**Kairen Building**

x: means example:

Now that I wrote a lot of stuff concerning the typology of the language, I should start making all the words. I will work from the most probable to the least probable, that way, I am most efficient. I can make a distinction between things that I am probably not going to change and things that are more up in the air. I will also be able to make multiple options for all of them here.

This is probably why it is good to have the final/semifinal version on computer, it is easier to go back and add different ideas/methods to areas of thinking.

**Verb Conjugations**

Verbs are present progressive by default.

The following are verb conjugations

* past progressive: -a
* future progressive: -u
* negative: -n (when there is consonant, -o)

However, this means that there will be cases in which there will be two vowels in a row:

Let “sa” = see

x: I saw, I was seeing = saa

There are 3 possibilities for double vowels

1. elongate vowel
   1. con: ambiguity with short vowel sa
      1. possibly even more ambiguity if there is a vowel sentence particle for questions
      2. x: saa a?
   2. pro: most logical pronunciation
2. glottal stop between vowels
   1. con: sounds awful to me at least
      1. possible ambiguity if there is a vowel question particle:
         1. x: saa a? = sa’a a?/sa’a (‘) a?
         2. also sounds gross in this case
   2. pro: easy fix (although not that easy)
3. add buffer consonant in between; x = r
   1. x: saa -> sara
   2. con: adds more characters ( :( )
   3. pro: sounds the best
      1. less ambiguity
   4. If I do this fix, I need to remember not to have the most common vowels end in a or u because they will have this buffer consonant added in the past/future cases.

5. here would be the conjugation chart of sa if we take method 3

|  |  |  |
| --- | --- | --- |
| sa | sara | sau |
| san | saran | saun |

sa method 2:

|  |  |  |
| --- | --- | --- |
| sa | sa’a | sau |
| san | sa’an | saun |

sa method 1:

|  |  |  |
| --- | --- | --- |
| sa | saa | sau |
| san | saan | saun |

x: ku method 3

|  |  |  |
| --- | --- | --- |
| ku | kua | kuru |
| kun | kuan | kurun |

Honestly, method 3 looks fine. I’m pretty sure that I’m going to have to have buffer consonants in other situations as well. So I might as well use it here.

1. A situation where I might need a buffer consonant is if I am adding a suffix and there is another double consonant
   1. x: ent = the quality of this adjective
   2. ke = cool
   3. keent = coolness.
      1. should I just leave it like this, or should I add a buffer consonant?
      2. the difference here is that there probably isn’t a “kent” in the language for this word to be confused with, whereas there would be confusion between “sa” and “saa”
      3. I think I should actually just leave it like this
      4. therefore, this is a situation where I would not need a buffer consonant

It is possible that indeed there is a situation where I would need one, but I can’t think of it right now.

Ok, I feel good about this now

Conclusion:

* **past progressive: -a**
* **future progressive: -u**
* **negative: -n (when there is consonant, -o)**
* **add r between double vowels**

What does this mean for word building? Well, to avoid ambiguity, I probably should avoid words that would look like these vowels.

Here are some possible verbs to give me an idea of what to avoid:

sa (verb ending in a)

|  |  |  |
| --- | --- | --- |
| sa | sara | sau |
| san | saran | saun |

ku (verb ending in u)

|  |  |  |
| --- | --- | --- |
| ku | kua | kuru |
| kun | kuan | kurun |

si (verb ending in other vowel)

|  |  |  |
| --- | --- | --- |
| si | sia | siu |
| sin | sian | siun |

tak (verb ending in consonant)

|  |  |  |
| --- | --- | --- |
| tak | taka | taku |
| tako | takan | takun |

CVn, CV(r)a(n), CV(r)u(n), CVCo/a(n)/u(n)

that’s what I should avoid

I’m a little sad because I kinda like words that end in “n” and I like a and u and r like jeez these are all likable letters.

It just means that these likable letters will be in words that come up often! (verbs) So I should actually be happy about this.

I’ll just have to be more creative when coming up with the other words

EDIT: I’m rethinking this section….

To make it easier for me to make words, and for this to be learned, what if I just make the negation ending: -o no matter what.

Here’s what that would look like:

sa (verb ending in a)

|  |  |  |
| --- | --- | --- |
| sa | sara | sai |
| sao | sarao | saio |

ku (verb ending in u)

|  |  |  |
| --- | --- | --- |
| ku | kua | kui |
| kuo | kuao | kuio |

si (verb ending in other vowel)

|  |  |  |
| --- | --- | --- |
| si | sia | siri |
| sio | siao | sirio |

tak (verb ending in consonant)

|  |  |  |
| --- | --- | --- |
| tak | taka | taki |
| tako | takao | takio |

I’m using the -i ending for the future tense in this case. This makes it easier to some extent to detect the differences in pronunciation in takao and takio. This seems a lot easier to memorize, and I’m actually going to replace the previous stuff with this method. This means that I got to change the first person personal pronouns from i, is, ir to u, us, ur. I might want to change the second person too to make them more different

edit jun 21:

su (verb ending in u)

|  |  |  |
| --- | --- | --- |
| su | **suru** | suim |
| suo | **suruo** | suimo |

ki (verb ending in i)

|  |  |  |
| --- | --- | --- |
| ki | kiu | **kirim** |
| kio | kiuo | **kirimo** |

co(verb ending in o)

|  |  |  |
| --- | --- | --- |
| co | cou | coim |
| **coro** | couo | coimo |

le (verb ending in other vowel)

|  |  |  |
| --- | --- | --- |
| le | leu | leim |
| leo | leuo | leimo |

tak (verb ending in consonant)

|  |  |  |
| --- | --- | --- |
| tak | taku | takim |
| tako | takuo | takimo |

**Compound Word Formation**

I know this isn’t on the typology list, but I’ve been thinking about this a lot and how to do it right and ugh I can’t stop thinking about it its been years.

SO

I want people to be able to come up with their own words for the language really easily because I want to have that personalization factor involved. German is often praised (especially on the psychology channel) for allowing such words and using them in common conversation. It improves people’s mental health for them to have such precision with expressing themselves and how they feel clearly; it’s almost cathartic when you find out that you have the right word to describe how you feel. I really want to create that experience for someone learning this language.

So there are a number of methods with compound word formation (one of which I had been using for a while until I realized it would be difficult for people to remember what the word meant because it contained so few of the original parts (I can actually use that as an abbreviation system! It’s not completely pointless!))

Here are scenario words and scenario letter merger points:

1. nit + bel, C + C = nitbel
2. kai + ren, V + C = kairen
3. ren + aku, C + V = renaku
4. kai + oyu, V + V = kaioyu

The way I combined them up there was the simplest way, just sticking them together (METHOD 1). This is what most languages use. I think there are two problems with this

1. C + C can lead to difficult words to pronounce. However, this isn’t really an issue because people can suck it up LOL
2. the larger issue is: If I decide to make Kairen a copula dropping language (which would give a lot more brevity), there would be an ambiguity problem
   1. x: nit = night (n), bel = beautiful (adj)
   2. x: nitbel = night beautiful [beautiful like the night] (adj)
   3. nit bel = the night is beautiful
      1. I don’t know if this is a real problem in this case. You wouldn’t just say “nitbel” as a sentence. Actually! you might if someone asked you to describe something
      2. x: how is she?
      3. nitbel~~ (she is nitbel, NOT the night is beautiful)
   4. So this is actually an issue

Because this is an issue, the way to solve it might be adding a letter between the words as a compound merger letter. This would make the compound word and the equivalence sentences distinct.

Method 2:

1. nit + bel, C+C -> C + e + C, nitebel
2. kai + ren, V + C -> V + e + C, kaieren (I’m adding a vowel instead of a consonant here because there is no way to make sure that the consonant pronounceable)
3. ren + aku, C + V -> C + e + V reneaku
4. kai + oyu, V + r + V kairoyu

This method is actually pretty confusing. I think the only time you need to do this is when you are combining words that could be in an equivalence sentence, x: n + adj. You probably won’t need to do it for kai + ren, n + n. But actually i’m wrong, there are times where you might want to have two nouns in an equivalence sentence.

x: love is pain

x: lovepain could be a compound word

Some more thoughts:

Probably the ambiguity between nitbel and nit bel is an edge case. There are things that would improve distinction:

1. context: probably you wouldn’t say nit bel when asked how is she. the situations are very far apart
2. intonation/spacing: you probably would say nit … bel vs nitbel and there would be more intonation on each word individually.

That’s only the speaking side. Remember that this language is going to be largely used textually, and the strings “nit bel” and “nitbel” are different and easy to tell that they are different!!

So why all this worrying? goodness hakan

Another thing I was thinking about was whether to have hyphens between the words in a compound word to make sure that the user knows it is a compound word and so that I don’t overlap a compound word with an actual word

x: ren + aku = ren-aku vs renaku bc renaku might be a word

I think this is unnecessary. I will just write them together.

OK, so I’m choosing method 1, and I justified how this would not create a problem with copula dropping. So now let’s conclude and move to copula dropping

Conclusion: compound words are formed by putting words together with no spaces. go figure

**Copula Dropping**

So the issue here is mostly with the computer. If we drop copulas, it will become more difficult to program an AI to recognize an equivalence (i think). For now, let’s just say that the goal is RECOGNITION OF VERB.

1. I’m saying this because I don’t know yet what the ai is going to do with that information, or what the AI will think of it, or ahafoiaweljkf;lsd all that nonsense
2. I’m just trying to recognize now. so GOAL = ai recognizes that there is an equivalence verb “si” there.

What are the arguments of the verb si? well we can equate some things:

1. noun adj: the car is red. this is akin to saying “the car’s color is red”
2. noun noun: love is pain
3. adj adj: red is beautiful
4. adv adv: quickly is efficiently

you can do this with any combination of parts of speech! hmmm, there must be something else.

Essentially, the sentence structure is = [thing a] [thing b]. so the problem becomes how do you find out where the gap is between the two things. The advantage with the copula is that you are explicitly telling the ai where the gap is, and that there is an equivalence being made.

You can also equate more than one thing easily with a copula.

the car is beautiful is wonderful.

Actually, this sentence has ambiguity.

Noun = adj = adj

1. car is beautiful and car is wonderful, noun’s property = adj, noun’s property = adj
2. car is beautiful and beautiful is wonderful, noun’s property = adj, adj = adj

so I don’t know for sure….

mathematically speaking, this sentence appears like

car is beautiful and beautiful is wonderful and car is wonderful (transitive property). but car and beautiful and wonderful are not all the same thing…

Maybe there should be two different verbs:

si = property equivalence (noun’s property = adj)

sa = direct equivalence (“is the same as”)

so x: car sa beautiful and car sa wonderful is false

but beautiful sa wonderful would be legal

but that definition of sa wouldn’t be useful because there are few synonyms in kairen. it would make more sense as just “is” but locked to the same parts of speech.

x: car si beautiful si wonderful = car is beautiful and wonderful

x: car si beautiful sa wonderful = car is beautiful and beautiful is wonderful

but this brings up a good point, why not just say “car is beautiful and wonderful” XD

sigh

I think the better solution for the computer (and our minds) would be to ditch “sa” and do this:

sentence 1 parse:

car si beautiful si wonderful = car is beautiful

car si beautiful si wonderful = beautiful is wonderful

car si beautiful si wonderful = car is beautiful is wonderful (car is beautiful and beautiful is wonderful)

sentence 2 parse: let en be and

car si beautiful en wonderful = car is beautiful and wonderful (car is beautiful and car is wonderful)

so I just cleared up confusion with having multiple copulas in the same sentence. I think I’m set on keeping the copula. It will just make programming so much easier.

pros of dropping copula: brevity, especially for cslike.

response: but you probably are going to drop other grammatical features when making callouts anyways! “1 (is) (in) tunnels”. Unless I’m going to allow you to drop “in” in regular speech. That seems crazy but lets consider that for a moment

1. when you can drop in: when in can be inferred.
2. Hakan is (in) office; noun = noun
   1. the difference is that office is a location. there is no way that I can be an office. EDIT: yes there can: if there is an office called “hakan”
   2. This means that I would have to define a set of nouns that are locations. ugh
      1. this is possible with a map of a cslike where there are a limited number of locations for people to be in and they are all callouts.
      2. I would have to list them anyways because the map would need a set of string for all the callouts (actually i might not need to but it’s good to have that data stored somewhere just in case)
      3. then the ai parses:
         1. noun is noun2
         2. check nouns against locations
         3. noun2 = location
         4. noun is (in) noun2 / however you want to process this callout

But in an open world game or something with a myriad of places, this is probably not as easy to do? Or maybe…

1. playing open world game, lots of types of houses
2. “he’s in red house”
3. for the ai to process that and conclude with the mental information of him being inside that house (let’s say its a zombie and he’s an enemy), it needs to know that house is a location right?
   1. because it would find “house” in a list and see that it’s a location
   2. OR it would just use “house” in a find objects within proximity query and find the red one of them,
      1. that means that all the objects that are houses must either
         1. be named house in the hierarchy
         2. or have a script attached that defines the type of object = house

method 2 is actually a lot more fluid than method 1, and is probably how I would develop an open world game with kairen. However, if i have a script attached to each object with a set of properties, couldn’t I just have “location” as a property?

so “he is red house” would be legal in this case………………. hmmmmmm

ugghhhhh

idk

the only time when this would be for sure an issue would be when you are using a person’s name

1. x: john is red house
   1. is john in red house?
   2. or is john the red house (is red house’s name john?)

but that is kinda an edge case??? maybe?? maybe not considering you don’t have a list of common names so there is no prior information to tell whether it is one case or another.

I think this is a little silly but is probably a good enough reason to not allow “in” dropping. This will probably be the case for a lot of prepositions.

NOT TO MENTION: what if john is on the house or under the house. “in” might not always be the most common preposition! (although I suspect it is)

1. there might be a conversation where it can be inferred from the conversation that a person is under the house (john is a plumber ….), and a user might say “john house” to mean john is under the house. but it would be difficult to set up an ai for inferring which preposition the user is using
2. you can combat this by making the default preposition (when there is none) to be “in” but probably it is not going to be easy to remember for people.
3. in addition, it is not a big deal to require a preposition. cslike gamers are going to drop it either way, because that is a specialized case and the grammar is very exact.

**Conclusion!!:**

In correct grammar:

1. no copula dropping
2. no different types of copula
3. actually nvm, I could make a “is the same as” verb, but it would not be just an “is” that is restricted to only the same parts of speech
4. when there are multiple copulas, each pair is processed individually. If you want more than one item to be equated, use ‘and’
5. no preposition dropping

in cslike grammar:

copula dropping

preposition dropping

probably a host of other things that break grammar (although I would like to minimize them so that people don’t learn too many bad habits that they would have to break when they use kairen in a different context!)

what elseeeee

*4 demonstratives*

*2 directional systems (local and world) each with 6 words*

*evidentiality particles*

*personal pronouns*

*everyone, anyone, etc pronouns*

*yes/no/maybe*

*gradient words*

*start and end verbs*

*causation constructions (make)*

*relative clause marker (that)*

*etc*

so there is a lot more to do. Strat: define the words that will be needed most often in cslike so that they are the shortest they can possibly be. Then continue to define the rest of the words.

**Gradient Words (starting with adjectives (there might only be adjectives))**

So I made a large graphic in Sketch and realized that i was working too slowly to find the optimum solution for this issue. I finished up some examples, was dissatisfied with the solution, and I got out my notebook to work faster to find the right solution. I think I finally did, so I’m going to spare myself typing all my thought process and instead type just the final product. I will attach the sketch file however, because that is digital and it’s easy to do that stuff.

UGH I finally came to the solution and its what any old shmo would come up with.

OK, so first you must acknowledge that there are two types of gradient adjectives (or maybe all adjectives are gradient adjectives, actually nvm, x: on off, there are some adjectives that are state switches)

There are infinite (big, small) and finite (brightness on a computer (0 - 255)) adjectives.

This solution accomodates both. I will make a sketch document showing it now ahahahahaha.

This is basically the best solution I have for the Kairen gradient words issue.

**CONCLUSION:**

1. **-o makes an adj opposite**
2. **-e makes an adj middle**
3. **ve means very**
4. **vo means a little**
   1. under the hood: ve means push out by .5 unit distance of previous layer
   2. vo means push in by .5 unit distance of previous layer

I will bring up a physical paper that includes information about the current “final” status of the language, that way, it is easier to make quick changes….

Actually, I’m going to just make a document:

**Common words in a cslike**

1. Personal pronouns
   1. I is very common because it’s what you have 100% certainty of
   2. 2nd person is difficult because how does ai know who you are talking to…
      1. if game’s communication is text chat at bottom left of screen, you won’t use “you” much
      2. probably you will use names of other players, and a word for enemy (maybe)
      3. names of teammates must be as short as possible
         1. use letters
         2. if game is 3v3, then you only need 3 letters: k, p, t or something
         3. then a word for enemy (maybe)
         4. if 5v5, then need 5 letters: k, p, t, f, v or something
            1. probably choose consonants

this means we have to talk about single letter words. These are prime real estate! The most important words in the game will be:

**I**

**third person singular pronoun (he is going here, etc etc)**

**third person plural pronoun (they’re going there etc etc)**

**{game specific} names of 3 - 5 players (single consonant)**

**{game specific} 2 - 3 bombsites (single consonant)**

**common verbs for commands:**

1. **go (to a bombsite)**
2. **follow (a player)**
3. **peek at the same time with me/breach (breach this room!) {offense}**
4. **watch (a corner) [aim at it waiting to shoot] {defense}**
5. **(set up a crossfire?) hmmm**
   1. **maybe, tell one person to watch one corner/area (area more likely), and ask other player to set up crossfire with it (set trap)**
   2. **probably going to be one verb**
   3. **{defense}**

**Common verbs for statements:**

1. **is (both for is located at and is status; i guess I don’t need to make different ones for either because the properties will be clear (and if i did, there would be an infinite number of is’s i would have to make)**
2. **die (x: i’m dead/i died)**

**common particles:**

1. **? for yes/no questions**
   1. **i’m thinking having “o” be this, or some vowel that wouldn’t be used much…**
   2. **for others, the question word would suffice for marking the sentence as a question.**
2. **yes/no (it is disputed over what part of speech these are, so i’m just going to call them particles**

**demonstrative adjectives (the four) (to designate different objects quickly)**

Should I have multiple 3rdPS pronouns so that I can distinguish between different people?

how would that work?

assume that all 3 enemies are identical (not going to use gender to turn it into he, she, zhe)

then

I’ll figure this out later. Now let me spin off of this and execute each of the tasks brought up by these objectives.

**Personal Pronouns**

Personal pronouns are the only (?) places in the language where there will be pseudo plural words.

English version of the words:

|  |  |  |
| --- | --- | --- |
|  | **singular** | **plural** |
| **1st person** | I | we (inclusive), we (exclusive) |
| **2nd person** | you (singular) | you (plural) |
| **3rd person** | he/she/it (gender/animacy neutral) | they |
| **4th person (hypothetical)** | one | ones?? (maybe, maybe not) I’m not sure if it’s going to be useful |

Kairen words:

This is a big entry choice btw.

1st person singular, 3rd person singular have to be 1 char (for cslike).

Because these words can come after a verb, they should be different from verb conjugations to avoid ambiguity. The possible vowel endings for conjugations are a, u, o. So the vowels that can be used by themselves are i, e. those should be the 1st and 2nd persons respectively.

|  |  |  |
| --- | --- | --- |
|  | **singular** | **plural** |
| **1st person** | i | is (inclusive), ir (exclusive) |
| **2nd person** | yu | yus |
| **3rd person** | e | es |
| **4th person (hypothetical)** | ho | hos |

**Demonstrative Adjectives**

The demonstrative adjective system works like this;

Demo1 (these are all placeholders by the way) is for an object closer to speaker

demo2 is for an object closer to audience

demo3 is for object close to both

demo4 is for object far from both

Here are my entry choice words:

demo1: ti (closer to **i**)

demo2: tu (closer to **yu**)

demo3: te

demo4: to

I’m just trying to help the learner by making words similar in meaning similar in pronunciation. Then I can use the remaining one “ta” to be the number 2 to be easy to remember. Which reminds me, I should do the numerals.

**Numbers**

As previously discussed (really in every language I have tried to make (or more realistically, ever version of Kairen I have tried to make)) I will use the CJK method of numbers:

2 10 = 20

etc.

This means that 1 - 9, 10, 100, 1000, etc have to be as short as possible in word text. Same goes for the radix digit (I’ll use , because the computer might get confused if it sees . and thinks it is the end of the sentence).

I’m going to work on making a fractional system for the numbers as well, I’ll have to take lessons from japanese on how they do this

As of right now, I think I already have a number system, because I remember encoding my social security number in it. But I don’t know where I wrote it (or i could look for it on google drive but it’s probably blocked) and I think things are different now anyways. So here is the current idea for what these numbers are

1 nu

2 ta

3 ri

4 fo

5 va

6 si

7 ze

8 me

9 na

10 den

100 hud

1,000 qoz

10,000 denqoz

100,000 hudqoz

there should be a structure for these higher ones to go on forever. This means, there should be versions of the numbers that can be used as prefixes. This needs to be distinct from the numbers themselves in order to create compound words that are different from just adding a numeral to a thing:

x: biannual = every two years

let yia = year

let ni = generic noun (plus others?) -> adjective particle that translates here as: “something that happens every \_\_\_”

tayiani = 2year =?= biannual (this would be the most straightforward way of constructing that word)

tayiani sounds like ta yiani which would mean 2 yearly, which is different

I wonder if this would be a problem? It would definitely be a problem if there wasn’t “ni”. There is “ni” though, so I’ll have to think more about it. Are there any situations where you would want to say a compound word but have it be mistaken for that number of the rest of the word?

x: maybe you are abbreviating a longer phrase into one word (“yearly performance review” -> “yearly(s)”) This would mean that if you talk about “2 yearlies” you would sound like you’re saying “biannuals” which is a problem. This is reason to come up with an inflection for the numbers that turns them into prefixes.

add -r. a is buffer vowel

1 nur

2 tar

3 rir

4 for

5 var

6 sir

7 zer

8 mer

9 nar

10 denar

11 dennur (from dennu)

12 dentar (from denta)

x: let sis = system

duodecimal system (12 system) dentar sis (7 -> 3 syllables :))))) )

I’m thinking that if I use “r” as a buffer consonant, a lot of these prefixes or other places with a buffer consonant at the end will look like spanish verbs when they are just prefixes. eh whatever for now

I’m using resonating consonants as buffers becuase they usually have the highest number of consonants that they can be with without causing awkward combos of letters

Now, I’m going to use the complement of r (l) to make the ordinal numbers (or are they called cardinal numbers, I’m not sure exactly). Still using “a” as buffer vowel when needed

1 nul

2 tal

3 ril

4 fol

5 val

6 sil

7 zel

8 mel

9 nal

10 denal

11 dennul

12 dental

13 denril

100 hudal

1000 qozal

Now I should use the number prefixes to construct the higher numbers:

10 den

100 hud

1,000 qoz

10,000 denqoz

100,000 hudqoz

1,000,000 mil (the base for the series)

10,000,000 denmil

100,000,000 hudmil

1,000,000,000 tarmil (2illion essetially)

10,000,000,000 dentarmil

100,000,000,000 hudtarmil

1,000,000,000,000 rilmil

10,000,000,000,000 denrilmil

100,000,000,000,000 hudrilmil

1,000,000,000,000,000 folmil

10e18 valmil

10e21 silmil

10e24 zelmil

10e27 melmil

10e30 nalmil

10e33 denalmil

10e36 dennulmil

That’s actually pretty impressive because it just can go on forever. I feel really good about making the system for that. Not every conlang has an infinite supply of number names and has an easy way of remembering them!

Now lets work on the other side of the decimal point

the decimal point (radix) will be called pot.

so you could say numbers like this like this

123,489.3675 = nu hud ta den ri qoz fo hud me den na pot ri si ze va

how would you say fractions I wonder…

maybe just treat fraction line as a word x: ov

2/3 (instead of two third) ta ov ri

**Letter Names!**

I have to have a name for each of the letter that is distinct enough as to prevent ambiguity when whispered (so voiced and unvoiced variants have to be different) and over a staticy phone line. I want to minimize the need for a phonetic alphabet as much as possible.

I think they should all be one syllable, and have the sound of the letter in them.

To prevent overcrowding of “nice” syllables, I’m actually going to do VC for all the consonants; i’ll have to come up with something for the vowels that prevents them from being confused from actual words/particles.

Stop Consonants

|  |  |  |
| --- | --- | --- |
| op | ok | ot |
| eb | eg | ed |

I also need to have it be easy to memorize.

easy to memorize \\ easy to distinguish

Fricative Consonants

|  |  |  |  |
| --- | --- | --- | --- |
| of | oc | os | oh |
| ev | ej | ez | eq (unvoiced, but because it has no voiced counterpart, it is the best candidate for the “voiced counterpart” to h) |

Resonating Consonants

All of these are going to be voiced, but we will try to pair them logically and it’s really random which ones are going to be in which of the two slots

|  |  |  |  |
| --- | --- | --- | --- |
| on | or | oy |  |
| em | el | ew | ex |

x is closer to n than m in sound to most people’s ears (i think actually too) so I think it’s better as an “e” to distinguish it.

Vowels:

vowels can’t be just themselves (even though that would be easier because I’m using them for other things because they are prime real estate). maybe combine them with a word that means vowel (or at least is the first syllable of it (or even better, first letter!)) to make the names:

let jou = vowel

a: aj

e: ec (change to unvoiced to make it distinct from letter j)

i: ij

o: oj

u: uj

The special case with e is going to make one bit of ambiguity with ec vs ej (e vs j), but I’m really ok with that because there will have to be at least one point of crossing that will cause this issue.

Now for letter order. To make this easier for people to memorize, I’m going to group them by sound closeness. Within each group, I’m going to use english alphabetical order for similarity (this applies to unvoiced only).

k (ok)

g (eg)

p (op)

b (eb)

t (ot)

d (ed)

c (oc)

j (ej)

f (of)

v (ev)

h (oh)

q (eq)

s (os)

z (ez)

n (on)

m (em)

r (or)

l (el)

y (oy)

w (ew)

x (ex) (it’s partnerless, so it makes more sense to put it at the end)

a (aj)

e (ej)

i (ij)

o (oj)

u (uj)

**Yes, No, Maybe Words**

These words can be used to answer yes no questions. “I don’t know” can be used to answer any question.

I feel like there are a number of ways to answer a question based on case, so I’m going to flow chart it

Yes/No question example:

Is he in the tunnel?

Cases for speaker: Flow chart:

Actually I thought about this in the shower and you don’t really need two different yes/no/maybe for factual and intuitive evidence. There are other sources of evidence like dream and third party (who have their own evidence) and it doesn’t make sense to make a grammatical construction.

Instead, I should have yes/no be the end goal and have a number of evidentiality particles like láadan that just are appended after a verb (or at the end of a sentence/ or a statement (i got to figure out which is best)) to show where the evidence for it is.

This means that the yes/no/maybe will have only one copy.

What I need to figure out is how to simply this casework. I have a feeling that I’m approaching this problem wrong. Let me consider the ai case first, as that is more simple.

ai case:

Knowing is boolean. There is no forgetting, no foreseeing (at a base level). Just simple storing and recalling of knowledge.

**Factual** (no complex thinking, just recalling data verbatim)

the ai either knows something and can recall it completely or doesn’t know it at all.

x:

ai knowledge [john is in the house]

q: is John in the house?

ai: yes

[john is in the office (office and house are known to be locationa and only one location is allowed at a time)/john is not in the house]

q: is john in the house

ai: no

[]

q: is john in the house

ai: idk

yes/no/idk are the only possible responses at level 1 (factual)

**Intuitive**

ai has some past history of locations at times, ai has prediction algorithm, ai has some brain part that allows it to judge the probability of john being somewhere

[anything]

q: is john in the house

ai: (yes/no/idk *factually* (that’s gonna be a particle)) 0-100% probability *intuitively*

*factual and intuitive thinking are the only two thinking paradigms that I know of that an ai can use that don’t involve randomness*

How is the previous example different from how a human thinks?

A human has more than factually and intuitively to know something

A human doesn’t always know how well he or she knows something (factually, intuitively, dreamly, etc)