GUI for enterprise level high frequency trading systems, making focus on visualizing market microstructure analytics, such Limit Order Book dynamic, latencies, execution quality, and other analytics.

Chart

Description automatically generated

**Long Description**: enterprise level trading systems run in collocated server with no human interaction. With VisualHFT, we can have a look at what’s going on with the markets, risk, exposures, and many other analytics.

The main focus is to have a dashboard showing market microstructure information. That means that we can see L2 prices, from different venues and the aggregation of it. Also, we can see Limit Order Book dynamic and its resting orders.

The initial intention of this project was to support a high-frequency trading operation (running with latencies under 2 microseconds), but this project could be used for any type of trading.

Technologies used are C# and WPF, and ideally, we will be updating to support other platforms too.

Lot of things need to be improved, so please be patience. And if you have the skills to code, and ideas, happy to include contributions.

**About me**: I’ve been building high-frequency trading software for the past 10 years. Primarily using C++, for the core system, which always runs in a collocated server next to the exchange.

If you want to learn more about what I’ve been doing, checkout my blog at <https://medium.com/@ariel-silahian> or follow me in twitter @sisSoftware

**History**: I created this visualization dashboard to allow us to visualize what was going on the main HFT system, residing in a collocated software, and with no human interaction. The goal was to have a quick peek on what was happening, how many orders were being sent and how, how the market was and control some of the strategy parameters.

**How to Install and Run the project**: the server app must send the following websocket messages: show json messages

The system needs to be fed with the defined collection of JSON data. Also, for positions and executions, the information must be read from the database (MS Sql Server database). Also, this system will send some commands back to the core system via REST (start/stop trading, params change, etc)

Core Trading System: this is the system that will feed VisualHFT. Currently is not included in this repository, however, we have plans to include a small system as an example. It will need to have the following: REST and Websocket server, and persist positions into the database.

The overall configuration is that the core trading system (again, not included in this repo) will also be a websocket server sending the data, and also persisting the position data into the database so VisualHFT can read from there.

Diagram

Description automatically generated

**Screenshots**

Graphical user interface

Description automatically generatedA screenshot of a video game

Description automatically generated

Text

Description automatically generated with medium confidenceChart

Description automatically generated with medium confidenceGraphical user interface, chart, histogram

Description automatically generatedGraphical user interface

Description automatically generated

**Why I decided to open the project and the motivations**: too much work for one person. Besides, my focus has always been on the server side, designing high-frequency trading software. Not that much for the UI. Besides, there is no open-source project that provides market microstructure visualization and analysis. That’s why the goal of this project is to be able to display specialized data around microstructures.

If we can build up a community around this product, it could help to improve the current architecture, make it much more scalable and be able to add nice new features. Ideally, adding more advanced real-time risk metrics, analytics, and TCA (trade cost analysis).

**Things to improve**:

* currently using Telerik Charts. Replace them with some good opensource WPF charts.
* documentation and wiki page.
* code architecture.
* be sure to maintain a MVVC pattern.
* be able to add more UI (ie: web)
* throttling websocket input messages (when server send lot of messages)
* performance (real-time charts are taking up too many resources).
* generalization of the strategy parameters and their UI elements.
* scalability (able to be used by multiple users at the same time)
* more real time analytics and risk measurements
* server app for testing purposes (connect to binance, and simulate orders and executions)
* add unit tests
* Ability to have a FIX network sniffer as input data (as alternative to websockets)
* Security: even though these kinds of applications run inside a private network, there is no security at all involved.

**How to contribute**: follow the guidelines to contribute

[Setting guidelines for repository contributors - GitHub Docs](https://docs.github.com/en/communities/setting-up-your-project-for-healthy-contributions/setting-guidelines-for-repository-contributors)

*Important: We* ***will not accept*** *any changes to any of the existing input json message format. This is fixed and cannot be changed. The main reason for this is that we can break all existing installations of this system. Unless there is a “very strong” case that needs to be addressed, and all the community agrees upon that.*

*However, we could accept having new json messages, to be parsed and processed accordingly, without breaking any of the existing ones.*

**How to contact me**: for project questions use repository’s forums or through any of my social media profiles.

[Twitter](https://twitter.com/sisSoftware) | [LinkedIn](https://www.linkedin.com/in/silahian/) | Forums