices that will be modified (these parts have 7 not-fixed vertices)
  let numOfVerts = floor(random(2, 8));
  // checks if the bodypart is inside this array, returns true or false
  // for these parts: the first and the last vertex should not be moved (connect with other body part)
  let insideFixed2Array = checkInsideArray(bodypart, fixed2Array);
  if (insideFixed2Array) {
    for (let i = 0; i < numOfVerts; i++) {
      let currentVert = random(bodypart.perimeter);
      while (
        currentVert === bodypart.perimeter[0] ||
        currentVert === bodypart.perimeter[8]
      ) {
        currentVert = random(bodypart.perimeter);
      }
      modifiableVerts.push(currentVert);
    }
```

near future

Let's now talk about what are the next steps. Obviously, I have planned to implement states in the proposal, and I think It would now be the time to do that. I have the overall look and feel of the program and I feel like the narrative really is missing from the game.

I also want to implement some more algorithms. I have now found a way for the user to control the activation of the algorithms. In addition to that, I now understand a bit better how to code a method so I can reuse it with all body parts. I think implementing more of these algorithms will be a lot of fun and won't be too difficult considering the last few points I made. I really want my next algorithm to somehow use the name of the user, I will make a point about it!

The last thing I want to address in the near future, other than small visual tweaks, is the implementation of sound. For the last exercise, I created a generative heartbeat that is a tiny bit irregular. I want to have this sound as a background noise, and have it become louder and faster when you activate algorithms. I also created a wave sound using a sine wave and where the frequency is linked to the number of atoms on the screen. I don't know if I want to implement that sound directly into my program, but I will continue experimenting with that. Maybe one of the algorithms could be based more on sounds than on visuals?