Vitruvian Brain

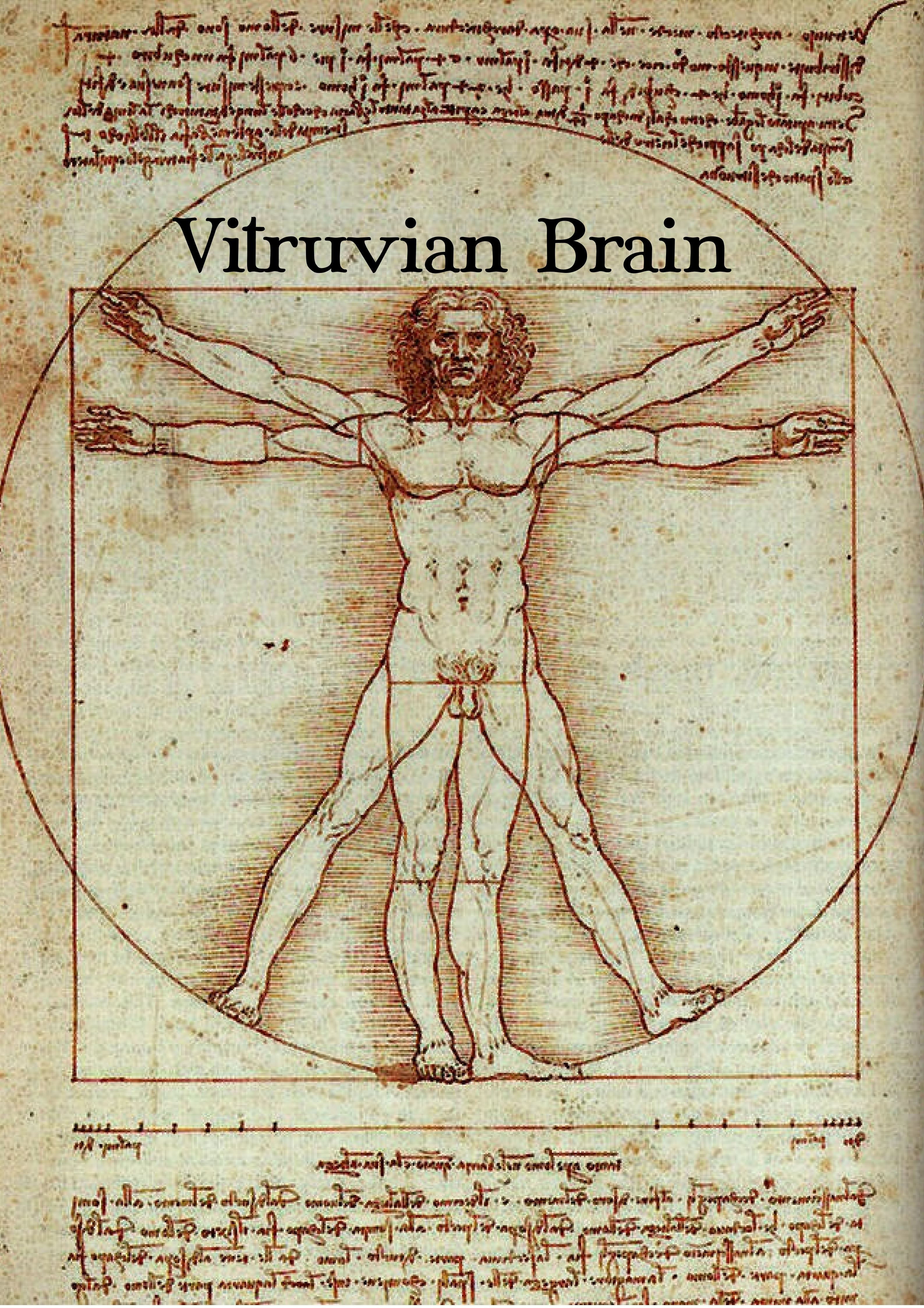

Table of content

One of the many problems with self-learning today is the lack of full-featured tools four your studies, current tools, even when they provide all the necessary features to a productive & plesurable learning session they might have other flaws such as lock-in, complexity, online-only, platform-specific etc.

The purpose of this document is to present the current state-of-art in study/learning/thinking tools, present their respective advantages and disadvantages and the solution currently being worked on, as to address all the flaws where others have "failed", picking the best parts, leaving the bad and trying building a standard that is Simple, Open Source, Plain Text, Robust and Extensible.

To summarize, this is an attempt to create a lock-in free, offline, platform-independent "standard" in which universal learning tools can be build upon.

Goals

The `Vitruvian Brain` standard has the goal of describing a system in which the following learning tools will be make available:

Zettelkasten, Incremental Reading, Spaced Repetition, Interest Repetition, Incremental Video, Mind Maps, Learning From Text(LFT), Incremental Thinking, Incremental Work, Getting Things Done(GTD), Interleaving and Incremental Writing.

It'll do so by describing a series of procedures, formats and protocols in which *any* developer[s] might choose to "implement" or "be compatible with" . This is to make sure the standard is long-lived and does not succumb to the fate of "bit-rot" in which programs tend to have a rather short shelf life. Protocols are much more resilient because they're implementation-independent, so anyone can write an implementation for it, one analogy might be that web browsers come and go all the time, but the HTTP protocol has shown itself to outlive most of its concrete implementations.

Why not Anki?

Anki is a flashcard spaced repetition application where you can review cards on a exponential schedule, only needing to keep the daily review.

Pros

- - Popular: A lot of people in medical school and the language learning community have been using anki for ages, so familiarity with them is a plus.
- - Community: A lot of pre-made decks and plugins

Cons

- - Light Lock In: Anki "locks" your files, videos, images, audios and flashcards in a non plain-text format thus "sucking" everything you throw at it and you have to do some work in get it back.
- - Complexity: maybe there's too many, unecessary options which might scare beginners.
- - Bad Integration: in order to intereact with something like a zettelkasten system you would need to either connect with AnkiWeb(API) or open the app every time(too wlo).

Anki Conclusion

Implementing anki into the system I've presented you with would be possible, but we would suffer from integration inefficiencies and the "what it could have been" effect.

Why not Supermemo?

Supermemo is a software program created by Pitr Wozniack. Its most known features are *Incremental Reading* and *Spaced Repetition Flashcards*.

Pros

- - Completeness: It has Spaced Repetition(SM17), Incremental Reading, Sleep Chart, Knowledge Tree, Neural Review and more.
- - Efficiency: It has probabily the most efficient interval calculating algorithm(SM17) and Incremental Reading is not implement to the level it has in supermemo, it has clozes, image clozes, topic schedule etc.
- - Community: It's considerably big, you can find plugins, workflows, tips, keyboard shortcuts etc.

Cons

- - Complexity: It looks like the central panel for a nuclear reactor station. It takes too much time to get used to the buttons, shortcuts, conventions, nomeclatures etc.
- - Platform Dependend: Windows Only(Wine is not always reliable)
- - Zettelkasten: I have heard you can implement it on SuperMemo, In my opinion it also suffer from a "what it could have been" effect in my view.
- - Lock In: It locks your files into a non-plaintext format.

Supermemo Conclusion

In terms of raw cognitive power and stufy efficiency, SuperMemo is not only probably the best tool around, but it quite literally pioneers a lot of the ideas in which we take as best practices today(Spaced Repetition, Incremental Reading, Incremental Learning in general). But I feel like we could take the best parts of supermemo, turn it into a concrete-implementation-agnostic and let people make their own tools according to it, it might leave to considerable growth in the field of study tools

Why not Obsidian?

Obsidian is a note-taking application where it works with plain-text markdown files, it's very commonly used to implement a zettelkasten system.

Pros

- - Almost Platform Independent: It's available for Windows, MaxOS, Linux and mobile.
- - No lock in: It's plain text files a folders
- - Popular: A lot of people in medical school and the language learning community have been using anki for ages, so familiarity with them is a plus.
- - Community: A lot of pre-made decks and plugins
- - Extensible: With The Obsidian Plugin API you can write features the tool does have but you want to use.

Cons

- - Files is the lowest level you could go, and block references don't work that well.
- - Concrete implementation that might not have everything users need and the possiblity of a plugin might be be as promising.

Obsidian Conclusion

Obsidian comes the closest to an "open" study tool. It does not lock your files in, have zettelkasten, incremental reading and spaced repetition plugins. My only problem with obsidian is the fact that its main "primitive" or "unit of work" is a markdown file, which is fine for most people but I'm pretty sure some people would desire a more "lightweight" way of creating relationship. One example would be a interesting quote, if you want to make this quote a node in the graph/zettelkasten you need to create an entire file for it, which is in my opinion too cumbersome/uncecessary, if you collect 100 quotes you'll have 100 more files ... of course you can all put them in a single file, but referencing them from this single file would go "against" the atomicity principle of zettelkasten and each individual one would not be visible in any graph.

Why not Roam Research?

Roam Research is a paid online note-taking application where people can access their *graphs* or note online.

Pros

- - Simple: Very slick and intuitive.
- - Almost Platform Independent: Works on the brower, it means it's almost universal.
- - Blocks: small, cheap and removes the "does this deserver a file?" effect.

Cons

- - Online Only:
- - Lock In: Roam has the option to export - [ ]
- - Paid: Roam is a paid tools which makes is non-accessible for a lot people.

Roam Research Conclusion

Roam is probabily the best optiosn between all of the presented here but ... It's not free. So that's a no.

Why [not] ZIR Standard?

`Vitruvian Brain` Standard is a series of documentation and practices in which developers might implement learning applications. By following those design choices it'll ease the cognitive load of inventing everything from scrath, they can have a battle-tested framework for study tools and focus on the delivery of said tools, instead of discussing which solutions are the best one.

Pros

- - No lock in: It's just files and folder.
- - Platform Independent: Since it's a standard and not a concrete implementation we might implement it in: C++, python, bash, Web, iOS, Windows, Linux, MacOS, Embedded etc.
- - Good Primitive: Bullets, files and folders, everythign else is build atop those basic primitives.
- - Bullets: small, cheap and remove the "does this deserver a file?" effect.
- - No "Software Rot": Since it's not even a concrete implementation, there's nothing to rot here, protocols outlive programs by the decades, that's the intent here.

Cons

- To be added ...

ZIR Standard Conclusion

The very purpose of the `Vitruvian Brain` standard is to learn from those previous tools, take their best points, mitigate their weaknesses and make an open standard from a hypothetical study tool with the features that provide the most benefit(Spaced Repetition, Incremental Reading, Incremental Writing, Zettelkasten, Interest Repetition, Interleaving etc.)

Principles

The `Vitruvian Brain` will takes inspiration from the `Unix Philosophy` such as:

`Do one thing and do it well`

`Plain text is the ultimate interface`: We don't make use of `JSON`, things such as configuration are also non-json plaintext.

Fundamentals

Bullets

Bullets are the "Working Blocks" of a ZIR system, bullet are simple lines in the following format:

:

Examples

20270216193456: [[Typography]] ==Typefaces== with a **taller** x-height are considered modern and easier to read.

20200522162753: #todo/review A [[Liver]] cell, although it carries the genes to do so, will generally not be able to function as a skin cell [[DNA]] [[Biology]] [[Totipotency]]

Explanation

20200522162753: ID

#todo/review: Tags

A [[Liver]] cell, although it carries the genes to do so, will generally not be able to function as a skin cell: Content(With Tags)

[[DNA]] [[Biology]] [[Totipotency]]: Tags

Bullet ID

The bullet id is the current timestamp including seconds when the bullet was made, this works as a timestamp and also as an local unique ID, considering you can't possibly create 2 bullets in the same second. All of the bullets are considered to be unique, if one is malformed, like, without an ID, it'll not try to correct it.

Bullet ID Format

- `YYYMMDDhhmmss`
    - `YYYY`: Year
    - `MM`: Month[s]
    - `DD`: Day[s]
    - `hh`: Hour[s]
    - `mm`: Minute[s]
    - `ss`: Second[s]

  All single-digit values will have a leading 0 so: 1 -> 01, 2 -> 02, 9 -> 09 etc.

Bullet Content

You might use plain-text for the content of the bullet or the standard markdown flavor; markdown links can be either \[Technology\](https://example.org) or [[Technology]]. Any embedding, or dynamic content will be implemented on software, this gives enough flexibility for your writing without any specialized program, but remember that IDs are still needed, and when you open an actual software implementation, you'll see the images, videos, sounds etc.

Bullet Location

Bullets can be in any file such as: `Javascript.md` `Journal.md` or `Medicine Summary.md`. But it is assumed you have a "main" file in which most of your bullets will resize in(the default target location). By default this file is simply called `Vitruvian Brain`.

You can modify the default location for the bullets if you want.

Files

Files should be automatically created when we find a new link such as: `[[HTML]]` if the file named `HTML.md` has not been already created, then it should be.

Tags

Tags are defined by the following format: # or #tag/

Incremental Reading

Spaced Repetition

- [ ] The algorithm for spaced repetition will be.

The main tag used for spaced repetition is: `#sr`, and all of its children will be used for configuring its previous reviews, next schedule, status etc.

The algorithm for spaced repetition will be configured in tags like this:

20200522162753: #sr #sr/review/dqwdqwddqw #sr/ireview/22020202 #sr/treview/32121312

Interest Repetition

Interest Repetition will take into account the current "mood" of the users while showing and scheduling new cards. When "raw", it's the same as an *Spaced Repetition* algorithm where the interval between the days in exponential

Spaced Repetition & Interest Repetition scheduling algorithm

Depending if the item is being reviewed because of motivation or because of time is important to the algorithm. If it's a time-only review, then a normal exponential scheduling will work just as fine. But when we review an item *Before* its schedule date due to momentum, the scheduling date will not be the same.

It's also important not to "burden" users with accumulating reviews, which might create displeasure in retaining information and that should not be the case, we'll allow for a more forgiving algorithm than Anki's modified SM2 Algorithm.

Which means users might skip a day without being heavily punished, we're here to maximize the pleasure of learning and retention, not punish users for missing a review day.

Example:

Interest Repetition Queue

The *Interest Repetition Queue* is basically a list of files, that is, items, in which they have an "interest point" assigned to them. This "interest point" will define the item's probbility, as well as its average frequence. This is due to the fact that this list of items(the queue) is not static, it's dynamic and it's constantly changing, every time you read an item, the queue will "shift" and item, or remove its first items, and add it to the end of the list. This way, in order to see an item twice(without any modiciation) you would need to read every item on the queue in order to see the other item again.

The *Interest Repetition Queue* is a list of your interest that will act in all 3 areas: Consumption, Retention and Application

For the "Consumption" area, this will be for items you want to read, watch or consume in general. This is where most of the learning will take place, since this is where you're consumption new content instead of remembering old content or re-thinking concepts you already know(meditation).

Some examples of items that are in the "consumption" area are:

- News about an industry you would like to know
- A book on neuroscience

- A blog about how machine learning backpropagation works.
- A infographic you want to integrate into your life
- A 10 seconds video you saw online about how a guys lubricates his joint and you want to incorporate this knowledge, behavior and habit to yourself
- A 7 seconds video where you clipped from an artist performing a keyboard-shortcut on a software program you also happen to use, in order for you to start using the shortcut yourself.
- For the "Retention" area, this is the items you want to have a memory for life, this might be
- Some examples for the "Retention" area are:

  A 7 seconds video where you clipped from an artist performing a keyboard-shortcut on a software program you also happen to use, in order for you to start using the shortcut yourself.

Application - [ ]

Examples: - [ ]

Interest Repetition Queue Scheduling Algorithm

Different from the *Spaced Repetition* scheduling algorithm, the *Interest Repetition* scheduling algorithm is not about hard time, or "dates". It's more about rations and probabilities.

While a *Spaced Repetition algorithm* might schedule your items like: - day(1) -> day(3) -> day(10) -> day(30) -> day(80)

A interest repetition has no concept of "days" or "dates", it has the notion of "rations" and "probability". Since every item in the *Interest Repetition Queue* has a "interest point" which ranges from -99 to 99. -99 means that is has a 99% probability of being "ignored" while reading your queue. while 99 means it has a 99% change of going up one step every time you read your queue.

This means that the queue is "self-regulating". Items will move up and down, being popped and shifted based on their interest point.

- [ ]

Plugins

In order to prevent software rot, it's recommended to develop plugins with the following technologies: HTML, CSS and Javascript.

This way, it's a simple .html file that can be opened in the browser, since browsers generally don't have access the the Operational System's

Filesystem, you'll have to copy your bullets from your main file and then paste in on the plugin. This way your data is protected from being deleted.

It's recommended to add a keyboard shortcut to copy all of the bullets to the clipboard to transform it more easily for plugins.

Main Files

- zettelkasten
    - - The file where most of your bullets will go.
- ir
    - It's optional, but avoids "bloating" your `Vitruvian Brain` file, where this one will store the bullets for your *Interest Repetition*.
- queue
    - It's the consumption queue for your interests, it might include things such as: Videos, Audios, images, PDFs, scripts for automation, scripts for plugins etc.
- statistics
    - It's the file with read/write/delete/qpop/qshift/qscan including the time you spend, so you can know what you read, what extract you took, what you deleted, how much time you spend reading, how much time you spend working, how much time you spend on a certain script or plugin , did you spend today more on the topic of *Biology* or *Clinical Pathology* etc.

Main Folders

- irq
    - It's the folder for the *actual* *Interest Repetition* files, such as videos, images, PDFs, HTML, Scripts etc.
- .trash
    - Trash folder

Folder Structure Example

- zir/
    - zettelkasten
    - ir
    - queue
    - statistics
    - irq/
        - 890123.video.mp4
        - 123124.pathology.pdf

        - 380921.pathology.pdf
        - 453809.mtDNA.pdf
        - 32879.human-anatomy.pdf

Exponential Interval Scheduling

Importing PDFs

Avoid having more than 1 final dot in a PDF file like: "book from John H. R. last edition" instead have: "book from john H R last edition.pdf" this will prevent bugs.

Requirements

Plain Text Files

Folders

PDF reader(with anotation is better)

Screenshot tool

Keyboard shortcut manager(options, but makes the process much faster)

The Tools

Zettelkasten

Incremental Reading

Interest Repetition

Spaced Repetition

Image Cloze Deletions

You'll probability need a PDF for this, the base format is an image file like .png .jpeg etc.

So if you want cloze deletions we recommend having an automatic PDF-to-image converter to have cloze deletions.

Incremental Reading

By having PDF files and a scheduling algorithm you can have incremental reading.

Principles

Stateless

Plain Files Only

Everything is a bullet

Todos & Priorities

202201291236 25: - [ ] #ilse-standard Instead of having a personal notation for priority, instead use the normal - and [ ] just adding a number in front of it(the priority) as in -> [ ] 3 [ ] 1 [ ] 4

Topics Three

Use tags like

- topics
  - #topics/languages
    - #topics/languages/german
    - #topics/languages/spanish
    - #topics/languages/french
    - #topics/languages/icelandic
    - #topics/languages/japanese
    - #topics/languages/korean
    - #topics/languages/chinese
  - - #topics/cs
    - #topics/cs/OOP
    - #topics/cs/FP
    - #topics/cs/alalgorithms
    - #topics/cs/data-structures
    - #topics/cs/version-control
    - #topics/cs/compilers
    - #topics/cs/interpreters
      - #topics/cs/interpreters/context-free-grammars
      - #topics/cs/interpreters/llvm
    - #topics/cs/machine-learning
      - #topics/cs/machine-learning/gradient-descent
  - #topics/biology
    - topics/biology/ecology
    - #topics/biology/biochemistry

    - #topics/biology/genetics
    - #topics/biology/zoology

Give the exponential nature of the schedule, you might put any file there and it'll open either when you want, or when the scheduler makes time for it. This might be used for painting, music, editing, incremental work, incremental documentation and more.

Incremental Learning

Stream Item Best Practices

Show an example of cçassofocatopm amd how more items might be better(aomtic)

20220122104532: - [ ] #ilse-standard It should be software agnostic and take the principles from the "Unix Philosophy", stuff like plain text, accesible to other tools

20220122104715: - [ ] #ilse-standard Since we're using a plain-text, folder approach, anyone should be able to write a different client for a different platform/medium with not much troublem such as, but not limited to: Tmux, Rofi, zenity, QT etc

20220128152712: [[Ilse]] #! #ilse-standard Incremental Reading notation ideas: - <2>, @2, =2, +2, ir/3, incremental reading priority 3, winner =2, this will show the "priority" from 1-5, 1 being urgent, while 5 is like "maybe some day"

20220128162512: - [ ] #ilse-standard #!, #!!, #!!! for blocking and ir/1, ir/2, ir/3, ir/4, ir/5 for incremental reading priority levels, and #!ir/3 for "pausing" a cloze from showing, also {{c1::}} will be for cloze deletions.

20220130141057: - [ ] #ilse-standard Extending your data is preffered to extending your program

20220204114840: - [ ] #ilse-standard Config variables -> folder for incremental reading, file for bullets, main folder etc

20220204125829: - [ ] #ilse-standard incremental reading priorities will use tags, or nested tag like ir/0 ir/1 ir/2 ir/3

20220205110038: - [ ] #ilse-standard stateless, it's just the data and then you process this data into something in real-time, this is to avoid complexity.

20220205123000: - [ ] #ilse-standard incremental-reading is done with 3 tools: PDFs, annotations and images.

20220205133745: - [ ] #ilse-standard In [[Obsidian]] when people make block refs, it means that they need a more granular approach to knowledge, and the software is forcing people into making bad choices, like making a quote into an entire file.

20220205152157: - [ ] #ilse-standard, our spaced repetition should work inline too, it should work SOLELY on tags like #sr/20220204/1,2,3,4,,5,5/ or whatever, filtering should be done via text and not something else.

20220205153617: - [ ] #ilse-standard to make something a flashcard add #sr, to add a repetition do: #sr/at/12341235 1-5, to pause #sr/paused

20220205163457: - [ ] #ilse-standard every config will be given with additional tags like #ir/paused

20220206124623: - [ ] #ilse-standard stateless, things should be calculated in real-time instead of storing thing for caching purpose or later reference.

20220206202733: - [ ] #ilse-standard the goals is that you should be able to have a ZIS system(Zettelkasten, Incremental Reading and Spaced Repetition) system with plain text files, folders, a screenshot tool and a PDF viewer

20220208213854: - [ ] #ilse-standard We might have a single stream that represents 3 differents things like: incremental reading, flashcards, todo, since it's only one stream, you don't have to worry about "missing" something. just keep grinding and you'll come to you

20220210163012: - [ ] #ilse-standard We might have an image where we slowly but surely divide into smaller units until each item representes one memory/action/concept

20220211142603: - [ ] #ilse-standard our stream thing will have the following goals: make an open standard for Zettelkasten, it has to be software independent -> standard, Ilse Langnar's Notebook -> implementation. Scripts -> implementation. Stream Framework -> Something that goes along very well? Also an open standard for Incremental Reading, Spaced Repetition and Interest Repetition(my thing) using only plain text files, folders, PDFs and screenshot

20220211152053: - [ ] #ilse-standard limitations of supermemo: windows-only(wine is bad), software-dependent, too complicated, not beginner friendly, non plain text + folders format(lock in)

20220211211141: - [ ] #ilse-standard stateless as possible, only the bullets are used, nothign else.

20220213195249: - [ ] #ilse-standard #ziis when you have script for items, it'll execute those scripts, because you might want to have something more elaborated like "open program a, do a stream query for "mp4" then open this .png file all at once"

- [ ] Add support for RSS for your first brain?