Evolution of Software

In terms of Structure & Design

Multi-Layerism

Modularity

Front-End / Midde-End / Back-End

Programs should grow like Plants

How to analyze design

- Dependency Graph
  Static & Acyclic (DAG)
  - Module dependency graph
  - Class hierarchy graph

- Call Graph
  Dynamic and with possible cycles
  AKA: Sequence Diagram

```
$ modulegraph -p /starcal scal2/ui_gtk/starcal2.py -g >
graph.dot
```

(then clean .dot file by hand)

```
$ dot -Tpng graph.dot -o mgraph.png
```

(modulegraph does not support dynamic __import__)

```
$ pycallgraph graphviz scal2/ui_gtk/starcal2.py
(The result is HUGE)
```

StarCalendar

How do I code

How do I publish my codes

How do I release a new version

My Tools

- Gedit (Pluma)
- Gnome/Mate Terminal
- git, gitg, colordiff
- pydoc
- Nautilus / Caja
- Chrome & Firefox

My Branches

- master
- next
- next-major *
- py3
- one for each challenging feature
- gh-pages

How to setup a simple homepage

with

Github pages (gh-pages)

Some Innovations of StarCalendar

- Parallel Calendar Types
- Encapsulated GUI codes and extreme modularity
- Event Occurrence Index
- Events structure and format
- Real Continuous Time Line

Events structure and format

- Designed better than iCalendar
  - Registered in 1998 by Microsoft as RFC 2445
  - Used by Google cal, Evolution, Korganizer, Apple calendar and almost every other calendar program
  - Designed very disorderly, not very much object-orinented
  - Hard to implement in both back-end and front-end (GUI)
  - Wikipedia says: "Recurring and repeating meetings still have a bit of mystery and ambiguity associated with them. Resulting in no true interoperability between the current calendaring and scheduling vendors."
  - Not compatible with some non-Gregorian calendars like Islamic and Hebrew calendars (as Wikipedia says). And applications implementing it do not support Jalali calendar either
  - For more information http://en.wikipedia.org/wiki/ICalendar

Events structure and format

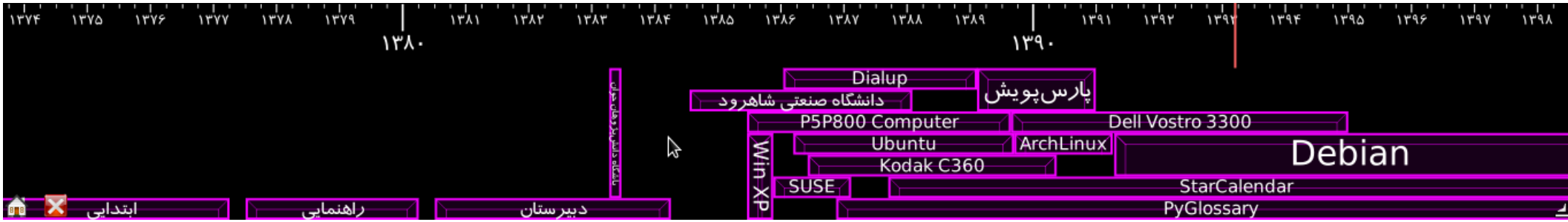
- Object Types
  - Rules (~22 types, with dependencies)
  - Notifiers (~4 types)
  - Events (~15 types, with some inheritance)
  - Event Groups (~9 types)
  - Accounts
- Use JSON file format, why?
  - Faster than XML, designed to be read/write by machine, not by human
  - Still plain-text (ASCII or UTF-8), editable by human
  - C-style explicit blocks, not weird blocks like iCalendar
  - Can be saved in **Pretty** format (with indentation), or **Compact** format
- No database

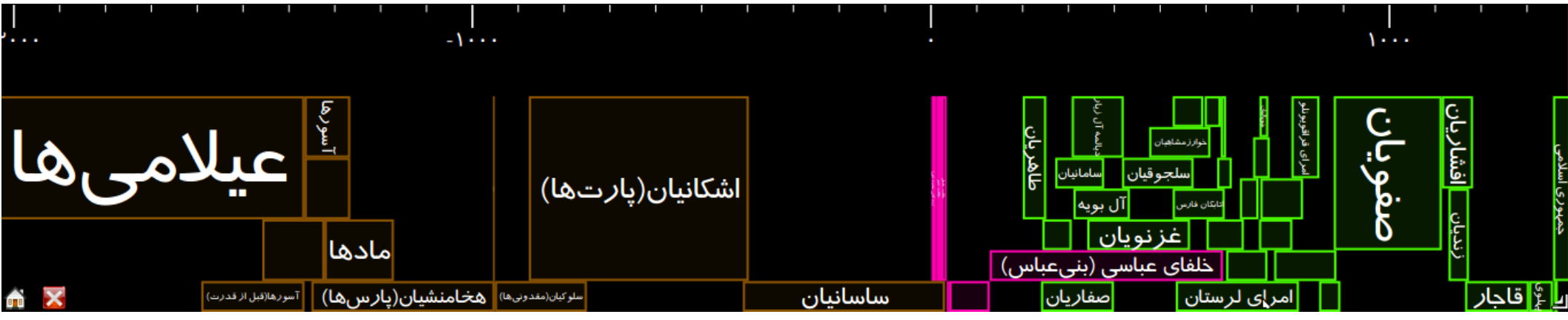
Event Occurrence Index

- **Super-Fast overlap queries**
  - I have 10,000 events in the last 10 years
  - Max 10 events per day
  - Give me all the events in this one day range, **right now**! (avg < 0.01 s)
  - Logarithmic (Sublinear) time: `O(query_time) ~ ln(evens_count)`
- Small index size (in-memory)
- Fast index generating
- Algorithm
  - EventSearchTree
    - A complex of
      - Red-Black BinarySearchTree
      - Binary Heap
      - Dictionary (HashTable)
    - 400 LOC
  - TimeLineTree
    - Customized (non-binary) Search Tree
    - 200 LOC

Real Continuous Time Line

- Dynamic scaling (from seconds to millions of years)
- Dynamic and fast event drawing using Graph Theory algorithms
  - Construct an Interval Graph
  - Color the graph with (practically) minimum colors in a fast way (trade-off)
  - Extract intervals with the least color and draw them, again and again...





For Ubuntu users

Install *python-appindicator*

to use StarCalendar in Unity

install-ubuntu script is added

in master branch (2.3.4 not out yet)

We need package mantainers for

Ubuntu

Debian

openSUSE

Fedora

ArchLinux

…

even Windows

Please Help if you feel like it

Please report bugs via

https://github.com/ilius/starcal/issues

or send it to

saeed.gnu@gmail.com

Your help is most appreciated

Thank you :-)