Glossary of terms:

Absolute root

Informatihedron

Neighborhood

Granularity level

Core identity trait

Spectrum

Collapsable states

Locked

Layer

Primacy

suppositions (assumptions made by the description text)

* processing
* Database (graph)
* Abstraction system
* Coordinate System (places abstracted info)
* Graph space
* GUI
* > Translator (Real estate agent) (queries the informatihedron in the database and returns the ontological data for user to view)
  + HUD
* Agent
  + User
  + AI

Crystal Ball is an intelligence amplification application for ontology acquisition

Platform for slowly building knowledge base of human understanding

Prototype

MVP Rules:

Only seriously populate the necessary ontologies

-> ie for In n Out prototype: Ingredients but not business operations

The magnitude of the abstraction numbers should be dictated by the number of layers removed from the origin (absolute root)

such that from the relative origin, the orientation of the view in the GUI displays whatever the next level of granularity is within that absolute root, but since the display is constrained to N layers, only a certain amount of granularity (class and type) levels (search scope) can be included in any given search result (instantiation of an informatihedron).

In each granularity level, the class of each item is its degree of granular relationship to the relative origin (level 1:1, 1:2, 1:3 degree of relation + depth per visible layer, 2:2, 3:3 and so on?). These are visualized as the layers of spheres in the overall sphere. The overall sphere is constrained to N layers based on processing projections pre-initiation (physical limitations).

A neighborhood is defined as a space of spheres of spheres of ontological information which may or may not be processable altogether in its raw form, so we can intuit that such and such an informatihedron must be in such and such type of neighborhood. Any informatihedron is a sphere of spheres, but what is meant by “informatihedron neighborhood” is the layers of a “highly specific informatihedron with many locked CITs whose first layer is strictly the primacy of the amounts of the CIT items within the boundaries of the CIT spectrum: all collapsable states of the locked CITs.”\* The 2nd layer is one unlocked CIT being moved by some amount which was dictated on the first layer. *\*= This means that what is more closely related is closer together.*

The third layer is then a different “highly specific informatihedron with many locked CITs whose first layer is strictly the primacy of the amounts of CIT items within the boundaries of the CIT spectrum: all collapsable states of the locked CITS”, where the difference between it and the original “informatihedron with many locked CITs whose first layer is strictly the primacy of the amounts of the CIT items…” is TWO unlocked traits, but one of those traits has been locked again by the 2nd-most-original informatihedron… and so on.

IE: with “wesley’s product” the first layer is the different acceptable configurations of amounts of each ingredient. Let’s say the primacy order we like is “mostly lettuce”.

The 2nd layer is the different acceptable configurations of some “mostly lettuce” version of “wesley's product” where we are also now looking at the spectrum of acceptable configurations of amounts of every ingredient that isn’t lettuce.

Because if we unlock one of the ingredients in wesley’s product, instead of keeping them locked, we are no longer inside the informatihedron neighborhood of what we were calling “wesley’s product”, even though this is technically the same search, we’re in a different thing called “wesley’s 2nd product that he likes more than where the first product CITs took him”.

* Explain venn diagram search\*\*\*\*\*   
  (previous search information (current search info) new information)
  + Apply to neighborhoods and NN with examples

\*\*\*\*\*

(wesleys product 1 (spectrum of primacy values for amounts of ingredients) wesley’s product 1 with set ingredient primacy but now we are changing one ingredient amount)

\*\*\*\*\*

In a neighborhood view, each sphere in the layer is a possible arrangement of values for the CITs making up the origin point, whereas in an informatihedron (search parametric assembly/constraining view) each sphere in the layer is an ontological entity connected to whatever the ontological CITs of the root are. ///

When you go one layer out in the neighborhood view, you are choosing the primacy of the ordering of the values from largest value to least. The next layer is dictated by a first layer path chosen for a walk, and is simply the process of specifying the amounts in the primacy order of CITs. If you want to change the original CITs, you are already in the “wrong neighborhood” to do so, and need to go to a different one, which means you need to go back to informatihedron view, not neighborhood view.

First, Wesley must find his ingredients from the list of in n out ingredients and then he must build his recipe. In order to build his recipe he must visualize the spectrum of amounts of ingredients according to their primacy in a set. Some of these sets are obviously garbage and can be discarded, some of them are closer to his search parameters of salad. For instance, let’s say there is product research that not many customers would like an onion salad and tomato or lettuce salad are preferred. What differentiates them are the amounts of the ingredients and what differentiates the amounts of the amounts is the primacy of the ingredient in the recipe (highest %). Crystal Ball allows you to arbitrarily use expert levels of “understanding of ingredients” to make a recipe with those ingredients even when you might be a complete novice by turning your brain into the neural net for the search for the instruction set to the solution closest in actual reality to what you’re intuiting as a solution.

* Compare bloom with Crystal Ball
  + What is traditional Data Mining vs our version of Data Mining
    - Differences in neighborhood population, informatihedron acquisition etc.
    - What is our version of Data Refinement

////

User inputs search parameters

* Search parameters that equal absolute root
* Search parameters that have narrow boundaries
* Search parameters that have wide boundaries

Each search parameter is a starting position in the coordinates of information, the ontological database of domains and their attributes which is searched

Those ontological specifics then are parts of their own neighborhoods, which creates the effect that whatever informatihedrons they themselves are part of must include the entire neighborhood of informatihedrons related to it inside of it and that seems redundant to the point of explosion

Question^^^^