Matthew Halpenny – 26871771
Research Evidence & UML – CART 353

Concept

The initial concept for my Creative Computation II project is an interactive art piece titled *poetry.dna*. Poetry.dna will be a website created with p5.js that monitors visiting users and mutates a selection of text every time the website receives a visitor. Initially this will be one mutation per user, but I may change this depending on results as evolution is a long process. For example, initially one visit would repopulate a selection of text altering what is displayed. I may add an interaction feature that mutates a single letter per click, for example. This ideally will not be a completely random mutation process, as in nature certain mutations are linked to fitness or have higher probabilities of affecting other mutations. This fitness model of mutation would allow for higher emergence of meaning in the text. I will be keeping private archives of all mutations to review any interesting emergences but part of the concept of the piece is to create an ephemeral experience where each site visit would yield a different text/poem encouraging frequent visiting of the artwork and critiquing the temporality of art.

Inspiration & Similar Projects

My biggest inspiration for this project comes from Eduardo Kac, whose work has been very influential to me for a variety of projects. The primary work that has influenced this project, poetry.dna, is Genesis. Genesis is “transgenic artwork that explores the intricate relationship between biology, belief systems, information technology, dialogical interaction, ethics, and the Internet. The key element of the work is an "artist's gene", a synthetic gene that was created by Kac by translating a sentence from the biblical book of Genesis into Morse Code, and converting the Morse Code into DNA base pairs according to a conversion principle specially developed by the artist for this work.” This DNA was mutated based on visitors over the web, which at the time in 1998 was much more niche.

http://www.ekac.org/geninfo2.html

The next inspirational piece I came across comes from an experimental game I played a few years ago. Although it has very little to do with my work, it deals with procedurally generated art, which before I played this game I had not come across. The game is called Secret Habitat. Secret Habitat generates a series of museums, terrain, and a mix of visual and sound based works in which you can explore. The concept of procedurally generated art is fascinating to me because it deals with a temporality not seen in most art pieces. Art traditionally is presented to us as a timeless work in which value only grows. With procedural art it must be appreciated in the moment, there is no “coming back” to a work. I feel this impermanent has the potential to create a deeper appreciation for what’s being experienced.

https://strangethink.itch.io/secret-habitat

Scope

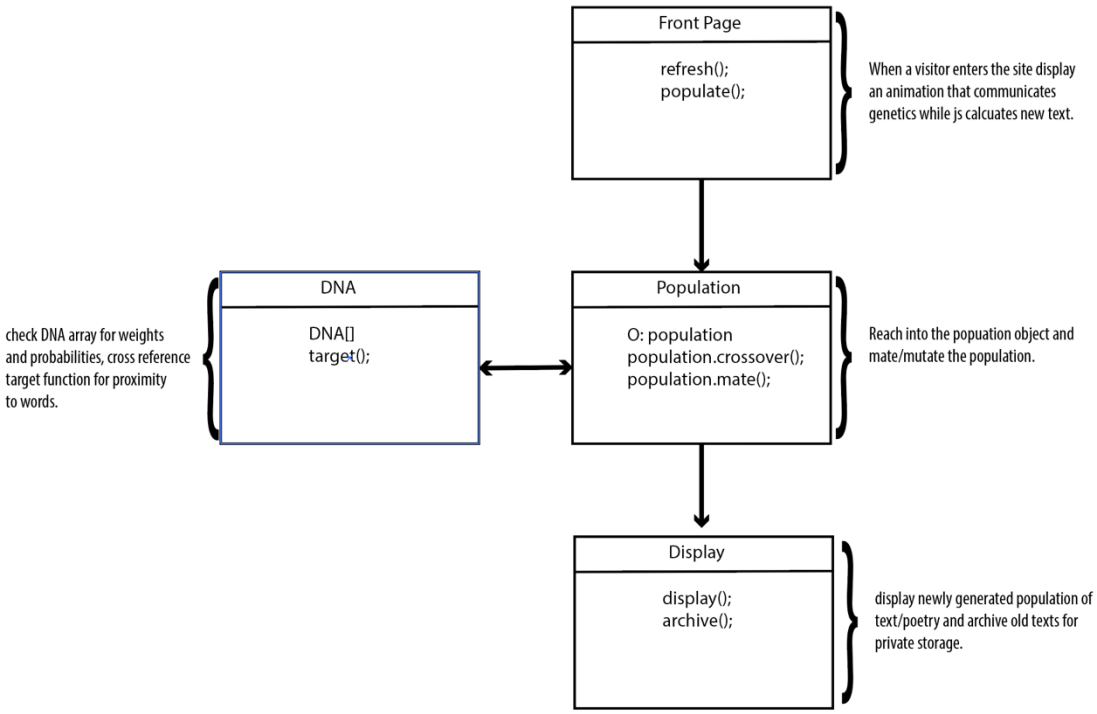
Originally, I wanted to work on evolutionary code in a gallery setting, having images evolve based on the interest level of gallery viewers without explicitly giving away information on what triggers evolution. I quickly realized having to set up LCD screens, cameras, and proximity sensors in a gallery space would be fairly challenging for a project of a few months. Additionally, the stress of finding the space and troubleshooting in it would take away a large portion of my time near the end of the project to polish off code. Instead I decided to explore evolutionary code in a web-based setting. This still allows interaction from users but gives me a much more comfortable space to work within. As I have some background with web design the HTML/CSS side won't take long, especially since I plan to keep the UI minimal as not to subtract from the artwork. The main task I will be focusing on with this piece is the actual evolutionary code and getting it to function in a way that makes the piece interesting. Since this chapter is included in our curriculum but we don't brush it until the end of the semester I will be reading ahead and working on sketches throughout the semester to speed up my understanding.

How will I code it?

My current best approximation of how to model the code would be to borrow principles from genetics and apply them to code with the help of Daniel Shiffman's chapter on evolutionary code. The first step would be to organize strings into discrete packages that would represent words. There are several different types of mutations, the first one I would explore (which is the simplest) is those that apply to a single base pair (letter in this case). In this case a letter could be added, removed, or changed. In nature the replacement is generally random but here I may add weights to mutations so, for example, if two consonants are side-by-side one may mutate into a vowel. Each letter would have a corresponding weight inside a selection array (appears X amount of times within Y) that would relate to the frequency of their usage in the English language. Another mutation method would be "crossing-over" where DNA is transferred between two different segments during replication. Here this could be represented as words swapping places within the sentence.

Evolution also relies on mating in order to utilize fitness throughout populations. I'm still working out the logistics of having several populations of text for reproduction but right now the idea is to use previous populations with high fitness (relation to real words) for mating. This would all be "under-the-hood" and would not display on the main page. An interesting method for including populations could be to assign each IP their own populations (generated upon visiting the site) that mate with the previous text displayed. This would further the unique interactions from users around the globe. If populations are not related to IPs they would be a certain number generated upon each visit and instead of mutating there would be a mating functioned called upon visiting that includes a percent chance of also mutating. In order to figure out which method works best I would have to run a few examples which I have not yet done due to my current skill level.

UML


Front Page
refresh();
populate();
When a visitor enters the site display an animation that communicates genetics while js calcuates new text.
DNA
DNA[]
target();
check DNA array for weights and probabilities, cross reference target function for proximity to words.
Population
O: population
population.crossover();
population.mate();
Reach into the popuation object and mate/mutate the population.
Display
display();
archive();
display newly generated population of text/poetry and archive old texts for private storage.