Developers documentation Enigma Research 0.1

*Jan Kampherbeek, April 25, 2023

Enigma Research - introduction

Enigma Suite is a software suite, written in C#. The program is free, and open source. This document provides some information for interested programmers. In future releases, I hope to augment this document.

Please read the User Manual for information about the functionality of Enigma Research.

I want to thank Gökhan Yu for convincing me to use C# It was the right choice for building a Windows based astrology application. Gökhan also provided valuable insights into the technicalities of C# and .Net.

License / open source

Enigma is Open Source. You can use it following the terms of the GNU General Public License (GPL).

The GPL allows you to use, change and redistribute this software only if your own software is open source. It does not have to be free, but the full source code should be publicly available.

For more information, see the file *gpl-3.0.txt* in the source's root.

Enigma uses libraries from the Swiss Ephemeris (SE). For the SE, additional conditions are in place. These conditions prohibit the use of the software unless it is open source and also free. If you want to charge money for a program using software from the SE, you need to buy a professional license from the SE.

For more information, see the file *se_license.htm* in the source's root.

To use software from Enigma in your program, that program has to be open source. If you include the libraries from the SE, it also has to be free. Buying a license from the SE does not change the condition from the GPL that the software should remain open source. If you want to create software that is not open source you can use the libraries from the SE but you will need to buy a professional license, and you cannot use any code from Enigma.

Technical aspects

Technical environment

- Language: C# version 10.
- UI: WPF/XAML.
- .NET environment: .NET Core 6.
- IDE: Microsoft Visual Studio 2022.
- Dependency Injection: Microsoft Extensions Dependency Injection.
- Unit testing: NUnit.
- Mocking: Moq.
- Database/persistency: JSON files.
- Logging: SeriLog.

Coding conventions

I will try to abide to the standards. For a definition check: https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions

Using the Swiss Ephemeris

For astronomical calculations, I use the Swiss Ephemeris (SE). The SE comprises a set of data and a 64-bits dll: *swedll64.dll*.

To access the dll, the attribute [DllImport] is used. All imports from the dll are defined in facades. As an example for the definitions I used the file swissdelphi.pas that Pierre Fontaine and others created to access the same dll from Delphi.

Icons

All icons in Enigma, except the main icon that appears on the screen, are from the icon set by Google, used for Material Design.

You can download the originals at https://fonts.google.com/icons

Installing the code

Clone the repository from GitHub: https://github.com/jankampherbeek/EnigmaSuite .

Copy swedll64.dll from Enigmasuite/Enigma/Enigma.Frontend.res to Enigmasuite/Enigma/Enigma.Frontend/bin/Debug/net6.0-windows and to Enigmasuite/Enigma/Enigma.Frontend/bin/Release/net6.0-windows

Projects

Enigma is build as a .NET solution that contains 7 projects. There is a separate project for unit testing, the other 6 projects contain the application. There is only limited communication between these 7 projects.

Project Frontend.Ui

Frontend.Ui is activated as the application starts. It takes care of some initializing. Its main task is showing information to the user and receiving input from the user.

Project Frontend.Helpers

This project contains helper classes for handling the Frontend.

Project API

The classes in this project receive requests from the Frontend, perform some basic validation and pass the request to a Handler in the *Core.Handlers* project. In most cases, the API returns a response to the Frontend. An API never contains any business logic.

Project Core.Handlers

A Handler orchestrates the fulfillment of a request. Possibly it uses some basic business logic but in most cases it will rely on helper classes. A handler may call other handlers. Sometimes it will simply pass through a request but it can also combine the results of several helper classes from the project *Work*.

Project Facades

The project *Facades* contains classes that can access the outside world. A range of classes is used to access the dll from the Swiss Ephemeris. Other classes take care of persistency.

Project Domain

Domain contains all domain objects, including enums, DTO's and records. *Domain* cannot access other projects and is itself accessible by all projects.

Astronomical aspects

I follow the usual approach using the Swiss Ephemeris but I need to mention some specifics.

School of Ram: hypothetical planets

Enigma supports the three hypothetical planets as proposed by the School of Ram: Persephone, Hermes and Demeter.

The calculations are based on the orbital elements and calculated separately, without accessing the SE.

School of Ram: oblique longitude

The School of Ram supports a solution for the projection of the solar system bodies to the ecliptic. This solution ensures a proper placing of bodies in a house. However, the projection to the ecliptic is skewed. The solution is called 'true place' and also 'astrological place'. I prefer the more correct term 'oblique longitude'.

Enigma implements a dedicated calculation of this oblique longitude.

Research

Control groups and random numbers

To create a control group, you need to calculate random values. The standard *PRNG* (Pseudo Random Number Generator) does not supply true random numbers.

Enigma uses *System.Security.Cryptography* from Microsoft, a *CSPRNG* (Cryptographic Secure Pseudo Random Number Generator). More information: https://download.microsoft.com/download/1/c/9/1c9813b8-089c-4fef-b2ad-ad80e79403ba/Whitepaper%20-%20The%20Windows%2010%20random%20number%20generation%20infrastructure.pdf

This solution is sufficiently random to support the creation of control groups.

I want to thank Cees Jansen for explaining the peculiarities of randomness to me.