Particle Simulator
STDISCM Problem Set 1 Specifications
Term 2 AY 2023 - 2024

Ryan Austin Fernandez
2401 Taft Avenue
Manila, Philippines
ryan.fernandez@dlsu.edu.ph

ABSTRACT
This document details the specifications of the first problem set in distributed computing. General instructions are provided for a particle physics simulator, which will be implemented using concurrent programming.

Author Keywords
concurrent programming, load balancing

CCS Concepts
•Computing methodologies → Shared memory algorithms; Concurrent algorithms;

GENERAL PROGRAMMING INSTRUCTIONS
You are to create a particle physics simulator using either C++ or Java. A user should be able to add particles to the canvas. A user can also add walls, which the particles will bounce off of. When rendering the canvas, ensure the frame rate does not drop while updating the environment.

RESTRICTIONS
Only either C++ or Java and their threading support library and UI libraries should be used in this problem set.

Other libraries or game engines should be used. You may request approval in our Discord server (a thread will be created for this purpose).

SPECIFIC INSTRUCTIONS
A user should be able to add particles to the environment with an initial position (x, y), an initial angle Θ (0 degrees is east, and degrees increase in an anticlockwise manner, e.g., 90 degrees is north), and a velocity V (in pixel per second). The particle will travel in a straight line, bouncing off the four walls of the canvas. Particles do not collide with other particles. All collisions are elastic, which means particles do not slow down or speed up after a collision.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

The canvas should be $1280x720$ pixels. Coordinate $(0,0)$ is the southwest corner of the canvas. Coordinate $(1280, 720)$ is the northeast corner.

Particles can be added in batches. This is in three forms:

- Provide an integer n indicating the number of particles to add. Keep the velocity and angle constant. Provide a start point and end point. Particles are added with a uniform distance between the given start and end points.
- Provide an integer n indicating the number of particles to add. Keep the start point and velocity constant. Provide a start Θ and end Θ. Particles are added with uniform distance between the given start Θ and end Θ.
- Provide an integer n indicating the number of particles to add. Keep the start point and angle constant. Provide a start velocity and end velocity. Particles are added with a uniform difference between the given start and end velocities.

Additionally, a user can also add walls, given two endpoints (x_1, y_1) and (x_2, y_2), which the particles will also bounce off of.

Ensure that your screen resolution is high enough to show all particles on-screen. Show the FPS counter on-screen every 0.5 seconds.

You are also to write a **1 - 2 page technical report** containing the following:

- Discuss how you implemented the load balancing of the particle physics computations.
- Explain how you computed the frame rate.
- Discuss how you implemented the load balancing of the rendering. Explain how you ensured a consistent frame rate. Discuss additional implementations you created to ensure that a constant FPS is met.
- Provide code snippets supporting your technical explanation.

The technical report must have one-inch margins and be typeset in Calibri 11-point font. **DO NOT** exceed two pages. Any pages after the second page will be ignored. At the end of the technical report, provide a record of contribution for each major activity for the case study.

1. Each row represents an activity, and each cell represents the percentage each member contributed to that activity. Each row's contributions must total 100.
2. The **TOTAL** row must be normalized to add up to 100.

Example:

Group Members:

- P1: Percy Jackson
- P2: Annabeth Chase
- P3: Grover Underwood

Activity	**P1**	**P2**	**P3**
Topic Formulation	50.0	25.0	25.0
Machine Definition	33.3	33.3	33.3
Formal Language and Computational Power	25.0	50.0	25.0
Applications	10.0	10.0	80.0
Raw Total	118.3	118.3	163.3
TOTAL	**29.6**	**29.6**	**40.8**

Have each member sign below the table above their full name and the date.

GRADING RUBRICS

Table 1 shows the grading rubrics for problem set 1.

FINAL DELIVERABLES AND DEADLINE

- A 1 - 5 min video demo showing the sample program in MP4. This video will be posted in class for the voting part.
- Technical report
- Submit a GitHub/GDrive link containing your source code. Include a README.md file for instructions on how to run your program.
- For C++, the program should be compatible with VS 2019 (x64) and C++ 20 compiler and ready for compilation. The program should already include any external dependencies.
- For Java, include a JAR file to be run.

Deadline for submission of all deliverables: February 16, 2024, 11:59 PM

REFERENCES

[1] Allen B Downey. 2005. *The little book of semaphores*.

[2] Maurice Herlihy, Nir Shavit, Victor Luchangco, and Michael Spear. 2020. *The art of multiprocessor programming*. Newnes.

[3] Maarten Van Steen and Andrew S Tanenbaum. 2017. *Distributed systems*. Maarten van Steen Leiden, The Netherlands.

Table 1. Rubrics for Grading Problem Set 1

Section	Excellent	Good	Developing
Frame Rate Consistency (20)	The frame rate is consistently at least 60 FPS with occasional drops to $\sim$50 FPS. The latency is barely noticeable. (20)	The frame rate is consistently at least 60 FPS with occasional drops to $\sim$30 FPS. The latency is noticeable but happens occasionally and not too long. (13)	The frame rate is consistently jittering between 20 FPS – 60FPS. The application cannot maintain a steady frame rate. (7)
Completeness of Functionality (20)	All features are implemented (20)	One to two features are missing (13)	Three or more features are missing. (7)
Presentation and Creativity (6)	The software looks polished and aesthetically pleasing. (6)	The software looks rushed. (4)	The software barely has a functioning and understandable user interface. (2)
Ease of Use (4)	The software is very easy to use to add particles/walls. (4)	Some features are difficult to use. (3)	The software is difficult to use. Hardcoding test cases is easier than using the user interface. (1)
Correctness of Load Balancing of Computation (20)	The load balancing of particle physics is explained clearly and correctly. (20)	There are some points of improvement in the load balancing of particle physics, or there are some aspects that are not explained clearly. (13)	There are plenty of points of improvement in the load balancing of particle physics, or there are several aspects that are not explained clearly. (7)
Correctness of Load Balancing of Rendering (20)	The load balancing of the rendering is explained clearly and correctly. (20)	There are some points of improvement in the load balancing of the rendering, or there are some aspects that are not explained clearly. (13)	There are plenty of points of improvement in the load balancing of the rendering, or there are several aspects that are not explained clearly. (7)
Clarity of Writing (6)	All solutions are presented clearly in the document. (6)	Some solutions are hard to understand how they were implemented in the document. (4)	Most solutions are hard to understand how they were implemented in the document. The document is difficult to read in general. (2)
Grammar (4)	There are 0 - 2 grammatical errors in the document. (4)	There are 3 - 4 grammatical errors in the document. (3)	There are >4 grammatical errors in the document. (1)
TOTAL (100)			