Implementation of Graphic Plugin Loading Platform Based on Python

Yu Liu$^1$ and Youhui Su$^{2,*}$

$^1$School of Mathematics and Physics, Xuzhou Univesity of Technology, Xuzhou Jiangsu, China
$^2$School of Mathematics and Physics, Xuzhou Univesity of Technology, Xuzhou Jiangsu, China
* Corresponding author e-mail: suyh02@163.com

Abstract. With the scripting language getting more and more mature, individual developers can also develop script programs according to their own needs based on the existing software library. In order to help individual developers to quickly and simply complete the software graphical problems, a graphical plugin loading platform is developed based on Python language. Through the specified plugin class standard, developers can quickly convert existing scripts or develop scripts according to plugin class, and easily realize the software graphical.

Keywords: Python, Graphical User Interface, Plugin sub-modules, Personal Developer

1. Introduction

With the scripting language getting more and more mature, and the threshold of software development getting lower and lower, individual developers can also develop script programs according to their own needs based on the existing software library, which provides great convenience for their own work and life. The scripts developed by individual developers not only can be shared internally within the developers, but also can be shared with non-developer users, such as sharing script programs with partners who lack development capabilities, so as to improve work efficiency. However, the premise of sharing scripts with non-developer users is to make script graphical. Graphical software is the basic service provided by modern software applications for non-developers, whose main function is to lower the threshold of using the software. However, as far as individual developers are concerned, the graphical cost of software is so high that it is likely to be higher than the cost of core functions, which greatly affected the enthusiasm of individual developers to share their independently developed scripts with non-developers.

At present, there are two commonly used graphical means, both of which are considered to be not suitable for individual developers after analysis. The first means is to customize a set of graphical user interfaces (from start to finish) based on a graphical library$^{[1,2]}$. In this way, exquisite interface can be realized and the interactive design is in line with the purpose of software development, however, the development cost is too high$^{[3-5]}$. The second means is to attach to a main program and convert the scripts into its plugin$^{[6]}$, which can not only lower the graphical cost but also realize its desired
functions. However, this means is limited by the usage scenarios and open interfaces of the main program, as a result, the target user groups and the functional scope of plugins are also limited such as plugin for Chrome browser and plugin for Tampermonkey [7, 8], etc.

On the basis of the advantages and disadvantages of the above-mentioned means as well as the focus of various means, a graphical plugin loading platform based on python-eel library is designed for individual developers who have certain development capacities. Based on the python-eel library[9, 10], it takes the browser kernel as the foundation to complete the graphical process through providing a batch of pre-set standard graphical interface and a standard plugin class for developers, so as to help developers complete the graphical process and plugin of existing scripts with low development cost or directly develop a standard plugin of a platform. Through the standard graphical interactive interface, under the premise of meeting the needs of users, the system sacrifices the characteristics of the software interface and lowers the development cost, in addition, it breaks through the limitation of being a plugin for a single browser, which can make it better adapt to the use environment of users.

2. Overall System Design

2.1. System Design Architecture

This system is composed of two main parts including plugin load control class and Eel Web service. With auxiliary functions provided by the tool class, this system can realize combined interaction between the interface and the background. In the startup, the main program initializes the plugin loading the control class, then starts the Eel Web service, waits for user's request, and loads the specific functional plugin as required.

The plugin load control class can be classified into four parts according to their functions, namely, initialization system, plugin installation and uninstall system, and plugin calling and parsing module. The plugin calling and parsing module consists of two parts, namely, the system-provided file search function and the calling and parameter input of specific plugins. The system architecture of this system is shown in Figure 1.

![Figure 1. System architecture diagram](image)

2.2. System development tools and operating environment

According to the design and development requirements of the system, python is used as the development language, in addition, the python libraries that need to be installed in the development environment include eel, shutil, os, imp, ctypes, and the auxiliary dll is everything. dll. The operating system is windows and there should be one browser selected from Chrome, Firefox or Eage, which should run the everything program to help to realize the system-provided file search function.
3. System Function Module Design

3.1. Eel Web services
The function of the Eel Web service in this system is to interact with users and realize graphical plugin. The Eel Web service interface is the main interface used for interaction between the system and the user, which accepts user's request and the response of plugins. At the same time, a set of standard interface functions for background's calling are defined in the web page, so as to complete the interface setting needed by the plugin. The Web service architecture is shown in Figure 2, and the running interface is shown in Figure 3.

![Figure 2. Eel Web Service architecture diagram](image)

![Figure 3. Eel web service running interface](image)

3.2. Utils module
The main function of the Utils module is to provide standard graphical operating functions. Different from the interface functions defined in Eel Web services, graphical operating function is the combined operation of the functions defined in the Eel Web service, which can be used by plugin and realize the preset functions through simple parameter transfer instead of the interface setting operation of
common functions. The Utils architecture diagram is shown in Figure 4.

![Utils architecture diagram](image)

**Figure 4.** Utils architecture diagram

3.3. **Plugin load control class**

As the specific realization of the system's plugin load module function, the plugin load control class is classified into four sub-functions according to the needs of plugin loading, namely, plugin load control class initialization, plugin installation, plugin uninstall, as well as plugin calling and parsing. The plugin calling and phrasing module can also be classified into system-provided functional plugins and custom plugins. The detailed architecture diagram is shown in Figure 5.

![Plugin Load Control Class architecture diagram](image)

**Figure 5.** Plugin Load Control Class architecture diagram

3.3.1. **Initialization.** There are two steps in the process of plugin load control class initialization. The first step is to obtain the project's basic running path, which is also the location of the project's configuration file. The second step is to check whether the configuration file exists. If it does not exist, create a configuration file, otherwise read the list of local plugins from the configuration file.

3.3.2. **Plugin installation.** There are two modes to install plugin. The first is to download the specified plugin from a specified website, which should be used together with the open interface of the specific website. The second is to call the installation method of an existing plugin, which is a strategy of manually calling the installation method to avoid that the error of default calling installation method
leads to the result that the plugin cannot be called later

3.3.3. Plugin uninstall. The plugin uninstall function is to uninstall plugins that are not needed currently, so as to save memory of the running system and improve running efficiency and user experience.

3.3.4. Plugin calling and phrasing. The purpose of the plugin calling and parsing function is to parse the command input by the user in one proper direction of three directions. The three directions are installation and uninstall of plugin, calling and parameter transfer of custom plugin, and calling and phrasing of system default plugin. The calling of system default plugin is at the lowest priority, whose function is to fuzzy search the location of the file with the specified name in the system. Its realization borrows everything program, therefore, using this function requires that the plugin system process coexist with everything process.

4. Conclusions
Characterized by strong robustness and expansibility, this system can provide a variety of functions for ordinary users through plugin architecture, graphical interactive interface. In addition, it is also a platform friendly to individual developers, whose graphical interfaces and standard plugin classes lower the developers’ cost in developing plugins for the platform and converting existing scripts to platform plugins. In this way, individual developers can pay more attention to the realization of core functions, and solve the problems that the core function codes are far less than codes used for graphical function when individual developers develop small scripts.

References
[1] John Hunt: The wxPython GUI Library. Advanced Guide to Python 3 Programming.73-96(2019)
[2] Zhiyong Tang,Kun Liu,Zhenji Yang,Zhongcai Pei,Zeyang Zhang: Object Tracking System for Video Recording based Qt and OpenCV. 2016 International Conference on Mechatronics Engineering and Information Technology. Atlantis Press (2016)
[3] Luo Hongyan,Xu Xu,Gao Chenglong, Li Mingyong, Liao Yanjian: Design and implementation of an automatic analysis system for magnetic resonance quality detection based on QT. Sheng wu yi xue gong cheng xu za zhi = Journal of biomedical engineering = Shengwu yixue gongchengxue zazhi 36(4).627-632 (2019)
[4] Kuntal, Bhusan K., Anirban Dutta, and Sharmila S. Mande: CompNet: a GUI based tool for comparison of multiple biological interaction networks. BMC bioinformatics 17(1), 185 (2016)
[5] Xia Jie,Li Shan,Ding Yu,Wu Song,Wang Xiang Simon: MUBD-DecoyMaker 2.0: A Python GUI Application to Generate Maximal Unbiased Benchmarking Data Sets for Virtual Drug Screening. Molecular informatics (2019)
[6] Pacheco Sayuri, Kaminsky Jesse C,Kochnev Iurii K,Durrant Jacob D: PCAViz: An Open-Source Python/JavaScript Toolkit for Visualizing Molecular Dynamics Simulations in the Web Browser. Journal of Chemical Information and Modeling 59(10), 4087-4092 (2019)
[7] Vagner Figueredo de Santana,Felipe Eduardo Ferreira Silva: User Test Logger: An Open Source Browser Plugin for Logging and Reporting Local User Studies. International Conference on Human-Computer Interaction. Springer, Cham, (2019)
[8] Undefined: OLG Köln: Keine Wettbewerbsbehinderung durch Browser-Plugin – Adblock Plus. 33(2), 115-120 (2017)
[9] Shao Yunqi,Hellström Matti,Mitev Pavlin D,Knijff Lisanne,Zhang Chao: PiNN: A Python Library for Building Atomic Neural Networks of Molecules and Materials. Journal of chemical information and modeling (2020)
[10] Zheng Shijie C,Breeze Charles E,Beck Stephan,Dong Danyue,Zhu Tianyu, Ma Liangxiao, Ye
Wei, Zhang Guoqing, Teschendorff Andrew E: EpiDISH web server: Epigenetic Dissection of Intra-Sample-Heterogeneity with online GUI. Bioinformatics (2019)