Using Python for development of an application for building and experimenting with GPSS simulation models

Aleksandar Hristov
Department "Information Technologies
in Industry"
Technical University of Sofia
Sofia, Bulgaria
ahristov@tu-sofia.bg

***Abstract*—The present paper aims to propose an open source application with graphical user interface for simulating models, created by General Purpose Simulation System (GPSS). The application is similar to the well-known discrete event simulator GPSS. The application has the following functionalities: Open a GPSS file, Enter or edit the model directly in the GPSS model editor window, experimenting with the entered GPSS model and displaying the results on the screen.**

Keywords— GPSS, Python, simulation, software development, Windows application

I. INTRODUCTION

Simulation modeling is preferred when a complex system has to be simulated, because analytical modeling [4] is a very rough approximation of reality. In the simulation model, the behavior of each component of the system is described by a set of algorithms that implement the events occurring in a real complex system.

The model time is modified at the moments that correspond to the occurrence of events in the real system. The processing of simultaneously occurring events is carried out sequentially (constant model time) by the simulation controller.

One of the most widely used simulation language is General Purpose Simulation System - GPSS, originally created by IBM. GPSS [3] is based on the concept of simulating discrete systems by moving dynamic objects (transactions) through static objects (blocks). This movement is performed following a certain logic set by the arrangement of the blocks in the GPSS program. There are multiple versions of the GPSS language, eg GPSS/H, GPSS/PC, GPSS World, as well as web-based ones, eg AGPSS, etc. The different GPSS versions have some differences in the syntax. They also have different limitations in terms of the number of blocks, compatibility with different hardware and operating systems and accessibility. For example, the free version of GPSS World can create models of up to 180 blocks, whereas the web-based versions are not always available 24/7. In order to address this problem, an author's GPSS simulator will be proposed below. For creating of this simulator, it has been chosen Python.

Python is a high level programming language [1] and has a wide range of programming tasks that it can solve. In other words, there is almost nothing that cannot be created with Python. Although Python is primarily used for web development, it is increasingly being used for desktop and mobile application development. Some of the advantages of Python are given below:

- It is an open source project and all the accompanying resources that are needed (modules, libraries and other tools) are also free and available;

- It is known for its productivity and efficiency - Python makes it possible to write programs with fewer lines of code;

- It is compatible with almost all known platforms and operating systems;

- It can handle large volumes of information and complex mathematical calculations;

- It has a huge number of pre-built libraries of code that perform individual functions. These libraries include variety of functions (from mathematical processing to graphics or computer vision).

As technology improves and evolves, the devices that are used are becoming more powerful and faster, which erases Python's only downside - its speed. Also, due to the Python multiplatform abilities, the applications build with it have some limitations related to the Graphical User Interface (GUI).

The present paper aims to propose a GPSS simulator as well as to describe the process of creating it and to conduct an experiment with model of multiprocessors for validating the simulation results. Python has been chosen as a programming language for creating this simulator.

II. APPLICATION FOR BUILDING AND EXPERIMENTING WITH GPSS SIMULATION MODELS

In this section, an open source Windows application with graphical user interface for simulating models, created by General Purpose Simulation System (GPSS) has been proposed. This application is similar to the visual environment for simulation of wireless networks [2]. As it was mentioned above, Python was used for the implementation of the application. Screenshot of the main screen of the developed application is shown in Fig. 1.

The application has the following functionalities:

979-8-3503-0329-2/23/$31.00 ©2023 IEEE

1. Open GPSS file – by clicking the 'Open GPSS file' a file dialogue is opened and the user should choose a GPSS file with .gps extension. The file is then loaded into the GPSS model editor - left part of the main screen;

2. Enter or edit the model directly in the GPSS model editor window;

3. Save to file - saves the model to a file named model.gps in the same folder as the application by clicking the 'Save To File' button;

4. Start GPSS Simulation – by clicking the 'Start GPSS Simulation', the model from the editor is saved to a file named model.gps and simulation starts. This gives the opportunity for experimenting with the entered GPSS model and displaying the results on the right part of the screen.

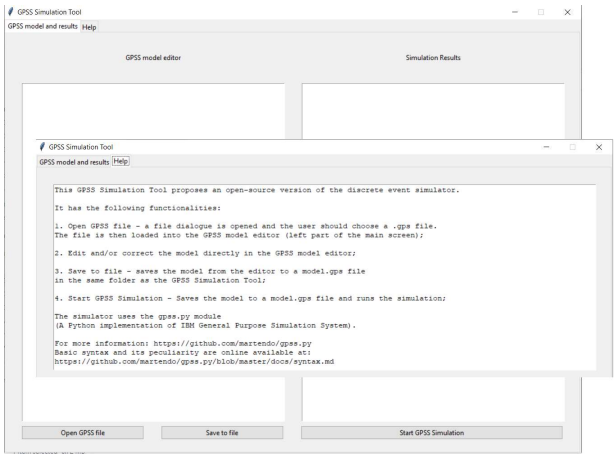

Fig. 1. Screenshot of the developed application

The application uses the gpss.py package [7], which is an open-source Python implementation of the GPSS system for simulating the models..

III. Implementation of the Application

Due to the limited scope of the present paper, a brief description of the more important parts of the Python code is given here. Full Python code is available at [6]. Below the steps of the implementation are given:

- The modules tkinter and gpss are imported
- The main window (root) is created
- Two tabs (GPSStab and helpTab) are created and added to the main window
- The tabs are split into rows and columns
- Three buttons (openFilebtn, saveToFilebtn and startGPSSbtn) are created and added into the grid
- Two textboxes (GPSSmodel and resultsFromGPSSTextbox) are created and added into the grid for the GPSStab and another one (infoTextbox) is created and added into the grid for the helpTab
- The openFile function is created
- The saveToFile function is created
- The startGPSS function is created

Below these 9 steps are described.

The modules tkinter and gpss are imported. Python has many GUI frameworks, but Tkinter is the only one built into the Python standard library, and is therefore the preferred choice for GUI application development. As it was mentioned above, Python applications have some limitations related to the Graphical User Interface (GUI). The implementation of the GUI of the proposed application is simple, but user-friendly.

The gpss package includes open-source implementation of the well-known discrete event simulator – GPSS and executes the statements that create a model. The gpss package is able to execute the following statements: GENERATE, QUEUE, ADVANCE, DEPART, TERMINATE with the relevant operands.

The main window (root) is created with the following parameters: title, dimensions, etc.

The tabs of the main window are created. The GPSStab is the main one and in it the user has the opportunity to enter or edit the model directly in the editor of GPSS models. Also in the right part of the screen, the results of the started simulation are available. The buttons for controlling the simulator are located at the bottom part of the screen. The second tab (helpTab) is where the user gets acquainted with the functionalities of the simulator and how it works. Those two tabs are added to the root window via the Notebook widget from the tkinter module.

The grid geometry manager is used in order to make the design responsive and to give the user the ability to resize the window of the application. Grid geometry manager uses the concepts of rows and columns to arrange the widgets. The GPSStab is split into 3 rows with the rowconfigure method. Also, the GPSStab is split into 2 columns with the columnconfigure method. The helpTab contains only one textbox so it is not split and contains one row and one column.

The openFilebtn, saveToFilebtn and startGPSSbtn buttons are added. The Button method from the ttk module is used for creating them. The buttons are then deployed in the grid using the grid method, with the relevant parameters such as row and column index, paddings and relative placement on the screen. The openFilebtn button is used to open a GPSS file dialog for selecting a file with the appropriate extension. The saveToFilebtn button is used to save the edited model to a file named model.gps. The startGPSSbtn button is used for starting the simulation. This gives the opportunity for experimenting with the entered GPSS model and displaying the results on the right part of the screen.

The textboxes GPSSmodel and resultsFromGPSSTextbox are added. The Text method from the tk module is used for creating them. The textboxes are then deployed in the grid using the grid method, with the relevant parameters such as row and column index, paddings and relative placement on the screen. Similarly, an infoTextbox textbox is added in the Help tab of the application.

The openFile function is created, which is called when the openFilebtn button is pressed. When the button is clicked a file dialogue is opened and the user should choose a GPSS file with .gps extension. The file is then loaded into the GPSS model editor - left part of the main screen.

The saveToFile function is created, which is called when the saveToFilebtn button is pressed. With this function the contents of the GPSSmodel textbox are written to a file named model.gps using the writelines method.

The startGPSS function is created, which is called when the startGPSSbtn button is pressed. It first clears the contents of the resultsFromGPSSTextbox textbox, and then calls the saveToFile function. The simulation is then started using the gpss.run method, and the result is saved using the gpss.createReport method. Finally, the result from the simulation is shown in the resultsFromGPSSTextbox textbox and the user can view or copy it. If there is a GPSS parser error, it is shown in the resultsFromGPSSTextbox textbox so that the user edit the model..

IV. Creating an Executable File from the Python Script

The Python script for the application is used for creating a Windows executable file using the PyInstaller tool, following these 3 simple steps:

- Open the Command Prompt (CMD);

- Navigate to the directory containing the Python script for the application;

- Execute the command: pyinstaller --onefile gpssGUI.py.

The approach for creating an executable file for MacOS X and Linux is similar.

V. Experiment

There are two ways of conducting laboratory exercises with students – from distance or online [9],[10] and traditional (in-laboratory). The author of this paper is PhD and assistant with specialty "Computer systems, complex and networks" at Faculty of computer systems and technologies and conducts laboratory exercises in various disciplines. One of them is "High Performance Computer Systems". Below is given a brief description of the course:

- It is compulsory for the students of specialty Computer and Software Engineering in the bachelor programme of the Faculty of Computer Systems and Technologies.

- The purpose of the course is to give the students a good understating of concepts and mechanisms related to the design of modern high-performance computer systems as well as to be able to apply the main architectural styles.

- Main topics of are: Taxonomy; Scalable high-performance computer systems; Vector processors; Massive parallel processors; Clusters of servers and workstations; Symmetrical and CC-NUMA multiprocessors; System interconnection networks for high- performance computer systems and complexes; Parallel GPU architectures; Supercomputers.

- The teaching methods include: lectures using video-presentation with beamer; laboratory exercises ending with presentation of the results, parallelism profiles and estimation of the performance parameters of the parallel system for the certain task.

One of the laboratory exercises is related to analytical and simulation investigation of multiprocessors. Multiprocessors with multiport memory is shown on fig. 2.

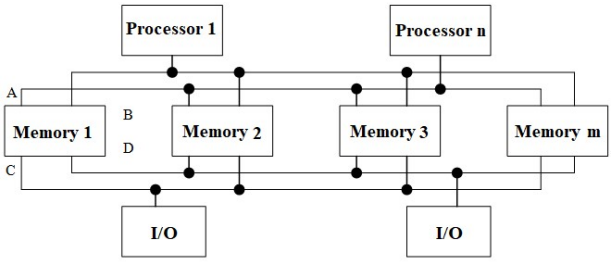

Fig. 2. Screenshot of the developed application

The purpose of this exercise is to create a simulation model for investigating the memory and processors

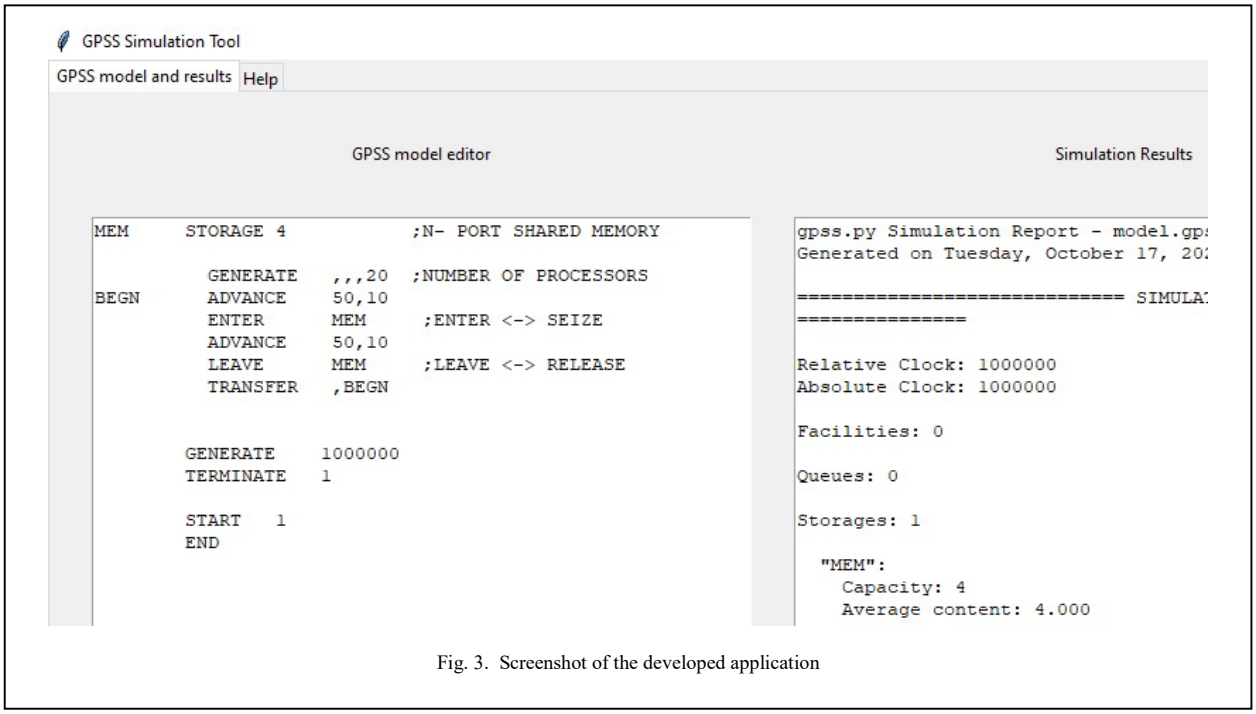

Fig. 3. Screenshot of the developed application

utilization of the multiport memory multiprocessor and to conduct experiments by changing values of the selected controllable factors, and finally summarizing and analyzing the obtained experimental results.

In this section, an experiment with a GPSS model of multiprocessors (fig. 3) has been conducted in order to validate the simulation results of the application.

At the beginning, the GPSS model (left part of the screenshot, fig. 3) includes definition of number of shared memory ports through the MEM STORAGE statement. The block GENERATE is used to create as many transactions as the number of processors there are in the multiprocessor. Then, in a loop starting with the BEGN label, the following actions take place: the transaction is held for a certain amount of time, corresponding to CPU processing time before it needs to access the memory; then transactions occupy the storage for a time corresponding to a successful memory access and performing load/store operation (the time in the second ADVANCE block) and finally transactions leave the storage (corresponding to releasing the occupied memory port).

As it can be seen, the experiment has been conducted with the following input factors:

- number of memory ports – 4 (ports A, B, C, D, fig. 2);

- number of processors – 20;

- processing time for 1 iteration of the loop – 50 ± 10 units;

- time for performing operation with memory – 50 ± 10 units.

As it can be seen from the right part of the screenshot, fig. 3, for these input factors, the average utilization of memory is 100%, the average processing time is 50.026 and the number of executed tasks (Entries) is 79954. Utilization of processors is calculated by the following manner:

$$utl = 79954.50/(20.1000000) = 20\% \quad (1)$$

The 100% memory usage and 20% processor usage means that the memory is the bottleneck in that multiprocessor.

The result for Entries is 79954, while the result from GPSS World Student version is 80004. The difference between the two values is 0.06%. One possible explanation for this difference is the different random number generators that these GPSS interpreters use. All other results are the same. This can be considered as a verification of the correct operation of the proposed application.

VI. Conclusion

In the present paper, an open source Windows application with graphical user interface for simulating models, created by General Purpose Simulation System (GPSS) has been proposed.

The functionalities of proposed application and introductions for using it have been described, thus the paper is a kind of user guide. The proposed visual environment is very convenient for both research and educational purposes.

The developed application allows integration and implementation with external forms for entering parameters of a specific GPSS model.

As a future work, through a user-friendly form for users unfamiliar with the GPSS simulation system, investigation will be done for the time of conducting successful DDoS attacks in IIoT networks [8], initialization time of a secure exchange using Zero-knowledge Proof and Smart Questioning [5] and etc.

Acknowledgment

The presented research is for the scientific-research project № КП-06-ПН47/27 “Possibility Investigation of Increasing the Cybersecurity of the Systems in Industry 4.0 using Artificial Intelligence” funded by National Science Fund under the Ministry of Education and Science in Bulgaria with contract КП-06-Н 47/7.

References

[1] Hunt, J. Advanced Guide to Python 3 Programming, 2nd Edition, Springer, 2023

[2] HRISTOV, V. A visual environment for simulation of 802.11n wireless networks, BJED, no10, 2012, pp. 37-44 (in bulgarian)

[3] Mitrev, R. Computer modeling and Simulation, Propeler,Sofia, 2021 (in bulgarian)

[4] Romansky R., Noninska I. Software tools for analytical modeling of workload, AIP Conference Proceedings · September 2022, DOI: 10.1063/5.0100691.

[5] Stefanova-Stoyanova, V., Stankov, I. and Danov, B. Exploring the Synergy between Zero-knowledge Proof and Smart Questioning, Proceedings of the 11th International Scientific Conference COMPUTER SCIENCE, IEEE Conference, Rec # 59259, 18 – 20 September, Sozopol, Bulgaria

[6] Hristov, A. GPSS Simulation Tool online available: https://github.com/sashkinaaa/gpssGUI

[7] Documentation of gpss.py package online available: https://github.com/martendo/gpss.py

[8] Hristov, A. et al. Developing and experimenting simulation models of DDoS attacks in IIoT networks using Python, "TELECOM 2023" (IEEE Conference record # 59629), Sofia, 16 – 17 November 2023, submitted

[9] Hristov, V., et. al. Conducting a cycle of remote laboratory exercises, International Scientific Conference Computer Science’2020, Velingrad, Bulgaria, October 18th – 21th, 2020, ISBN: 978-619-167-177-9, pp.166-172

[10] Nachev, V. Web-Based Remote Laboratory For Programming Arduino-Based Experiments, "TELECOM 2023" (IEEE Conference record # 59629), Sofia, 16 – 17 November 2023, submitted