Programming Reciprocal Love: CART 253 Project Proposal

1. Introduction

For this final project, my program will be a drawing simulator in which the user co creates a drawing with the program as an exercise of *zaagi'idiwin* (reciprocal love). By dragging the mouse onto the screen, the user can create an image that changes depending on where the location of the mouse on the screen in relation to invisible objects (named Manidoo). The user is interacting with these invisible forces, effectively co creating a drawing.

The program's structure is designed around the syntactic glossing of *zaagi'idiwin*, and playfully speculates relationality between humans and non-humans by reframing the project as a co creation in addition to the intention behind named functions. For this proposal, I will explain my intentions behind the primary concept (zaagi'idiwin) and how it will be represented withing the drawing simulator, and the related classes such as Manidoo will be explained in further depth in my final paper. Additional concepts, such as relationality with human and non-humans and further linguistic speculation will also be explored in my final paper.

2. Zaagi'idiwin

The primary inspiration for this project is my overall thesis subject, *zaagi'idiwin* (reciprocal love). As my research sits at the cross section of new media and linguistics, one way I intend to engage with the subjects is through the similarities between theoretical linguistics and computer programming.

Anishinaabemowin is a language that is morphologically driven, in contrast to languages like English which are word driven. In English for example, a sentence is comprised of multiple words strung together in a specific order (subject-verb-object) to communicate meaning. In Anishinaabemowin, this is not the case. Unlike many western languages, it uses free word order, with a strong preference towards verb-initial order and verb-medial order. Additionally, Anishinaabemowin is a polysynthetic language, where some sentences can be one word because of how much information is encoded into the morphemes[1]. Below is a gloss[2] (linguistic breakdown) of the word *zaagi'idiwin*:

[1] A morpheme is a piece of a word. Not as common in English, but an example would be 'happiness' in which 'happy' and '-ness' are combined together

Zaagi’idiwin

zaagi - ’ - idi - win

treasure.3IMP.VTA-CAUS-RECP-NMLZ

The bottom row of the gloss describes what is encoded into each morpheme. The gloss can be read more linearly as:

zaagi-: treasure (third person imperative, transitive animate verb)

-‘-: causation (marks that one thing is causing something to another)

-idi-: reciprocal (marks that there is reciprocity between one thing and another)

-win: nominalization (transforms verb to noun, *-win* is specific to conceptual nouns).

From the gloss alone, we can now see that zaagi’idiwin does not directly translate ‘love’, but instead can be more accurately described as ‘a treasuring that one causes to another in a reciprocal manner’. My project intends to represent this concept, by encoding the syntactic/semantic information into the program.

3. Zaagiidiwin()

I intend to create a function labelled as zaagi’idiwin(), which embeds the morphemes described above as their own separate functions that work together. In addition to these functions working together to form one larger function, each of these functions will correspond to the semantic/syntactic morphology. As such, I will also syntactically represent them. The planned function is as follows, with a further explanation of the positions:

```
Zaagidiwin() {
    Win() {
            Zaagi() {
            }
            Caus() {
                    Idi() {
                    }
            }
```

[2] Although it is a little overwhelming to look at, the hyphens separate morphemes. The gloss at the bottom row is to describe the function of each morpheme and the hyphens correspond to the row above. Periods are used to encode further data, which is often needed for verbs for describing person, for example.

```
        }
    }
```

Zaagidiwin() is what encodes all of the ‘morpheme functions’ within the program. This will be found in the gameplay state of the program (as a title state will be used to describe the drawing simulation, so that the user knows that they can create something).

The user will be using the mouse to create a drawing on the screen, using either shapes that leave a trail, or disrupting lines (I am still figuring this out). This is the zaagi( ) function, which represents the action we are taking.

Meanwhile, the caus() function is present in the program but not visible to the user. Shapes will grow and shrink (like the garden simulation) depending on if the mouse is hovering over the shape. The user is ‘causing’ this onto the invisible shapes. As such, no drawing made in the program will be the same.

The idi() function is the consequence happening to the zaagi() function. When the user hovers over the invisible circles, the colors or lines will be affected depending on each circle. The circles will likely be their own objects, to make diverse effects. It represents the reciprocal nature of interaction between the user and the shapes. The shapes themselves take on the object title *manidoo*, which is translated in English as spirit. While this concept will be explained in greater detail in my final paper, for now, it can be loosely seen as a metaphor for our interactions with the spirit world and non-spirit world in relation to Anishinaabe epistemologies.

Finally, win() will be the function that groups these functions together under zaagiidiwin(). As a nominalizer in linguistics, it encapsulates this encoded information as a concept. It may also host other functions to ensure the program runs as intended.

The ordering of these functions will be compared to its syntactic counterpart in theoretical linguistics, and I will engage in a discussion on its similarities/differences in my final paper.

4. Technical Challenges

The program itself will take advantage of arrays and object-oriented programming, however the primary challenges of the program lies withing the aesthetic output of the drawing through the idi() function, and constructing classes that will reflect the intentions of zaagi’idiwin().

To make the program fun to use, the drawing that the user creates should be visually appealing with little effort on the user. The program uses a simple interaction with hovering the mouse to draw and invisible forces behind the ‘screen’ working to change the visual output that is dependent on location. Ideally, all aesthetic choices will be embedded in the idi() function. To create a program with numerous possibilities for drawing and making sure that each drawing is visually appealing will require a lot of balance within the variations created. Additionally, it will be challenging to create variations that are not repetitive.

Another challenge will be creating classes, and the layering of the program. While the layering can reveal interesting speculations on how we interact with objects, it can also affect the overall design of the user’s drawing. Additionally, as I intend to have a specific nesting of zaagi’idiwin()’s interior functions, it will be a challenge to assemble and nest everything correctly. At the same time, it will provide much discussion on the differences between syntactic nesting and computer programming nesting. I am also unsure if I will be using sound as part of this variation, however if so, I suspect that manipulating sound changes will also be a challenge.

5. Conclusion

While other functions will be used to make this program work, the primary focus of this project is zaagi’idiwin(). My final paper will further describe the creative process and provide arguments for the design/linguistic choices in its ordering and what is happening inside of the functions described in the proposal.