Acta Scientific COMPUTER SCIENCES

Volume 3 Issue 10 October 2021

Short Communication

ZeroOne: Building and Enhancing Executing Simulation by Incremental Patches

Maurice HT Ling[1,2]*

[1] HOHY PTE LTD, Singapore

[2] School of Data Sciences, Perdana University, Malaysia

***Corresponding Author:** Maurice HT Ling, HOHY PTE LTD, Singapore and School of Data Sciences, Perdana University, Malaysia.

Received: August 17, 2021

Published: September 25, 2021

Abstract

Identifying all the required aspects before building a simulation is one of the major difficulties. This may be resolved by incremental simulation building. However, the simulator must be able to accept new codes into the simulation while the simulation is running. Here, I present ZeroOne, a simulation engine which allows for incremental simulation building by monitoring and processing a script file for new codes. ZeroOne is licensed under GNU General Public License Version 3.0 for academic and not-for-profit use.

Keywords: Simulator; Incremental Simulation Building; Matrix Trilogy; Python Programming Language

Introduction

The Matrix Trilogy has been a subject of many philosophical analyses [1-3]. One of the deepest questions raised is on the nature of reality [4] - Are we in a simulation? Are our experiences real? Bostrom [5] argues that the possibility that we are living in a simulation is high with technological advancements. This is known as simulation hypothesis and is supported by Chung [6] whom argues that the quantum physics is more consistent with digital space (discretized space) rather than continuous space in Newtonian mechanics; thus, suggesting that our physical reality is simulated. However, Bibeau-Delisle and Gilles Brassard [7] and Kipping [8] argue that the possibility that we are living in a simulation is less than 50% if considering that any advanced civilization will produce their own simulation [9], resulting in recursive simulations. Nevertheless, Hamieh [10] argues that the simulation hypothesis can be a useful tool in understanding and testing our physical reality.

One of the difficulties in simulation building is trying to identify all the required aspects before building the simulation [11,12]. This is similar to the Waterfall model [13] with its inherent difficulties [14]. Incremental simulation building using patches is increasing in popularity in software development [15] and may be suitable for simulation building. In Matrix Trilogy, The Merovingian (commonly known as the Frenchman) wrote a code in the form of a cake to be executed by a patron at Le Vrai restaurant by consumption. Similarly, Oracle might have written her codes in the form of baked cookies for Neo. This implies that the simulator must be able to accept new codes for execution while the simulation is running.

In this article, I present ZeroOne (name for the Machine City in Matrix Trilogy) as a simulation engine which allows for incremental simulation building using patches. Developed in Python programming language, the crucial component for loading of new codes while the simulation is running is achieved by reloading of a monitored module containing the new codes. ZeroOne is available at https://github.com/mauriceling/zero_one and is licensed under

GNU General Public License Version 3.0 for academic and not-for-profit use.

Implementation and usage

The core of a simulator is a simulation loop [16-18]. The purpose of the simulation loop in ZeroOne is to execute all the programs sequentially as each program is defined as a function. Hence, the pseudocode for ZeroOne simulator can be defined as

```
programBag ← list of programs
environmentBag ← list of parameters
while True:
 for ID in programBag:
  $environmentBag_{ID}=programBag_{ID}(environmentBag_{ID})$
 timecycle = timecycle + 1
```

Each element in the `environmentBag` corresponds to the parameter used by the corresponding program in the `programBag` by position. When the simulator is executed, two programs (`SUArchitect` and `SUReporter`) were loaded into `programBag` as the first and second program respectively. This results in the execution of `SUArchitect` and `SUReporter` at every cycle. Program `SUReporter` prints out the high-level status of the simulation, such as the contents of `programBag` and `environmentBag`. Program `SUArchitect` monitors a Python script file (denoted as `architect`) for additional codes to load into `programBag`. This is achieved by reloading `architect` module, and checking for new elements in `architect.archCode` and `architect.archEnv` variables. The variables `architect.archCode` and `architect.archEnv` contain programs to be loaded and its parameters, respectively. Therefore, if there are programs in `architect.archCode` that are not in `programBag`, will load them into `programBag` and its corresponding parameters into `environmentBag`. This allows for the new programs to be executed in the next simulation cycle.

Based on this implementation, incremental simulation building is essentially a continual coding of programs into architect module. When the new program is ready for load and execute in the next simulation cycle, the new program and its parameters are loaded into `architect.archCode` and `architect.archEnv` respectively. Sample codes for architect module is in ZeroOne's repository as `trial_architect.py` file. Finally, `python zeroone.py [architect]` is the command to start ZeroOne.

Conclusion

ZeroOne is a simulation engine capable of accepting new codes while the simulation is running; thus, providing the ability to enhance executing simulations incrementally using patches without halting and restarting the simulation.

Supplementary Materials

The introduction video for ZeroOne is available at https://bit.ly/01Demo1.

Data Availability

ZeroOne repository is hosted in GitHub, https://github.com/mauriceling/zero_one, and licensed under the GNU General Public License Version 3.0 for academic and not-for-profit use.

Conflict of Interest

The author declares no conflict of interest.

Bibliography

1. Milidrag P. "Platonism, Cartesianism and Hegel's Thought in the Matrix Trilogy". *Filoz Drustvo* 24.4 (2013): 268-282.

2. Constable C. "Baudrillard Reloaded: Interrelating Philosophy and Film via The Matrix Trilogy". *Screen* 47.2 (2006): 233-249.

3. King CR and Leonard DJ. "Racing the Matrix: Variations on White Supremacy in Responses to the Film Trilogy". *Cultural Studies ↔ Critical Methodologies* 6.3 (2006): 354-369.

4. Maudlin T. "Physics, Philosophy, and the Nature of Reality: Physics, Philosophy, and the Nature of Reality". *Annals of the New York Academy of Sciences* 1361.1 (2015): 63-68.

5. Nick B. "Are You Living in a Computer Simulation?" *Philosophical Quarterly* 53.11 (2003): 243-255.

6. Chung D-Y. "We Are Living in a Computer Simulation". *Journal of Modern Physics* 07.10 (2016): 1210-1227.

7. Bibeau-Delisle A and Brassard G. "Probability and Consequences of Living Inside a Computer Simulation". *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 477.2247 (2021): 20200658.

8. Kipping D. “A Bayesian Approach to the Simulation Argument”. *Universe* 6.8 (2020): 109.

9. Ling MH. Re-creating the Philosopher’s Mind: Artificial Life from Artificial Intelligence”. *IConcept Journal of Human-Level Intelligence* 3 (2012): 1.

10. Hamieh S. “On the Simulation Hypothesis and Its Implications”. *Journal of Modern Physics* 12.5 (2021): 541-551.

11. Ling M. “Of (Biological) Models and Simulations”. *MOJ Proteomics and Bioinformatics* 3 (2016): 00093.

12. Rice CC. “Factive Scientific Understanding Without Accurate Representation”. *Biology and Philosophy* 31.1 (2016): 81-102.

13. Royce WW. “Managing the Development of Large Software Systems: Concepts and Techniques”. In: Proceedings of IEEE WESCON (1987): 328-338.

14. Sherman R. “Project Management”. In: Business Intelligence Guidebook. Elsevier (2015): 449–492.

15. Dadzie J. “Understanding Software Patching: Developing and Deploying Patches is an Increasingly Important Part of the Software Development Process”. *Queue* 3.2 (2005): 24-30.

16. Bratley P., *et al*. “Simulation Programming”. In: A Guide to Simulation. New York, NY: Springer US (1983): 214-266.

17. Castillo CF and Ling MH. “Digital Organism Simulation Environment (DOSE) Version 1.0.4”. In: Current STEM, Volume 1. Nova Science Publishers, Inc. (2018): 1-106.

18. Castillo CF and Ling MH. “Digital Organism Simulation Environment (DOSE): a library for ecologically-based in silico experimental evolution”. *Advances in Computer Science: An International Journal* 3.1 (2014): 44-50.

Volume 3 Issue 10 October 2021