Algorithm Visualization Simulator:
A Simulation of Complex Algorithm Behaviors

Jack Brunswik

1/30/2026

1 Project Overview

1.1 Domain

Computer Science – Modeling and Simulation of Algorithmic Systems (Algorithm Analysis & Visualization)

1.2 Problem Statement

Algorithms are often taught and analyzed using static complexity measures (e.g., Big-O notation), which do not fully capture their dynamic behavior during execution. This simulation project aims to answer the following questions:

- How do complex algorithms evolve over time as they process data?
- How do different input characteristics affect algorithm performance and internal state changes?
- How can simulation-based visualization improve understanding of algorithm efficiency, bottlenecks, and emergent behavior?

The simulation allows users to observe algorithm execution step-by-step, focusing on state transitions, resource usage, and performance metrics rather than solely on final outputs.

1.3 Scope

Included

- Simulation of selected complex algorithms (e.g., sorting, graph traversal, or optimization algorithms)

- Visualization of algorithm states over simulated time
- Measurement of execution-related metrics (comparisons, swaps, memory usage, time steps)

Excluded

- Low-level hardware simulation (CPU pipelines, cache coherence)
- Real-time system benchmarking
- Full compiler or operating-system level modeling

2 System Description

2.1 System Components

- **Algorithm Entity**: Represents the algorithm being simulated (type, logic, parameters)
- **Data Structure Entity**: Input data (arrays, graphs, trees) with configurable size and distribution
- **Simulation Engine**: Controls time progression and event scheduling
- **Visualization Module**: Renders algorithm states and transitions
- **Metrics Collector**: Records performance and behavior metrics

2.2 System Dynamics

The system operates as a time-stepped simulation. At each simulation step:

- The algorithm processes part of the data structure
- State changes are recorded (e.g., swaps, node visits)
- Metrics are updated
- The visualization module renders the current state

Interactions occur between the algorithm entity and the data structure, mediated by the simulation engine, which ensures consistent progression and repeatability.

2.3 Core Models and Algorithms

The simulation incorporates at least three core models:

Algorithm State Transition Model

- Represents algorithm execution as a finite state machine (FSM)
- States include initialization, processing, comparison, update, and termination

Computational Complexity Model

- Tracks theoretical versus observed time complexity
- Uses operation counts (comparisons, swaps, node visits) as discrete cost functions

Input Data Distribution Model

- Models input randomness using probability distributions (uniform, sorted, reverse-sorted, random graph density)
- Enables controlled experimentation on algorithm behavior

Example algorithms to be simulated include:

- Quicksort or Merge Sort
- Breadth-First Search (BFS) or Dijkstra's Algorithm
- Greedy or dynamic programming-based algorithms

2.4 Assumptions

- Visualization time does not affect simulation time
- Input data fits entirely in memory
- Randomized algorithms use pseudo-random generators with fixed seeds

3 Implementation Approach

3.1 Programming Language

Python will be used due to its rapid development cycle, strong support for simulation and visualization libraries, and readability suitable for analytical purposes.

3.2 Development Environment

Libraries:

- NumPy (data modeling)
- Matplotlib / Pygame / Tkinter (visualization)
- NetworkX (graph-based algorithms)

IDE: PyCharm

3.3 Simulation Type

Discrete-event simulation. Algorithm execution is modeled as a sequence of discrete events (comparisons, swaps, node visits), enabling precise tracking of state changes over time.

3.4 Data Collection Plan

The simulation collects and analyzes the following metrics:

- Number of operations per time step
- Total comparisons and swaps
- Memory access count
- Algorithm completion time (in simulation steps)
- Performance comparison across input sizes and distributions

Collected data is used to generate plots and comparative visual reports evaluating algorithm behavior under varying conditions.

4 Literature Review

4.1 Core Models and Algorithms

This project builds upon established research in algorithm analysis, simulation modeling, and algorithm visualization.

Source 1: Cormen et al. – *Introduction to Algorithms*

Model/Algorithm: Classical algorithm analysis and complexity modeling using operation counts and asymptotic analysis.
Application: Supports the computational complexity model used in the simulation.
Adaptation: Operation counts are recorded dynamically over simulated time.

Source 2: Knuth – *The Art of Computer Programming, Volume 1*

Model/Algorithm: Stepwise execution analysis with precise operation counting.
Application: Motivates tracking internal algorithm states.
Adaptation: Low-level operations are abstracted into uniform-cost events.

Source 3: Law & Kelton – *Simulation Modeling and Analysis*

Model/Algorithm: Discrete-event simulation and probabilistic modeling.
Application: Forms the basis of the simulation framework.
Adaptation: DES concepts are applied to algorithm execution.

Source 4: Hopcroft et al. – *Introduction to Automata Theory, Languages, and Computation*

Model/Algorithm: Finite state machines.
Application: Models algorithm execution stages.
Adaptation: FSM states are extended with metrics and visualization hooks.

Source 5: Diehl – *Software Visualization*

Model/Algorithm: Visualization of dynamic software behavior.
Application: Guides visualization design.
Adaptation: Techniques are simplified for clarity and educational focus.

4.2 Related Work

Existing tools influencing this project include:

- Algorithm visualization platforms such as VisuAlgo
- Sorting algorithm animations
- Graph algorithm simulators for BFS, DFS, and Dijkstra's algorithm

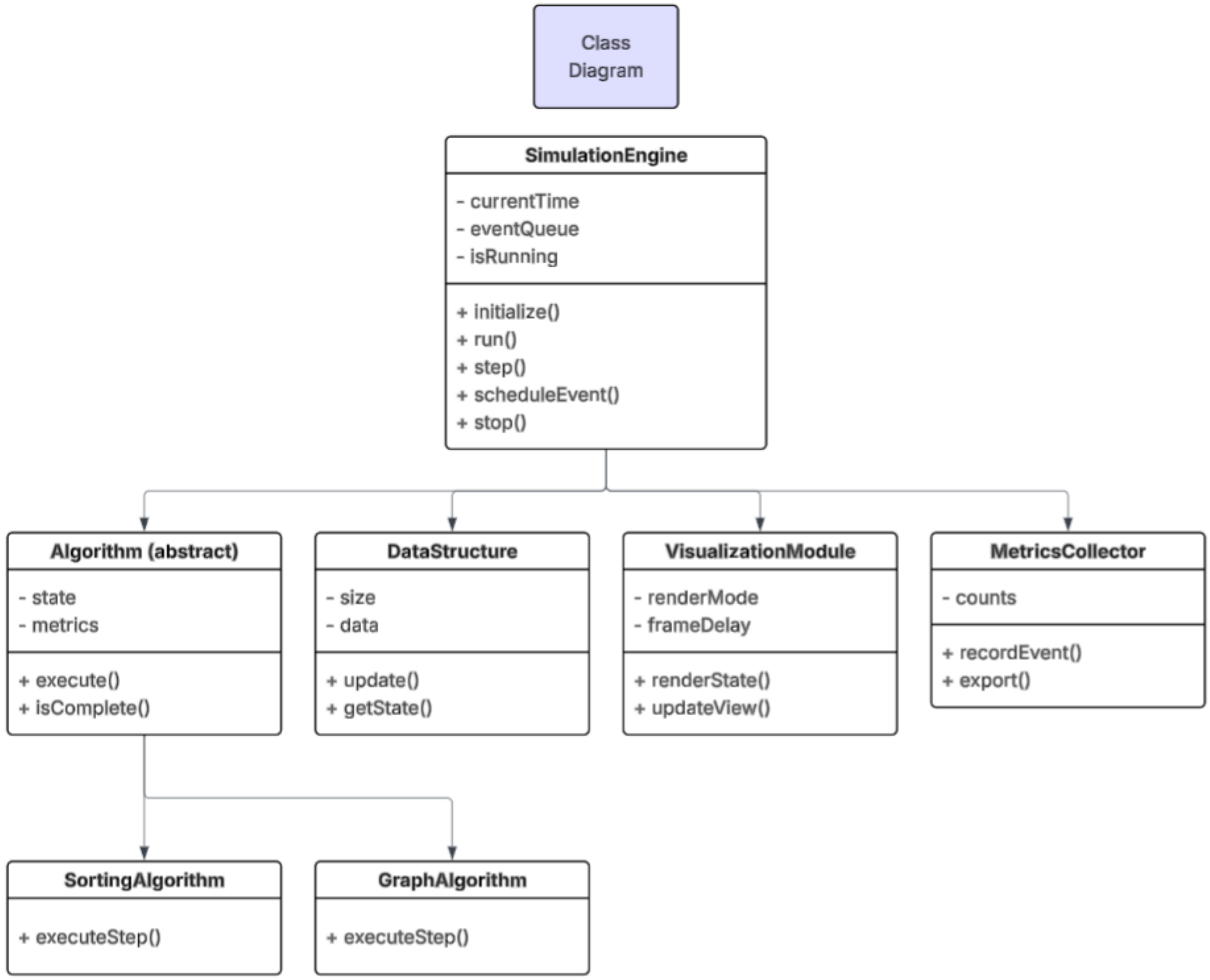

Figure 1: UML Class Diagram of the Algorithm Visualization Simulator

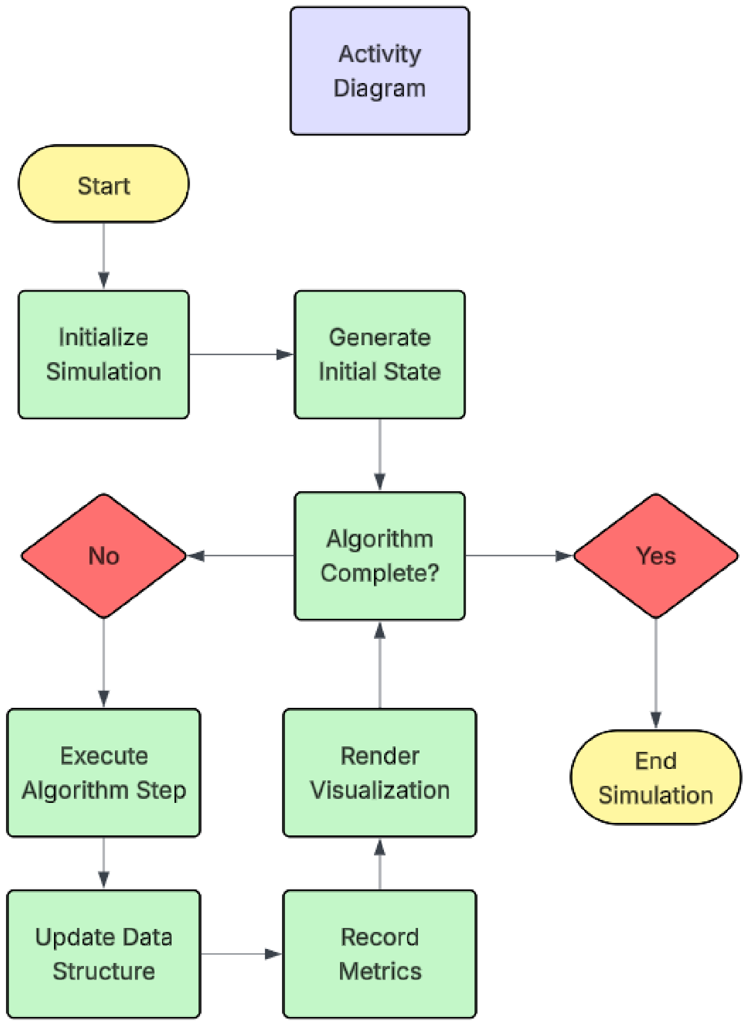

Figure 2: UML Activity Diagram of the Algorithm Visualization Simulator