Simple Programming Language for Creating a Simulation Environment with Mobile Robots

Young Joon Kim and Yong-Ho Seo*

Department of Intelligent Robot Engineering, Mokwon University
88 Doanbuk-ro, Seo-gu, Daejeon, Republic of Korea
young.admin@gmail.com, yhseo@mokwon.ac.kr

Abstract

This paper proposes a simple programming language that is a kind of functional concise script for creating a simulation environment with mobile robots as a 3D software toolkit for education. Most robotics simulation toolkits require professional knowledge regarding development language and their APIs, and these prerequisites present a significant obstacle in implementing robotics simulations. In order to resolve these problems, we adopted a functional concise script composed of simplified service commands and minimum options to build a simulation environment. In experiments, we educated high school students and teachers on how to use the proposed robot simulation creation toolkit, and then verified that they could build a robotics simulation environment and control simulated robots easily in a two-day training course.

Keywords: *Robotics Simulation, 3D Toolkit for Education, Functional Concise Script, Mobile Robot*

1. Introduction

Robot simulation is essential not only for commercial robotic products but also for research and education of robotics technology. To develop a robot, it is necessary to complete several procedures such as design validation and data acquisition. These procedures can be carried out via a robot simulation before actual robot development. In addition, robot simulations are anticipated to become popular among the public and to be more widely used by robotics researchers as well; possible examples include education materials for enhancing learner's creativity and learning tools for robotics majors who want to develop robots and experience practical exercises [1].

Although robot simulation is extensively employed in robot development, there is no tailored simulation program due to the difficulty of refactoring the program for public users. Due to this reason, robot simulations have not been disseminated widely. Robot simulation is a challenging field for robotics researchers, requiring not only knowledge of robotics but also other professional knowledge such as 3D rendering, expertise in physical engines, *etc.* [2].

In this research, to resolve this problem, we propose a functional concise script based execution environment named 'SPL (simple programming language)', which can be used to produce a diverse simulation environment and robot simulation without training. It is expected that only two days will be needed for novice users who do not have prior knowledge about robotics to design their own robot.

* Corresponding Author

2. Development Procedure for Robot Simulation

Table 1 introduces the products of a simulation program and its supporting language. It is easy to recognize that most products support general programming languages such as C, C++, C#, and Java [3].

Table 1. Robotics Simulation Tools and Development Languages

Prod. Name	Prod. Type	Language
Gazebo	Open Source	C++
MSRDS	Commercial	C#
Virtual Robot Experimentation Platform	Commercial	C/C++
AnyKode Marilou	Commercial	C/C++, C++ CLI, C#, J#
Webots	Commercial	C/C++, Java, Python, URBI

There are various types of robot simulation programs such as commercial products including MSRDS (Microsoft Robotics Developer Studio) and open source programs such as Gazebo. Each program supports its own program languages [4-5]. Figure 1 shows an example of a MSRDS robot simulation environment.

Figure 1. An Example of a MSRDS Robot Simulation Environment

Even though a programmer is already skillful in general programming languages such as C, C++, C# and Java, it still takes at least three months to carry out a robot simulation quite freely. This can be a barrier for them to use robot simulations freely. Therefore, in this research, we propose a functional concise script for constructing a simulation environment and controlling a simulation robot. This script can be used easily even by beginners and laymen. We show the effectiveness of our script through the results of an experiment conducted in a high school class.

3. Functional Concise Script, SPL

In general, a robotics simulation is composed of many commands, components, and complex options. If these commands and options are constructed from general programming languages, it is necessary to learn many codes, thus placing a burden on the user.

Functional concise script reduces a number of commands and minimizes the use of command options by applying Off-The-Shelf service. Each service is simplified by its own commands and other options, as shown in Figure 2.

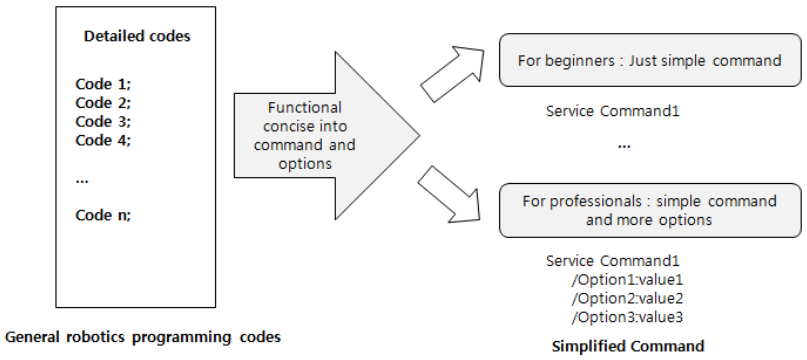

Figure 2. Functional Concise into Service Commands and Options

The key point of the functional concise script is that frequent options are configured as default settings and not revealed to the users. Novices generally do not need to know the detailed options and only simplified commands are required because all detailed parameters can be set with default values. Professionals, on the other hand, can access more detailed functions by using more exposed options, as shown in Figure 3. This means that the level of conciseness can be controlled by exposed options.

In this research, we defined two types of script: definitional script and control script. Most robotics entities and simulation environments are defined by using definitional script. This definitional script is composed of commands and options, as shown in Figure 4. Its context follows a general command-line OS command, which includes options so that the developer can learn the instructions easily.

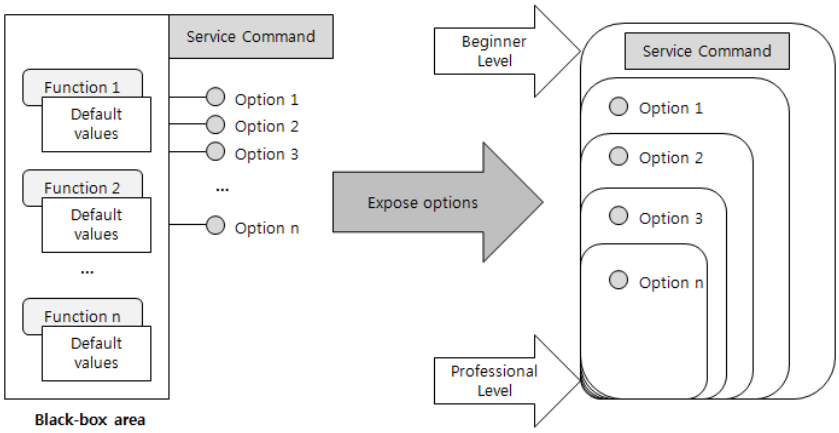

Figure 3. Manipulation of Functional Level by Exposing Options

The control script is used for flow control while the iteration structure and definitions of functions for providing functionalities are similar to a general programming language. It is designed to follow JavaScript grammar [6], as shown in Figure 4.

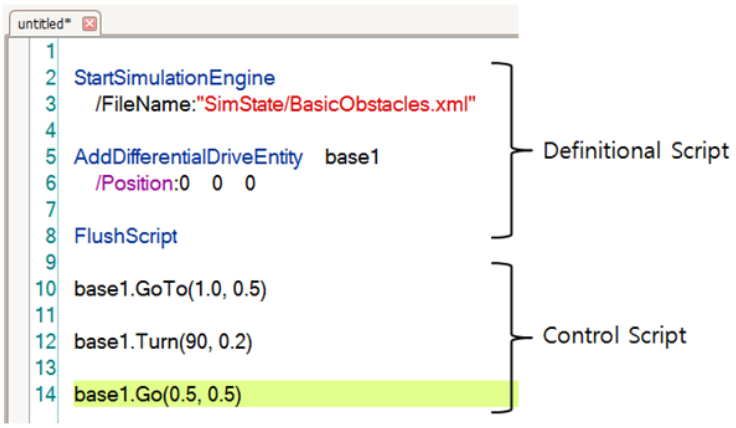

Figure 4. Script Parts for Defining Mobile Robot

Figure 5 shows an example of a graphical diagram for functional concise script. Because the functional concise script is composed of simple commands, it can be converted to a graphical diagram easily.

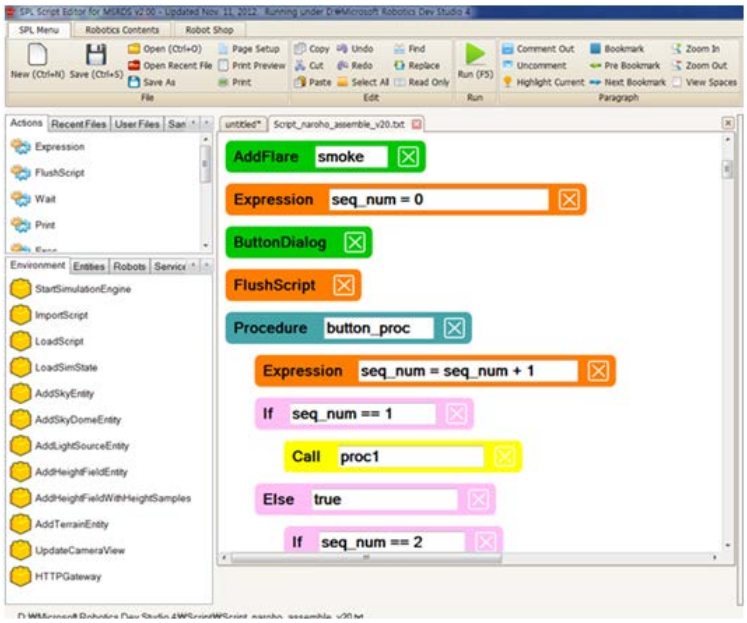

Figure 5. Graphical Diagram of Functional Concise Script

Figure 6 shows a flow chart of an execution of functional concise script. First, the script engine separates input commands into two types, control commands and definitional commands. Definitional commands build the environment and robotics entities and control commands control the defined entities

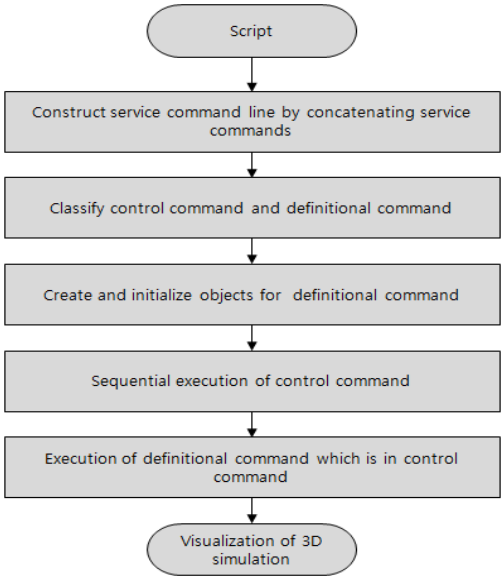

Figure 6. Flow Chart for Executing Script

4. Experiments

We implement functional concise script based on a MSRDS R4 version simulation and conduct an experiment following the procedure outlined below.

1. Construction of obstacle environment
- Two obstacle boxes
2. Mobile robot
- Two wheels, LRF sensor, and bumper sensor
3. Avoidance of obstacles
- Robot moves through the obstacles based on the LRF sensor data

4.1. Simulation Environment Composition Script

Various elements such as the floor, sky, and sun are located properly in window form. In addition, two long boxes are added as obstacles, as shown in Figure 7.

```
StartSimulationEngine "SimState/basicsim.xml"

AddNewEntity  ent1  /Position:0 0.25 3

AddBoxShape
  /Dimensions:10 0.5 0.5  /Mass:10

AddNewEntity  ent2  /Position:0 0.25 -3

AddBoxShape
  /Dimensions:10 0.5 0.5  /Mass:10
```

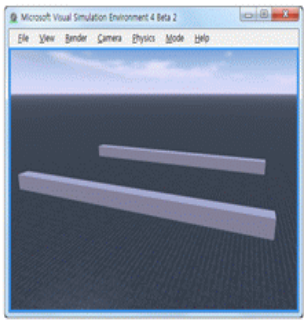

Figure 7. Script for Definition of Environment

4.2. Mobile Robot Composition Script

The mobile robot is equipped with two wheels, a bumper, and a laser ranger finder. We use Pioneer's P3DX for 3D modeling, as shown in Figure 8.

```
AddDifferentialDriveEntity   base1
  /Position:0   0   0
  /Mesh:"Pioneer3dx.bos"
  /WheelMesh:"PioneerWheel.bos"

AddBumperEntity   bumper1
  /ParentEntity:base1

AddLaserRangeFinderEntity   lrf1
  /Position:0  0.4  0
  /ParentEntity:base1
```

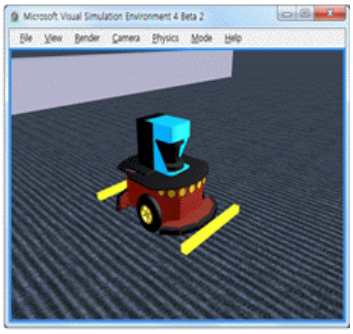

Figure 8. Script for Definition of 3D modeling Mesh

4.3. Sensor Application Script

The robot senses the distance between its location and the obstacles. It rotates 180 degrees and moves forward when the distance is lower than a certain threshold. The distance is measured by a simulated LRF (Laser Range Finder). Figure 9 shows an example of script for mobile robot control and its simulation result of execution the script.

```
FlushScript

base1.Go(0.4, 0.4)

while (true)
{
   distances = lrf1.Get()
   d180 = distances.DistanceMeasurements[180]

   if (d180 < 1000)
   {
      base1.Turn(180, 0.3)
      base1.Go(0.4, 0.4)
   }
}
```

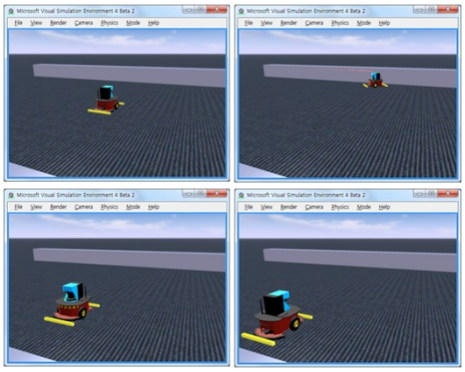

Figure 9. Example of Script for Mobile Robot Control and its Simulation Result

4.4. Experimental Results

In order to verify the performance of functional concise script, a four day training course was carried out with 50 high school students. These students were trained for two days on the use of functional concise script and an additional two days were provided for a team project. Figure 10 shows their implementation results. Table 2 presents a comparison of required time to train students for building the robotics simulation.

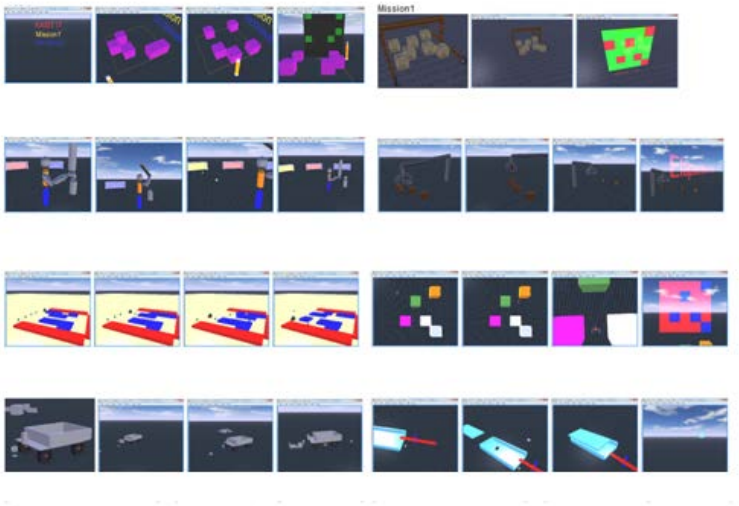

Figure 10. Examples of Implementation Results for Test Groups

Table 2. Comparison of Required Training Time for Building Robotics Simulation

Subjects	Required Time	
	Using Previous Programming Tools	Using Proposed Functional Concise Script
Development Tool Training	1 Months	2 Hour
Robotics Simulation API or SDK Training	1 Months	-
3D Modeling Training	1 Months	-
Entity Design	1 Week	2 Hour
Robot Design	1 Week	2 Hour
Robot Control	1 Week	1 Hour
Joint Design and Control	1 Week	3 Hour
Sensor Design and Control	1 Week	2 Hour
Total	4 Months	12 Hours

As shown in Table 2, generally about 4 months are needed to take all prerequisite classes for robotics simulation and thus robotics simulation subject is usually composed of a one semester curriculum. However, when using the proposed functional concise script, only 12 hours are needed to train high school students.

5. Conclusion

We propose a programming language called functional concise script for robot simulation that is useful for beginners. We believe that it can contribute to propagating robot simulation programs. Beginners who do not have prior knowledge about robotics can handle this program after a short training, as verified through an experiment. They can produce a simulation robot after only two days of training. In future work we will focus on developing a mobile device based script that is more portable and convenient.

Acknowledgements

This work was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (MEST) (2011-0013776). This work was also supported by the NAP (National Agenda Project) of the Korea Research Council of Fundamental Science & Technology.

References

[1] S. F. Railsback, S. L. Lytinen and S. K. Jackson, "Agent-based simulation platforms: Review and development recommendations", Simulation, vol. 82, no. 9, **(2006)**, pp. 609-623.
[2] Y. J. Kim, "MSRDS Simulation Environment and External Interface", Korea Robotics Society, Spring Edition, **(2010)**, pp.16-21.
[3] L. Brakmo and L. Peterson, "Experiences with Network Simulation", Proc. ACM SIGMETRICS 96, ACM Press, New York, **(1996)**, pp. 80-90.
[4] Microsoft Robotics Developer Center, http://msdn.microsoft.com/robotics.
[5] O. Almeida, J. Helander, H. Nielsen and N. Khantal, "Connecting Sensors and Robots through the Internet by Integration Microsoft Robotics Studio and Embedded Wen Services", Proceeding of IADIS International Conference, **(2007)**.
[6] HelloApps, http://www.helloapps.com.

Authors

Young Joon Kim received his BS degree from the Department of Science Education, ChungBuk National University, in 1993 and received MS degree from the Department of Computer Science, ChungBuk National University, in 1995. He was a lead of web development team at eBay Korea in 2002 and was a robotics evangelist at Microsoft Robotics Group between 2002 and 2009. He is currently CEO and Chief Architect of HelloApps.com and a Ph.D. candidate at the Intelligent Robotics Laboratory, Mokwon University. His research interests include robotics simulation and robotics authoring tool and STEM education.

Yong-Ho Seo received his BS and MS degrees from the Department of Electrical Engineering and Computer Science, KAIST, in 1999 and 2001, respectively. He also received a PhD degree at the Artificial Intelligence and Media Laboratory, KAIST, in 2007. He was an Intern Researcher at the Robotics Group, Microsoft Research, Redmond, WA in 2007. He was a consultant at Qualcomm CDMA Technologies, San Diego, CA in 2008. He is currently a Professor of the Department of Intelligent Robot Engineering, Mokwon University. His research interests include humanoid robot, human-robot interaction, robot vision and wearable computing.