Programming Reciprocal Love

Vanessa Racine

CART 253: Creative Computation

December 7 2023

Programming Reciprocal Love

1. Introduction

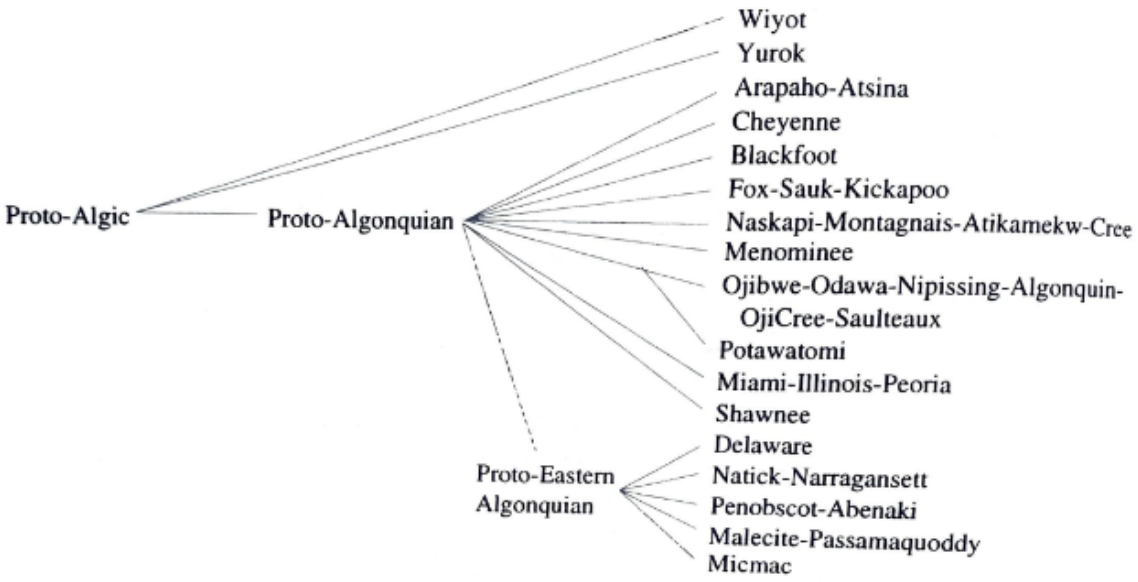
In this final project, my program is a drawing simulator in which the user co creates a drawing with the program. By dragging the mouse onto the screen, the user can create an image that changes depending on where the position of the mouse on the screen relative to an array of seven circles present in the program but invisible in its output. The program's structure is designed around the syntactic glossing of *zaagi'idiwin* (reciprocal love), and playfully speculates relationality between humans and non-humans by reframing the project as a co creation. The simulator's script, using P5.js., speculates on syntactic structures of *zaagi'idiwin* by having the simulation engage with a corresponding function, zaagiidiwin(), in which the nesting functions are the word's morphemes. Through minimal visuals using P5.js, my project intends to establish preliminary connections between computational syntax and linguistic morphosyntax through programming. Additionally, the program's output alludes to Anishinaabe storytelling with intentional naming of functions and objects.

This paper will first provide background on relevant Anishinaabemowin linguistic features (including definitions), then will outline similarities and differences between morphosyntax and programming languages (in this case, P5.js). Building on this analysis, I will then discuss the project itself in relation to *zaagi'idiwin* and zaagiidiwin() with a focus on syntactic structures followed by a brief discussion surrounding its metaphorical/epistemological motivations within the program. Finally, I will discuss the limitations of bridging linguistics and programming, specifically within this project.

2. On Anishinaabemowin and P5.js Syntax

First, some clarifications will be made. Anishinaabe is the people and nations, where Anishinaabemowin is the language. This language falls under the Algonquian language family (see image), and encompasses local names like Ojibwe, Chippewa, Odawa and Algonquin. The image below is the Algonquian family tree, taken from Valentine's *Nishnaabemwin Reference Grammar*[1], to provide additional context to where these languages are placed in relation to other Indigenous languages.

[1] Valentine R. (2001). *Nishnaabemwin reference grammar*. University of Toronto Press. p. 14



ALGIC LANGUAGES (AFTER BLOOMFIELD 1946, TEETER 1967, TODD 1970 AND GODDARD 1979)

For the context of this project, there are two areas of Anishinaabemowin that I will be focusing on: (1) morphology and (2) syntax. These areas can combine into morphosyntax, but in this section I will be discussing each separately for ease of reference. Going forward in this paper, when I reference morphosyntax, I am discussing the relationship of these two subjects together.

Anishinaabemowin is a polysynthetic language, meaning that its morpheme-to-word ratio is very high in comparison to other languages like English, which has a high inflection and higher ratio of words-to-morpheme. Inflection is how words change to express grammatical categories like tense, case, person – this is done through affixation. Morphemes are minimal pieces that express meaning and can be defined as a "system of adjustments" that contribute to the way speakers intend their utterances to be interpreted [2]. 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. Phrases can be constrained to one word, or a single word can be made up of multiple morphemes. The word for blueberry pie,

[2] Payne, T. (1997). *Describing Morphosyntax: A Guide for Field Linguists.* Cambridge: Cambridge University Press. doi:10.1017/CBO9780511805066 p.20

miinibaashkiminasiganibiitoosijiganibakwezhigan [3] is a famous example of Anishinaabemowin's morphology as it also encodes information on how to make one.

As part of morphology and syntax, it is important to define what a root and stem are. A root is the uninflected base of a word, or part of a word in which affixes are added on. A stem is an inflected version of the root. English does have complex roots and stems, for example, *corrupt* is a stem and *rupt* would be the root. However, *rupt* is not actually grammatical in English, but can be inflected to *disrupt, corrupt, erupt,* among other stems.

In addition to the morphological aspects of the language, Anishinaabemowin is a verb-driven language. Concepts expressed as nominals in English such as days of the week, time, expressions of weather and seasons, are verbs in Anishinaabemowin[4]. These verbs are highly inflective and take on different forms based on transitivity and animacy. Transitivity is if/when a verb accepts one or more objects (Pippin eats apples vs. Pippin rises), whereas animacy is grammatical gender that is determined by the participant. In the scope of this project, animacy is irrelevant due to the constraints of P5.js.

This simulation is made through P5.js. and follows this library's restrictions first and foremost. It is a library for creative coding and made for drawing and interacting with shapes (although much more can be done with it with some creative brainstorming). More generally, programming is action-based; the user inputs functions and executes a program that can be read by a computer. This comes as an advantage when transmediating Anishinaabemowin morphosyntax into P5.js because it is a verb-based language. Much like Anishinaabemowin, functions act like verbs and the category of verbs in P5.js can express color, shapes, and numbers. My inspiration for this project comes from the nesting of functions within P5.js in which it has similarities to how Anishinaabemowin strings morphemes into words/phrases.

The work done in this program is made through speculative decisions based on linguistic theory and is not related to computational linguistics or computational semantics. Language is not being processed in this project, nor is Anishinaabemowin being translated. Instead, I use morphology and syntax as a creative medium within the constraints of P5.js.

---

[3] "Miini-Baashkiminasigani-Biitoosijigani-Bakwezhigan." *The Ojibwe People's Dictionary,* Edited by Nora Livesay and John D Nichols. ojibwe.lib.umn.edu/main-entry/miini-baashkiminasigani-biitoosijigani-bakwezhigan-na

[4] Valentine R. (2001). *Nishnaabemwin reference grammar.* p. 130

3. Transmediating *Zaagi'idiwin* to zaagiidiwin()

The primary inspiration for this project is my overall thesis subject, *zaagi'idiwin* (reciprocal love). The simulation created for this final project is an imitation of zaagi'idiwin. For the program to feel reciprocal, the user must suspend disbelief and assume that dragging their mouse over a blank canvas triggers unseen forces that change the output of the drawing. Simply, the mouse hovers over an array of circles that shrink over time, triggering it to grow while also changing variables of the shape that is displayed. Describing this as a reciprocity is not entirely true because the output given by the user's interaction is programmed to do so. There is no mutual benefit that is created in this interaction, the program gives what it has been designed to give. However, for this level of project and programming, it is always fun to imagine and the imagining of this relationship can still be effective in understanding reciprocal relationships.

In this section I will discuss the two primary relationships. First, how I designed the program around morphemes using linguistic glossing and then the ordering of these functions using insights from linguistic syntax.

3.1. Relationships to Morphemes

Below is a gloss[5] (linguistic breakdown) of the word *zaagi'idiwin*:

zaagi'idiwin
zaagi - ' - idi - win
treasure.3IMP.VTA-CAUS-RECP-NMLZ

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).

[5] 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 to describe inflection.

Zaagi-, glossed as the verb 'to treasure', is the stem of this word. The root of *zaagi-* is zaagi'. The other morphemes of this word are suffixes. To differentiate the root from the other morphemes, I chose to represent this as an object within Object Oriented Programming. By calling this object in zaagiidiwin(), it serves as a 'pseudo-root' for the function when comparing at face value. Within this rudimentary approach in Object Oriented Programming, a root presented as an object This object in the program is how the user is interacting with the rest of the elements – the variations in the mouse's output are all stored within this object (almost like a treasure box).

The rest of the morphemes are labelled as functions. One complication that arose from creating a 'mirrored' version of the word was the morpheme representing causation, which appears as ' in zaagi'idiwin. As such, I opted to use the gloss, caus, as the name for this function. The caus() function triggers growth of the Manidoo array, stored as an object (discussed in 3.3). When the mouse hovers over the array of circles, the user is causing the circles to grow. Overall, this function is simple, but allows for more time within the program as the circles are set to shrink and disappear over a time.

The idi() function is a function that represents reciprocity, as per its gloss. Reciprocity is mutual benefit, and this is where the program becomes an imitation since as previously stated, the program does not actually benefit from anything the user does. Inside of the idi() function is a switch case, where each instance of the array is assigned a function from the Zaagi object in order to change the output of the mouse interaction. The array 'reciprocates' different shapes, colours and sizes in the drawing.

Finally, the win() function is a loop, in which elements in the array are looped through and displayed. In Anishinaabemowin, *win-,* is a specific nominalizer; it is used to produce "nouns of action, product, place and instrument" [6]. It is commonly used for concepts in Anishinaabemowin, such as the concept of reciprocal love. It is quite difficult to have nominals in programming when functions are default action-based – at least in the scope of beginner's P5.js. Thematically, having win() as the loop groups together the caus() and idi() function, but win()'s position within my program does have linguistic significance over computational significance due to its nesting within zaagiidiwin().

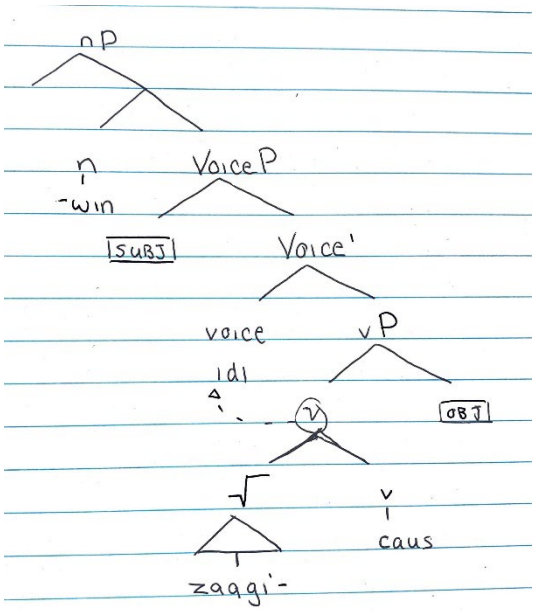
[6] Valentine, R. (2001). p.505

3.2. Relationships to Syntax

The function that encompasses the simulation is titled zaagiidiwin(), and is the only function that is found within draw(). This was an aesthetic choice, for minimalism, but also to begin imitating the syntactic structure (linguistic) of *zaagi'idiwin* in my program.

To premise, the syntactic structure I made of *zaagi'idiwin* is flawed. There are some aspects that I still need to research. First, the placement of causation is still being discussed within the field of linguistics. Legate's analysis on voice and causatives suggests that VoiceP must be distinguished from vP, in which causation is often found in vP[7]. Legate argues and presents evidence through Acehnese that the causative v and Voice are both independently and simultaneously realized and argues that the causatives are embedded under v_{caus}, rather than within vP. In short, it is very uncertain as to where causatives are placed. Even within Legate's analysis, it may not capture the behaviour of causatives in Anishinaabemowin because her analysis implies that it is outside of the root/stem. In Anishinaabemowin, *zaagi'* is the root, which has the causative attached [8].

The syntax tree below (made by me with assistance from Sigwan Thivierge), provides a loose derivation of zaagi'idiwin:



[7] Legate, Julie Anne. *Voice and v: Lessons from Acehnese.* The MIT Press, 2014. P.111

[8] If this is going over your head do not worry it is advanced syntax and is confusing to me too. All to say, no one really knows where causatives go. A whole thesis can be written, as Legate only presents this argument with Acehnese.

What is going on in the syntax tree regarding the labels and arrows is irrelevant for the project discussion[9], as I will be focusing on the position of each morpheme. Notably, *-win* is at the top of the tree in a separate node from voiceP. This makes sense as the nominalizer transforming the verb into a noun, and the morphemes attached to the verb are under the same branch. Where this tree becomes speculative is the placement of the causative. I placed it closer to the root, however it may be higher up in the tree, according to Legate's analysis of causatives.

Below was my intended function of zaagiidiwin():

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

Unintentionally, my intended function suggests that the causative and *-idi-* are closer together syntactically, as I made this during my proposal when I did not have a syntax tree made. The motivation behind this decision is that a person *causes* reciprocity. In hindsight, this prototype may be wrong as reciprocity is something that is caused. This would reverse the position in the prototype and it becomes a step closer to the syntax tree representation. However, since my initial prototype, the final code is very different as I introduced OOP (see image below).

[9] The small arrow in the tree is deep structure movement. If you are interested in what movement is, looking at wh-movement in English is an example that shows how questions are formed.

```
function zaagiidiwin() { // draw
    if (state === `title`) {
        push();
        background(255);
        textTitle(); // displays title text
        keyPressed(); // press enter to go to game state
        pop();
    }
    else if (state === `game`) {
        push();
        win(); // loop through all the manidoog in the array and display them - contains caus() and idi()
        pop();
    }
};

// loop - found in draw -- changes in array
function win() {
    zaagi = new Zaagi(mouseX, mouseY, size);   // display zaagi object
    for (let i = 0; i < universe.manidoog.length; i++) {
        if (universe.manidoog[i].alive) {
            universe.manidoog[i].shrink();  // shrinks the circles over time
            universe.manidoog[i].display(); // displays the manidoog array
            caus(universe.manidoog[i]); // circles grow when mouse is hovering over
            idi(i); //

        }
    }
}
```

In this preview of code, more functions are needed for the program itself to work, but focusing only on the morpheme functions shows that within zaagiidiwin(), win() nests Zaagi(), now and object, and inside of the for-loop is caus() and idi(). The nesting within win() does resemble the tree to an extent. Zaagi() is outside of the for-loop, and when compared to the syntax tree, the root branch is also separate from the other morphemes. In keeping with my initial placement of caus() and idi() from the prototype, it also suggests that they are closer together within syntactic structures. In this program, both functions need to be in the for-loop for the intended output but the order in which they are placed does not change the outcome.

4. Metaphor and Storytelling within Programming

While making this program, I wanted to include some storytelling within the programming itself rather than the output. Partly motivated by classroom discussions on what is ‘visible’ in a program, I wanted to expand on this by engaging with the idea of two worlds. Within Anishinaabe frameworks, the spirit world is one that is not visible to us. Tricksters like Nanabush can pass through both seamlessly, and red is often associated with spirits as it is the color that can exist in two worlds.

The invisible array of circles is a simple metaphor for representing two worlds. The arrays are not visible to the human eye but present, much like the spirit world. In Anishinaabemowin, the word commonly known for spirit is *manidoo,* with *manidoog* being its plural form. I decided to name my

object as Manidoo() as a way to bridge the classroom discussion with storytelling. Adding to the discussion of invisibility, line 52 in the Zaagi() object has a console log with noting “possibility lives in nothing”. This is an easter egg of sorts and provides further insight to Anishinaabe understandings of creation. Gitchi-Manidoo is known as the great spirit who brought life into creation from nothing. Other translations, and the one that I am used to, is not great spirit but rather great mystery. By interpreting creation as being made from a great mystery, it comes to an understanding that in nothing, in silence, or in the dark, is possibility. It is the idea of possibility that brings life and new things. Therefore, despite the user seeing nothing, as even the canvas matches the typical white background of a screen, possibility can grow (and shrink) within the canvas.

The program, as mentioned before, is an imitation of reciprocal love. Love, as we know, comes in many forms and can be gentle. The love in this program is on the gentler side, and while I as the programmer am aware that the reciprocation is fabricated (and not actually reciprocity), users who are unaware of the program will experience an initial surprise when their interaction does cause changes in the output. It is that moment of surprise that ultimately captures the essence of reciprocity, even if it is spoiled moments later when the belief is no longer suspended.

5. Limitations and Considerations

In this final section I will outline additional technical and theoretical limitations and considerations with weaving together linguistics and programming, especially as it relates to Anishinaabe worldviews.

P5.js is constrained. As presented in the syntax and general overview of the language, there are elements that it cannot capture linguistically. The primary being nominals, and then extending into finer details like grammatical gender, which in Anishinaabemowin presents differently from that of western languages. Grammatical gender is categorized by animacy, which is influenced by the surrounding environment and context. Computers by default, are only found within one environment (usually indoors) and cannot account for human-generated contexts like storytelling and spirituality.

At the same time, computers are made of minerals and stones. Within the framework of animism and grammatical animacy, what is experienced as living is different from that of western perspectives. Some stones do have animacy, implying that the stones and hardware made for

computers can exhibit the same[10]. Our interactions with computers then, should be accounted for as an animate, transitive action. Using grammatical animacy is also problematic in this speculation, as it is somewhat a myth that animacy equates to spiritual importance / aliveness. For example, the word for strawberry – an important berry in Anishinaabe culture – is categorized as inanimate. However, grammar alone cannot account for animacy within Anishinaabe epistemology. In a very long quote taken from Cecil King, he states:

> "Does animate then mean having or possessing a soul? Is this a sufficient explanation? I think not. Is the animate-inanimate dichotomy helpful in describing the structure of my language? I think that it is limiting, if not wrong outright. For in Odawa anything at some time can be animate. The state of inanimateness is not the denial or negation of animateness as death is the negation of the state of aliveness. Nor can something have a soul and then not have a soul and then acquire a soul again. In Odawa the concept of animateness sis limiteless. It can be altered by the mood of the moment, the mood of the speaker, the context, the use, the circumstances, the very cosmos or our totality."[11]

Yet, the limitation of programming still creates a computational wall and is still deterministic. Simply, this program is not sophisticated enough to be considered as reciprocal, but it can be speculated that more advanced programming (perhaps through deep learning AI) can build that bridge.

6. Conclusion

Within the wider scope of my thesis, this course project provides a first undertaking in multidisciplinary research creation. This simulation introduces users the concept of two worlds (ours and spirit) through the teaching of zaagi'idiwin with metaphor. Within this simulation, the user is interacting with 'manidoog' (spirits). For the duration of the simulation the user can create a unique image and by extension, interaction, and reciprocity. Although this metaphor is incomplete due to the program language's constraints, expressing language through programming is a way in which I can blend syntactic understandings of Anishinaabemowin in my proposed research creation project.

---

[10] Kite, Suzanne (2023) *Hél čhaŋkú kiŋ ȟpáye (There lies the road): How to Make Art in a Good Way.* PhD thesis, Concordia University.

[11] Valentine, R. (2001). p.119

Bibliography

Kite, Suzanne (2023) *Hél čhaŋkú kiŋ ȟpáye (There lies the road): How to Make Art in a Good Way.* PhD thesis, Concordia University.

Legate, Julie Anne. “Chapter 5: Voice and Causatives”, *Voice and v: Lessons from Acehnese*. The MIT Press, 2014.

“Miini-Baashkiminasigani-Biitoosijigani-Bakwezhigan.” *The Ojibwe People’s Dictionary,* Edited by Nora Livesay and John D Nichols. ojibwe.lib.umn.edu/main-entry/miini-baashkiminasigani-biitoosijigani-bakwezhigan-na

Payne, T. (1997). *Describing Morphosyntax: A Guide for Field Linguists*. Cambridge: Cambridge University Press. doi:10.1017/CBO9780511805066

Valentine R. (2001). *Nishnaabemwin reference grammar*. University of Toronto Press.