Development of Software for Simulation Learning Utilizing the Features of Java and Processing 4

Toshiaki Yokoi
Tokyo City University, Japan

Abstract

This paper describes various original simulation programs developed for a "Computer Simulation" class. Java and Processing 4 were chosen for the development of the simulation programs because of their excellent graphical user interface and visualization capabilities. I developed application programs with the same functionality using Java and Processing 4, and compared their development manhours, amount of code, and time required for development. The results showed that Processing 4's capabilities exceeded my expectations. For the modeling of physical simulations, the "Fisica" library for Processing 4 proved to be more capable than expected. With "Fisica", I was able to create complex physical models in a small number of lines of program code that were easy to understand. Next, I took up the "Brachistochrone problem," one of the famous variational problems, and compared the theoretical solution, the numerically approximate solution using piecewise functions, and the solution using "Fisica". As a result, the numerical approximate solution using piecewise functions showed a high accuracy that was almost identical to the theoretical solution. On the other hand, the "Fisica" solution, although less accurate, is closer to the actual behavior and is more interactive, making it suitable for introductory learning. Finally, the "Sheepdog Project" was discussed as a practical project assignment for a constructivist approach to animal behavior. By designing the thinking of a sheepdog to drive a flock of sheep into a sheep pen in a reliable and short time, I was able to confirm the students' interest and engagement with the task.

Keywords: Fisica, Java, Processing 4, Visual illusion

1. Introduction

The course "Computer Simulation" introduces concepts of analytic modeling and computer simulation, using projects drawn from multidisciplinary areas of computational engineering science [1]. Models progress sequentially through problem statement, mathematical modeling, computer modeling, qualitative and quantitative property, summarization for decision makers (see Figure 1).

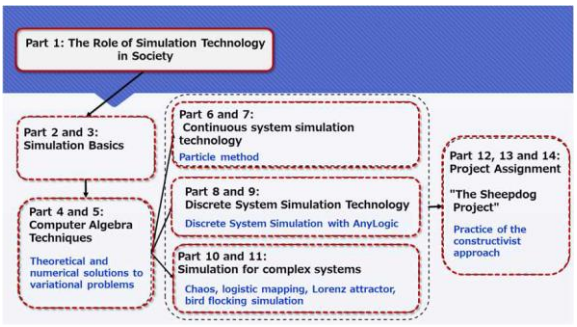

Figure 1. Graphical syllabus of the course "Computer Simulation" at Tokyo City University

2. Utilizing Java and Processing in Education

2.1. Early adoption of Java for education

In the first curriculum of our former department "the department of environment and information" established 1997 at the Yokohama campus, we chose the Java for the programing language in education since it is appropriate for the Internet society. Also, the graphical user interface and multimedia functions are very attractive in education to stimulate student motivation to learn.

2.2. Features of Processing 4

The Processing 4 [2] is a Java-based language which provides easy programming development environment with wide variety of libraries. Overview of its features is as follows:

- Simple development screen, simple basic program structure (see Figure 2).
- Plenty of libraries are available (Open CV, Fisica, etc. (see Table 1).
- Java integration is available because it is written in Java.

This feature makes the Processing application program development easier for the people whose

 | DOI: 10.20533/ijicr.2042.4655.2023.0142

primary occupation is not programming (artists, architects, teachers, composers, data scientists, etc.). Furthermore, there are plug-ins for NetBeans IDE and IntelliJ IDE to provide useful and real-time assist for Processing 4 program development.

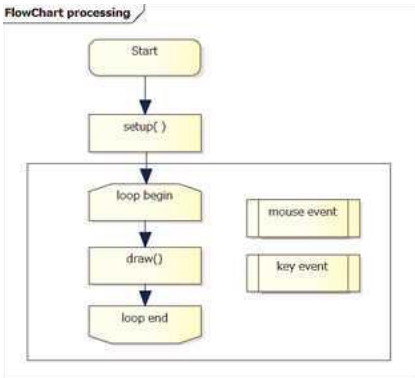

Figure 2. Fundamental program flow in Processing 4

Table 1. Primary function categories for Processing

Category	Primary Contents
3D	3D Graphics
Animation	Animations
Compilation	Packaging of various functions
Data	Data Communications
GUI	Graphical User Interface
Geometry	Generation of 2-D and 3-D geometric figures
Hardware	Interfacing with sensors and other hardware
I/O	Data input/output
Language	Natural Language Processing
Math	Mathematics
Simulation	Physics simulation and flocking simulation
Sound	Sound reproduction, sound generation and analysis
Typography	Typography
Utilities	Utilities
Video & Vison	Video and Computer Vision
Other	Miscellaneous

3. Comparison of Java and Processing 4 Features by Example

3.1 Visual illusion program development

In the visual illusion, we take the Münsterberg illusion as one example of a basic, interactive program. The Münsterberg illusion was introduced by the physician and psychologist Munsterberg in 1897. When squares of white and black, etc., are shifted to each other above and below a straight line, the straight line at the boundary of the squares appears to tilt. Interactivity was incorporated by using the mouse event function to shift the horizontal position of the entire row of squares by dragging the mouse over the row of squares. Figure 3 shows the screen in Processing 4. To compare the program volume, I compared the number of program lines after reformatting on the NetBeans IDE and the Processing 4 IDE, with all comment lines removed. The Java program had 113 lines, while the Processing 4 program had 79 lines, indicating that the exact same functionality can be described more concisely in Processing 4.

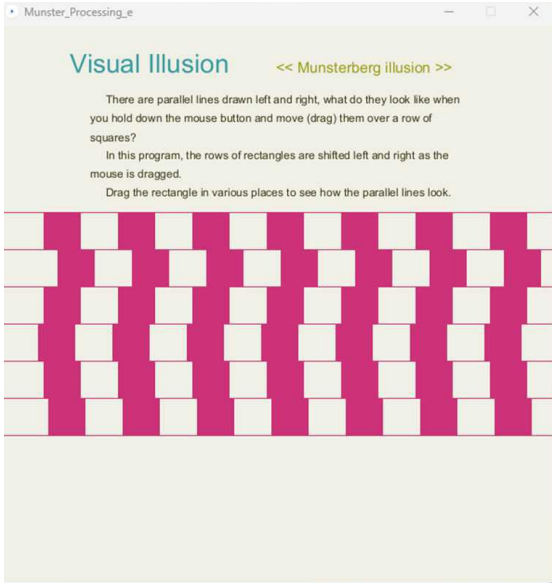

Figure 3. Munsterberg visual illusion by Processing 4

3.2. Hooked on Java examples [3]

3.2.1. "Neon sign" example. This example consists of a simple program structure that switches images at random times between the image of a glowing Neon tube and the image of a tuned-off Neon tube to represent a blinking neon sign (see Figure 4).

Figure 4. "Neon Sign" example

The length of the source code was 52 lines in the Java version, but the length of rewritten code in Processing 4 was 42 lines. The time required for porting from Java to Processing was about 30 minutes.

3.2.2. "Electric circuit" example. This example is to learn the relationship between current, voltage, and resistance. When the appropriate battery and resistor are selected and the switch is turned on, the light bulb lights up; if the current is too high, the bulb breaks and makes a noise; if the current is too low, the bulb does not light up (see Figure 5). The length of the source code was 613 lines in the Java version, but the rewritten code in Processing was 466 lines.

 | DOI: 10.20533/ijicr.2042.4655.2023.0142

Figure 5. Electrical Circuit Example

The time required for porting from Java to Processing was about 2 hours.

4. Comparison in Physical Simulation

4.1. "Fisica", a physics library for Processing 4 [4]

"Fisica" physics library for processing is wrapper for JBox2D [5], a 2D physics engine. It is available on Processing as one of contributed libraries. "Fisica" enables us to create not only realistic simulation model but ideal apparatus model without friction and air resistance, etc. The primary parts are shown in Table 2. The "FCompound" can combine plural parts into an object to create complex shape objects. Additionally, "FJoint" makes rotatable joint among plural objects.

Table 2. Primary parts of physics library "Fisica" for Processing 4

Class Name	Shape
FCircle	Circle
FBox	Rectangular
FLine	Line
FPoly	Polygon
FBlob	Elastic polygons
FCompound	Combination of multiple forms

The procedure for using "Fisica" is very simple. Firstly, we initialize "Fisica" settings and create an instance of "FWorld" which represents the virtual world in the setup method. Secondly, we set the presence or absence of walls on the four sides of the window. Then we create the objects to appear and set their physical properties and visual properties (color, image, transparency, etc.). Finally, we add them to the virtual world. The physical state (position, velocity, acceleration, etc.) can be obtained from each object during execution.

4.2. "Cannon" in "Hooked on Java"

To evaluate the functionality and performance of "Fisica", I use a simulation program for Cannonball ballistics that appears in the first book on Java, "Hooked on Java" (see Figure 6). This program allows the user to adjust the firing angle, initial velocity, gravity value, and air resistance using GUI. The physical calculations are performed sequentially using formulas derived from the equations of motion. For comparison, the original Java version of the program was rewritten in Processing (see Figure 6), and a version using the "Fisica" for Processing (see Figure 7) was created. The time required to create the programs and the differences in functionality among them are summarized in Table 3.

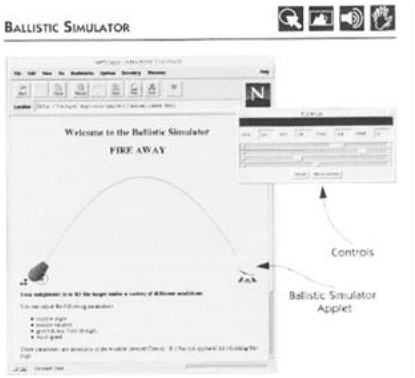

Figure 6. "Cannon" example in Java

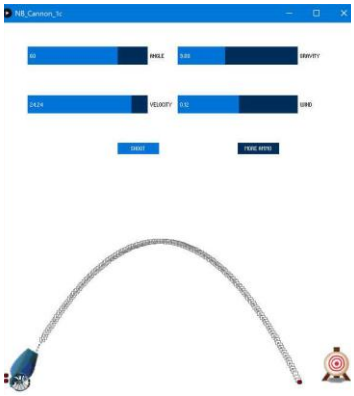

Figure 7. "Cannon" in Processing 4 (no use of "Fisica" library)

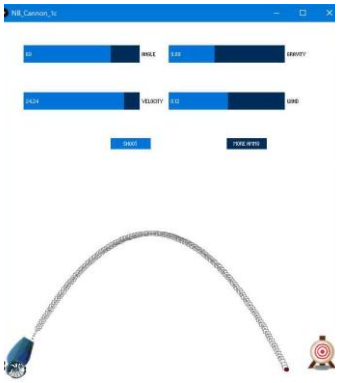

Figure 8. "Cannon" in Processing 4 using "Fisica" library

Table 3. Comparison on program development task

	Java	Processing	Processing + Fisica
Decipherment time	14 hours	-	-
Working time	-	40 hours	32 hours
Number of program lines (excluding comments)	341 lines	306 lines	233 lines
Number of Windows	2	1	1
Ease of expansion	△	△	○

※Comparison based on approximately 10 hours of prior study of Processing

In the original Java version program, the parameter settings are changed in a separate window, while Processing programs use external GUI library “ControlP5” for slide-bar, check-box, etc. Since there is no difference in the physics of these three models, there is no difference in the ballistics results, and the animation of the image and the playback of the sound are the same when the target is hit. To evaluate the amount of work involved in rewriting a Java program into Processing and using "Fisica," I first selected students with intermediate Java language skills. After teaching the students how to use the Processing language, I explained the structure of the original Java version of the "Cannon" program. Next, I asked the students to rewrite the Java program into Processing and create a version using "Fisica." Then I evaluated the degree of difficulty of the work. The results are shown in Table 3. The results show that the time required to create a simulation model with "Fisica" was 80% of the time required in program without using "Fisica." Also, the size of source code was 76% of the case without using "Fisica."

4.3 Utilization of “Fisica” to education

The usefulness of "Fisica" for physical simulations was confirmed in section 4.2. In this section, we describe a concrete example of creating a learning model.

4.3.1. Weighting Scale Simulation. Figure 9 shows a simple balance created using "Fisica". The weights in

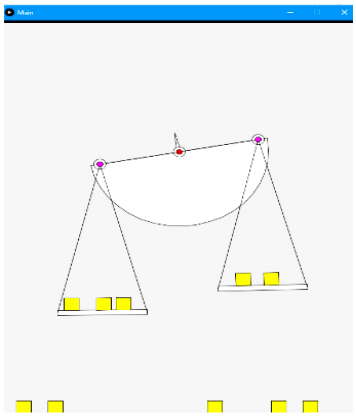

Figure 9. Weighting Scale model

yellow are set to make contact with the left and right plates, but not with the strings suspending the plates or the semi-circular part in the center, so that the learner will not be disturbed when placing the weights on the plates. The semi-circular parts have weights and are designed to be balanced by the difference in weight be-tween the left and right sides in proportion to the angle of inclination. This model can be run on a tablet, smart phone, or personal computer, and we believe it can be used in online learning and in home study preparation and review. If you wish to create an upper-dish balance, you can easily do so by setting the dish and support rod to non-rotatable.

4.3.2. Curling Simulation. This example is a simulation of a curling competition. The subject is a reproduction of the third-place game (Japan vs. Great Britain) in the women's curling competition at the 2018 Winter Olympics in Pyeongchang. The final throw by Great Britain resulted in Japan scoring and winning (see Figure 10 and Figure 11). In the curling competition, which is also called "chess on ice," I thought that it could be used as a teaching material for strategy and tactics, not only for competitors but also for enthusiasts to consider. I believe that it will be useful for groups of people to enjoy learning about other tactics options.

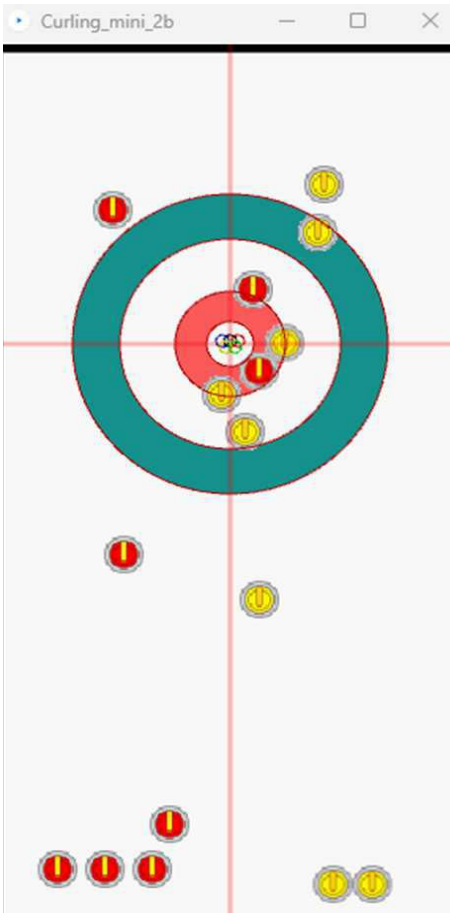

Figure 10. Curling model using Processing 4 (before the last throw)

 | DOI: 10.20533/ijicr.2042.4655.2023.0142

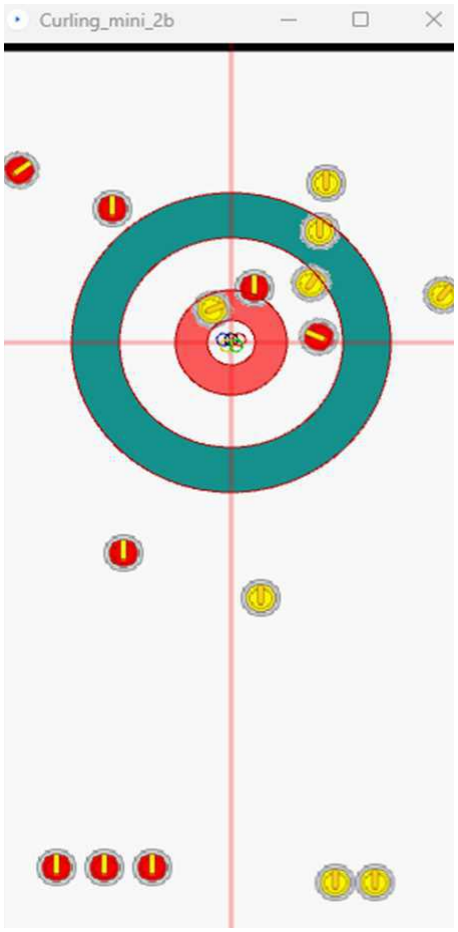

Figure 11. Curling stone allocation after the last throw for figure 10

5. Variational Problem Simulation

This example is the Brachistochrone problem, which is one of the variational problems that cannot solved without using variation method [6]. We will compare the results of a theoretical solution, a numerical solution using the piecewise function approximation, and a solution using a simple model utilizing "Fisica".

5.1. About Brachistochrone problem

The problem, presented by Johann Bernoulli in 1696, is to find "a curve connecting A and B such that, given two points A and B not on the same vertical line, a single mass point slips from A to B in the shortest time" (Figure 12).

$$T=\int_{xa}^{xb}\sqrt{\frac{1+y'^2}{2gy}}dx \tag{1}$$

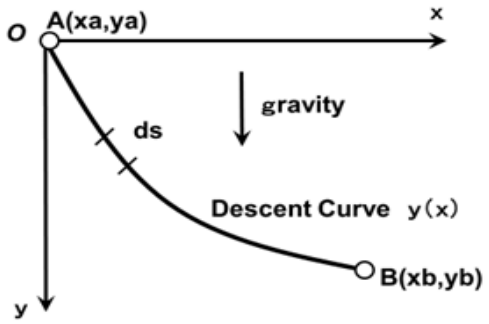

Figure 12. Brachistochrone problem

The time T is expressed in the form of Equation (1), and the problem is to find a function y(x) such that T is minimized.

5.2. Theoretical solution

The theoretical solution of the function y(x) such that the value of equation (1) is minimized is a cycloid curve, as derived by Johann Bernoulli, Jacob Bernoulli, Newton, de l'Hospital and others (equations (2) and (3), Figure 13).

$$x=\frac{C}{2}(t-\sin\ t) \tag{2}$$

$$y=\frac{C}{2}(1-\cos\ t) \tag{3}$$

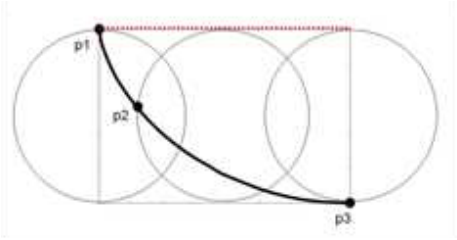

Figure 13. Theoretical solution curve (cycloid) for Brachistochrone problem

5.3. Numerical solution by Java using piecewise function approximation model

In this section, we present a method for representing an unknown curve in a piecewise function model [7], where the x-axis direction is equally divided into n parts and only the corresponding y-axis values are variables (Figure 14). This approach allows the variational problem to be transformed into an extreme value problem for plural variable functions. The method based on piecewise function approximation also allows visualization of

 | DOI: 10.20533/ijicr.2042.4655.2023.0142

the optimization process (Figure 15). Figure 16 shows a graph comparing the approximate solution by piecewise functions with the theoretical solution by the variational method: the theoretical value of the shortest time required is 0.709251529 [s] when the coordinates of point A are (0,0) and those of point B are (π/2, 1.0), and the time by piecewise function approximation is 0.710748133 [s]. The error of the approximate solution with respect to the theoretical solution is +0.21%.

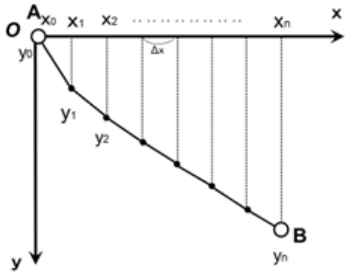

Figure 14. Piecewise function approximation model

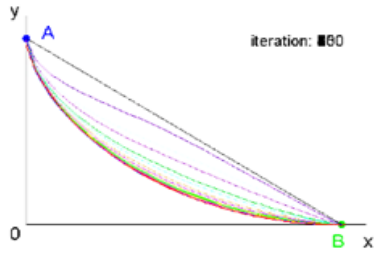

Figure 15. Solution curve transition

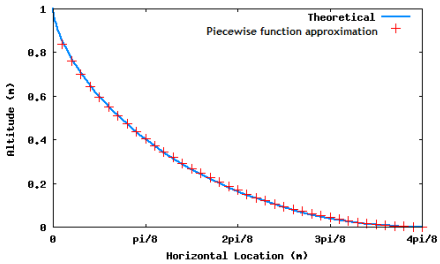

Figure 16. Result comparison between theoretical and piecewise function approximation

5.4. Processing solution using “Fisica” library

Finally, I present an application of the physics calculation library "Fisica" for Processing 4. As shown in Figure 17, 18 and 19, I created a model that approximates a straight line, a convex curve, and a cycloid curve with a polygonal line. In each case, a small quality point was prepared, the fixation was released by a key operation to start the fall, and the passage time of the specified interval was measured by the program. Note that in some cases, contact with the slope could not be detected at the time of iteration calculation if the fold line was set too finely, so a coarse fold line was used for this approximation.

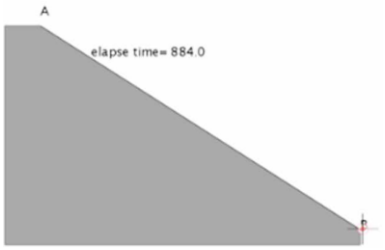

Figure 17. Fisica, Linear slope model

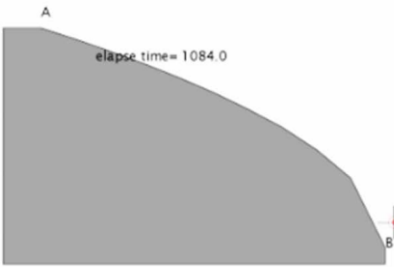

Figure 18. Fisica, Convex curve model

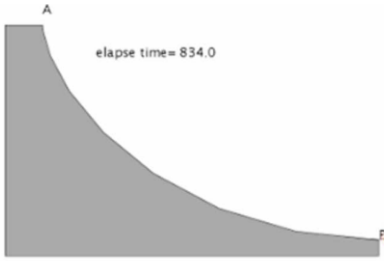

Figure 19. Fisica, Cycloid model

Method and Slope Curve Type	Elapsed time (s)
Theoretical, Linear Slope	0.840
Theoretical, Cycloid	0.709
Piecewise Function Approximation, Close to Cycloid	0.711
Fisica, Linear Slope	0.884
Fisica, Convex Curve (Parabola)	1.084
Fisica, Cycloid	0.834

Table 4. Elapsed-time comparison

6. Constructivist Approach to Animal Behavior

Finally, the "Sheepdog Project" is discussed as a practical project assignment for a constructivist approach to animal behavior. To motivate students with advanced programming skills, I have developed an animal behavior simulator [8] [9]. In this paper, I introduce the simulator, named "Sheep farming," and present an example of its implementation as a project assignment. The simulator reproduces the behavior of

 | DOI: 10.20533/ijicr.2042.4655.2023.0142

a flock of sheep according to the Boid theory proposed by Craig Reynolds [10] [11]. Through the assignment, students are to programmatically represent the behavior patterns of the dog in the simulator by means of a Java program, and to guide the flock of sheep and drive them into the sheep pen in a reliable and short time. Boid theory has been used to automatically generate realistic behaviors of many virtual animals in movies (e.g., the movements of a flock of bats in the movie "Batman Returns" [12]).

6.1. Overview of sheep farming simulation platform

Figure 20 shows the startup screen of a sheep farming simulator created in Java. The main field is a pasture with sheep and dogs running around. The top panel has a run button, a speed control slide, and input/output file names. the amount of Java code is about 2600 steps. The class structure of the simulator is shown in Figure 21. The main class is "Meadow," which manages the virtual world; instances of "Meadow" ask all sheep and dogs to perform the following behaviors, subject to predefined limits Sheep herding behavior is implemented in the class "Sheep" so that instances of "Sheep" tend to flock and avoid the vicinity of dogs.

The student modifies the dog's behavior by way of intelligence implemented in the class "Dog." However, all the students can do after the start of the program is to monitor the progress.

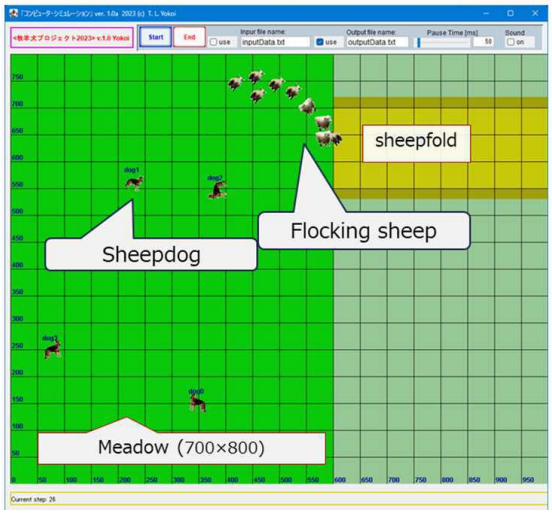

Figure 20. Platform for "Sheep Farming" Project

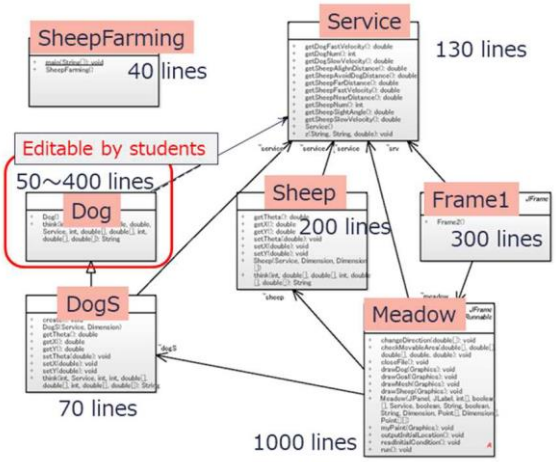

Figure 21. Class network for "Sheep Farming" platform

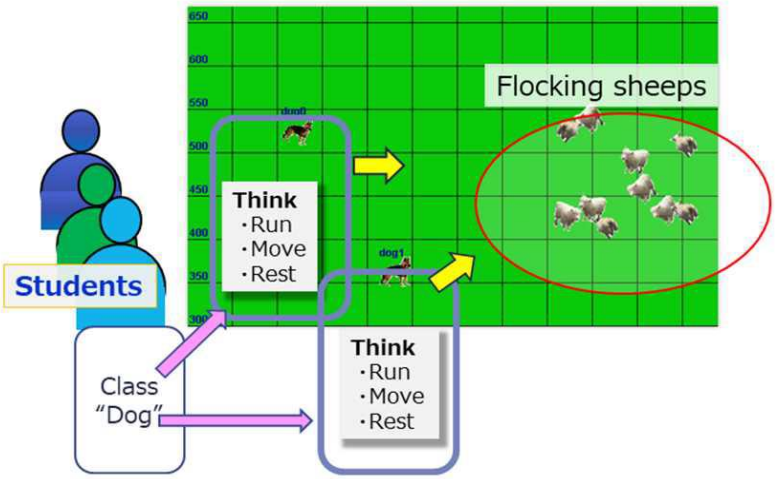

Figure 22. "Think" design of sheepdog by students

 | DOI: 10.20533/ijicr.2042.4655.2023.0142

6.2. The Evaluation value for good behavior

The goal of the project task is to drive all sheep grazing on the grassland into the sheds in a short time, with as few sheepdogs as possible (no more than 5), and in a reliable manner. Therefore, it is required to design "dog behavior" such that the following indices are minimized. Note that the success rate is calculated from the results of at least 20 consecutive runs, starting with a random arrangement.

Evaluation value = (Number of dogs) x (Maximum steps required for successful cornering) ÷ (Success rate)

When a dog cannot easily be driven in a certain step, a mechanism to detect this and start over should be incorporated.

6.3. Result of Students' Struggle

The simulator has been running for 18 years, and in each year the shape and location of the sheep pen has been changed, so the students have worked with a variety of approaches that you would never guess at the final presentation. Many examples were based on tactics to evaluate the size of the sheep herd and the location of the center, such as setting up a dog in the role of slowly moving toward the sheep herd without disrupting it (see Figure 23). The resultant evaluation value distribution for all participants is summarized in Figure 24.

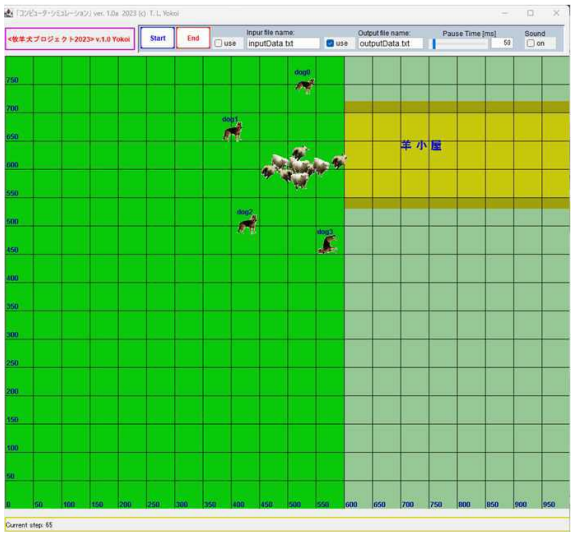

Figure 23. Successful collaborative behavior strategy example

The feedback after all the lectures confirmed that most of the students enjoyed this simulator and learned both object-oriented methodology and simulation techniques, even though they struggled to solve problems.

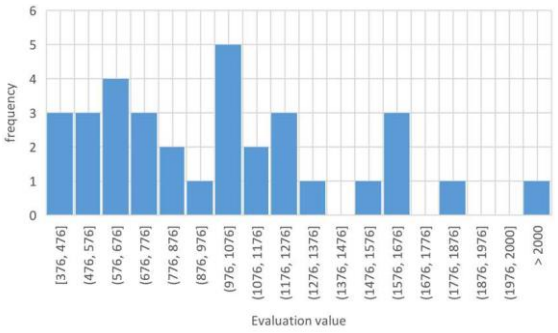

Figure 24. Evaluation value frequency distribution of the "Sheep Farming" project

7. Conclusion

In this paper, I described various original simulation programs developed for teaching in the course "Computer Simulation." From the comparison on the development effort, amount of code, and development time, the result revealed that the ability of the Processing exceeded all expectations. In the physical simulation modelling, the "Fisica" library for Processing enabled us to create complex physical models in a small number of program lines with easy-to-understand codes. The "Brachistochrone problem" was discussed as one of the most famous variational problems. As a result, the numerical solution using piecewise function showed high accuracy that is almost consistent with the theoretical solution, but it is more attractive as a method that can take realistic phenomena such as air resistance into account. By designing the thinking of a sheepdog to drive a flock of sheep into a sheep pen in a reliable and short time, I was able to confirm the students' interest and engagement with the task.

8. References

[1] Websrv.TCU. (2023). Syllabus of Computer Simulation. https://websrv.tcu.ac.jp/tcu_web_v3/slbssbdr.do?value(risyunen)=2023&value(semekikn)=1&value(kougicd)=yab723201&value(crclumcd)=y217002 (Access Date: 5 September 2023).

[2] Processing. (n.d.). https://processing.org/ (Access Date: 28 August 2022).

[3] Van Hoff, A., Shaio, S., and Starbuck, O. (1995). Hooked on Java - Creating Hot Web Sites with Java Applets, Addison-Wesley.

[4] Fisica. (n.d.). http://www.ricardmarxer.com/fisica/ (Acc-ess Date: 28 August 2022).

[5] JBox2D. (n.d.). http://www.jbox2d.org/ (Access Date: 5 September 2023).

[6] Brachistochrone problem. (n.d.). https://mathworld.wolfram.com/BrachistochroneProblem.html, (Access Date: 28

 | DOI: 10.20533/ijicr.2042.4655.2023.0142

August 2022).

[7] Yokoi, T. (2010), Study on the Brachistochrone problem and its extended problem by use of computer algebra and numerical calculation, The 29th Annual Conference of the Japan Society for Simulation Engineering.

[8] Yokoi, T. (2003). Construction of Educational Environment for Ecological Simulation utilizing Java, Journal of Information Media Center, Faculty of Environmental and Information Studies, Musashi Institute of Technology, Vol.4, pp.62-66.

[9] Yokoi, T. (2015). Development of a Computer Simulation Platform for Learning Constructive Approach of Animal Behaviors, the 34th JSST Annual Conference (JSST2015).

[10] Reynolds, C. (n.d.). Simulated Boid flock avoiding cylindrical obstacles, http://www.red3d.com/cwr/ (Access Date: 5 September 2023).

[11] Davison, A. (2005). Killer Game Programming in Java. O'Reilly. pp.592-613, Flocking Boids.

[12] Batman Returns. (n.d.). https://www.youtube.com/watch?v=eIo_S0aHyfI (Access Date: 5 September 2023).

 | DOI: 10.20533/ijicr.2042.4655.2023.0142