Pandemic Simulation Engine Proposal

Features and Architecture

Features

Overview

The Pandemic Simulation Engine will be a cross-platform code library designed to take scientific data models and generate game state for simulation based games. The scientific models will be the primary input to the simulation engine, and will be used to drive state changes at each simulation update. The results of the simulation will be recorded on a global blackboard that can be queried by a game running the simulation. The long-term objective of this engine is to be extensible and scalable enough that it can be easily expanded to run multi-faceted simulations based on whatever scientific models are provided. The engine will be designed as a cross-platform C++ library that will be data driven and compatible with the infrastructure of a typical game.

Cross Platform C++ Library

To maximize compatibility with any future game projects that will utilize this engine, the simulation engine will compile down to a platform-agnostic code library. It will be a headless tool that can be used for game's, scientific modeling, or training.

Model Driven Simulation

This simulation engine will be driven first and foremost by scientific data models and their accompanying inputs. To import these models into the engine, the user can create data files to define SimModels and SimGraphs within the simulation engine. When a user defines a SimModel object, they are defining simulation work to be done within the engine given the simulation's global state, the current states of agents, and graph structures. In essence, these objects will specify how the simulation updates itself, defining the state variables that will be stored in simulation agents, how graphs are generated at startup and potentially restructured during runtime, how agent variables are updated, the state variables that will be output to the global blackboard, and how those global variables will be aggregated. In this way, the simulation can be driven by pre-validated scientific models.

Agent-Based Simulation with Graph Structures

Complex model based simulations will be made possible by utilizing an agent-based simulation approach that uses graphs to represent complex real-world relationships and networks. Like SimModel objects, SimGraph definitions will be specified with data files and each represent a single SimGraph object within the simulation. SimGraph Definitions will specify which SimModels will be attached to the graph, and the dependency order of the SimModels, so that requisite computation work can be performed and outputted to the global blackboard before running a SimModel that requires it. The SimModels attached to a given graph will determine what state the SimAgents of that SimGraph will store.

Scaled and Extendable Simulation

By allowing the specification of models in SimModel and SimGraph definition files, and letting users load any number of SimGraphs with any number of SimModels attached to them into the simulation, the simulation engine lends itself to being extended and scaled to represent any number of desired scientific models in its simulations.

Simulation Management

There may be cases where it is desirable to manage the simulation state so that very specific scenarios may be simulated by the engine. In that case, the engine will provide methods to control and edit global blackboard data. Since global blackboard data is used to drive a simulation's update, editing global blackboard data is an easy way to initially contrive scenarios to see how they play out. The global blackboard will be implemented, stored, and passed between engine and client in a JSON format.

Queryable Simulation State

After each simulation update, when the simulation's global state has been updated to reflect the state of its SimGraphs, games can query the simulation state to observe how key variables have changed over the course of the simulation. These queries will allow the user to ask for any simulated variable by name. The SimModel data files will specify exactly which variables are simulated and what those variables are called. By querying these variable names, users have full access to any data that was simulated by the engine.

Technical Design

Components:

SimModels

All input models must eventually be represented as SimModels Definitions within the simulation engine. SimModels will hold a list of SimCommand objects, which will specify exactly how computational work is performed on SimAgents in a SimGraph. SimModels may include the following information:

- SimModel String Identifier
- Define a Initialization Command: SimInitializeCommand
    - What state variables will be appended onto SimAgents within any SimGraph this SimModel is attached to, and their corresponding defaults.
    - How to generate the graph (How many agents, how to connect edges)
- Define a Update Command: SimComputeCommand:
    - How to compute and update each SimAgent's appended variables.
        - Specifies all required computation inputs, such as the name of a required blackboard variable or SimAgent variable, or the value of a SimAgent's edges in a SimGraph.
        - Specifies the methodology for computing and updating the SimAgent's appended state variables.
- Define a Update Command: SimAggregateCommand
    - What variables will be aggregated and appended onto the global blackboard.
    - Specify how these global variables are calculated using the aggregated state of SimAgent variables within an owning SimGraph.
- Define a Update Command: SimRegenerateCommand
    - How to regenerate a graph's edge network given current graph and blackboard state

SimGraph Definitions

Like SimModels, SimGraphs in the simulation are specified with definition files, with each SimGraph data file corresponding to a single graph that will be generated by the simulation. SimGraph definitions are concerned with encoding the order of SimModels that will be used to update them, as well as listing the ID's of graph dependencies. SimGraph Definitions may include the following information:

- SimGraph String Identifier
- Which SimModels are attached to this SimGraph
    - Specify an dependency order of SimModel work to be performed, if one SimModel requires that another finish its graph agent updates and global blackboard update before another.
- The ID of SimGraphs that this SimGraph depends on during an Update.

SimGraphs and SimAgents

SimGraphs encode structures within which the SimAgent state will be simulated, as well as the SimModels that will update and define the graph. The meaning of SimGraphs nodes, edges, and their respective states will all be contextually defined. SimAgents are the nodes of a SimGraph, and can represent anything from people, to hospitals, to electrical grid transformers, or even entire populations. SimGraph structures can be as simple as a single node if the SimModel does not require a graph structure to simulate its state. SimAgents will have a unique state and use whatever SimModels are attached to the SimGraph to update themselves. This update can use any available information that is required to make these updates, including existing agent state, graph structure, edge values, and global blackboard data. All SimGraphs in the simulation will be stored in a list on the simulation's global state, and will be updated at each update in the simulation. Importantly, SimGraphs will store the dependencies of any SimModels it owns, and ensure that SimModels do their computations in the proper order in cases where one SimModel relies on the outputs of another.

Global Blackboard

The Global Blackboard is a part of the simulation engine's global state, and is the primary location for storing and querying the overall state of the simulation. The Global Blackboard serves as the medium of communication between the simulation and game client. It also allows for data to be shared across different SimModels and SimGraphs. Global blackboard state variables will be defined and appended by all the SimModels across all SimGraphs in the simulation, and these variables will serve as the inputs and outputs of various SimModel computations during simulation update. The global blackboard will be stored as a JSON object.

Simulation Update

The simulation update step occurs as follows:

- SimGraphs on the SimEngine are looped through in dependency order and updated.
- When a SimGraph is Updated, the SimGraph loops through every SimModel in its SimGraph Definition. Then, it loops through every SimCommand on a given SimModel, passing itself into each SimCommand for updating.
- Each SimCommand will update the SimGraph, SimAgents, or Global Blackboard according to its type and state data generated from the SimModel at initialization time.

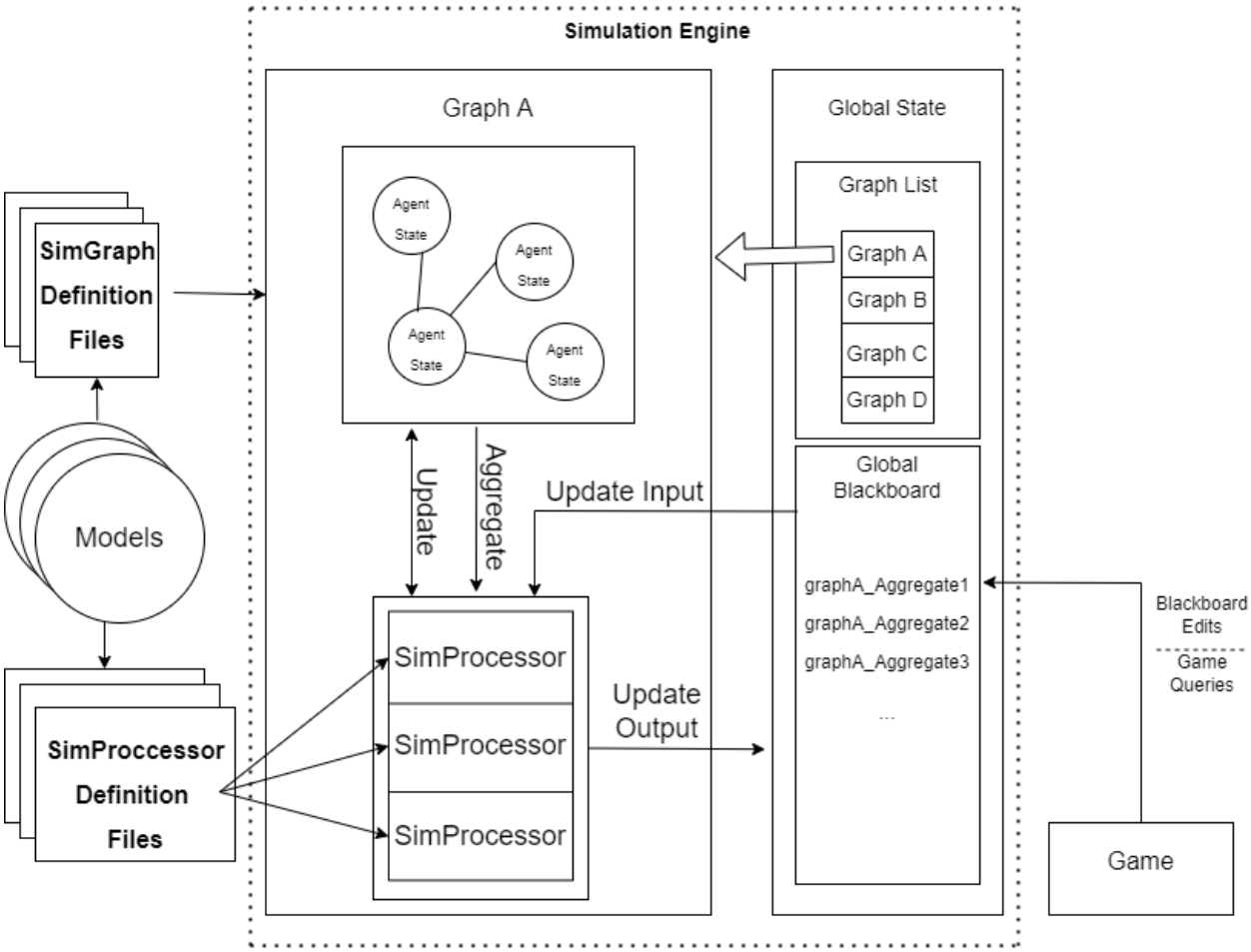
SimCommands

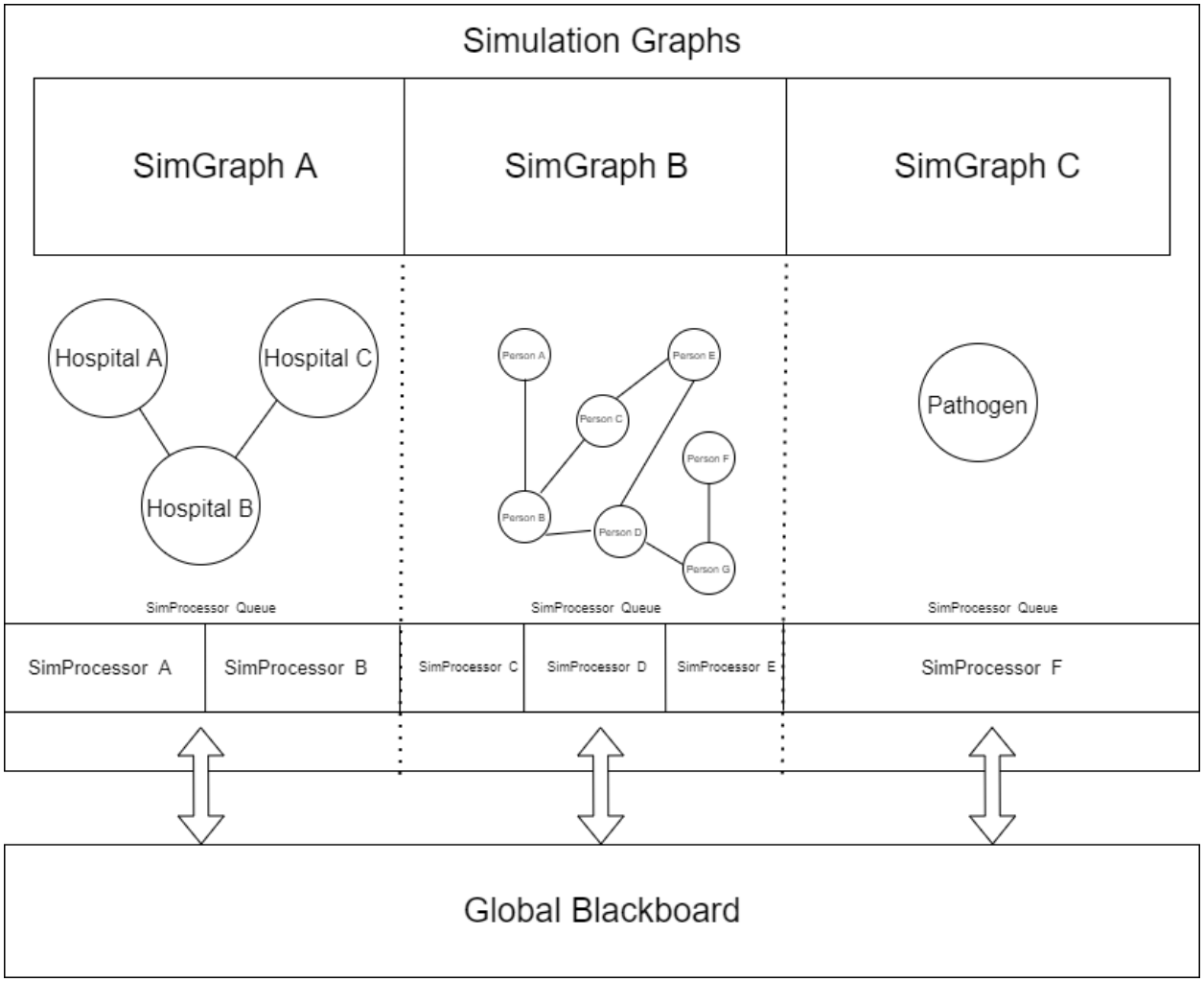
Where SimModels Definitions represent models, SimCommands represent a specific module of work performed by a model within the Simulation Engine. This includes graph initialization, general computation, graph update from script, graph regeneration, or graph aggregation. SimCommand instances are driven by SimModels, which will define a SimCommand for graph initialization and an ordered list of SimCommands to perform during a SimGraph update. SimCommands take only a SimGraph as an argument, through which they can access all the graph's agents as well as the SimEngine object its Blackboard. SimCommands will have types, which will define a generalized task the Simulation Engine can perform.

External Libraries:

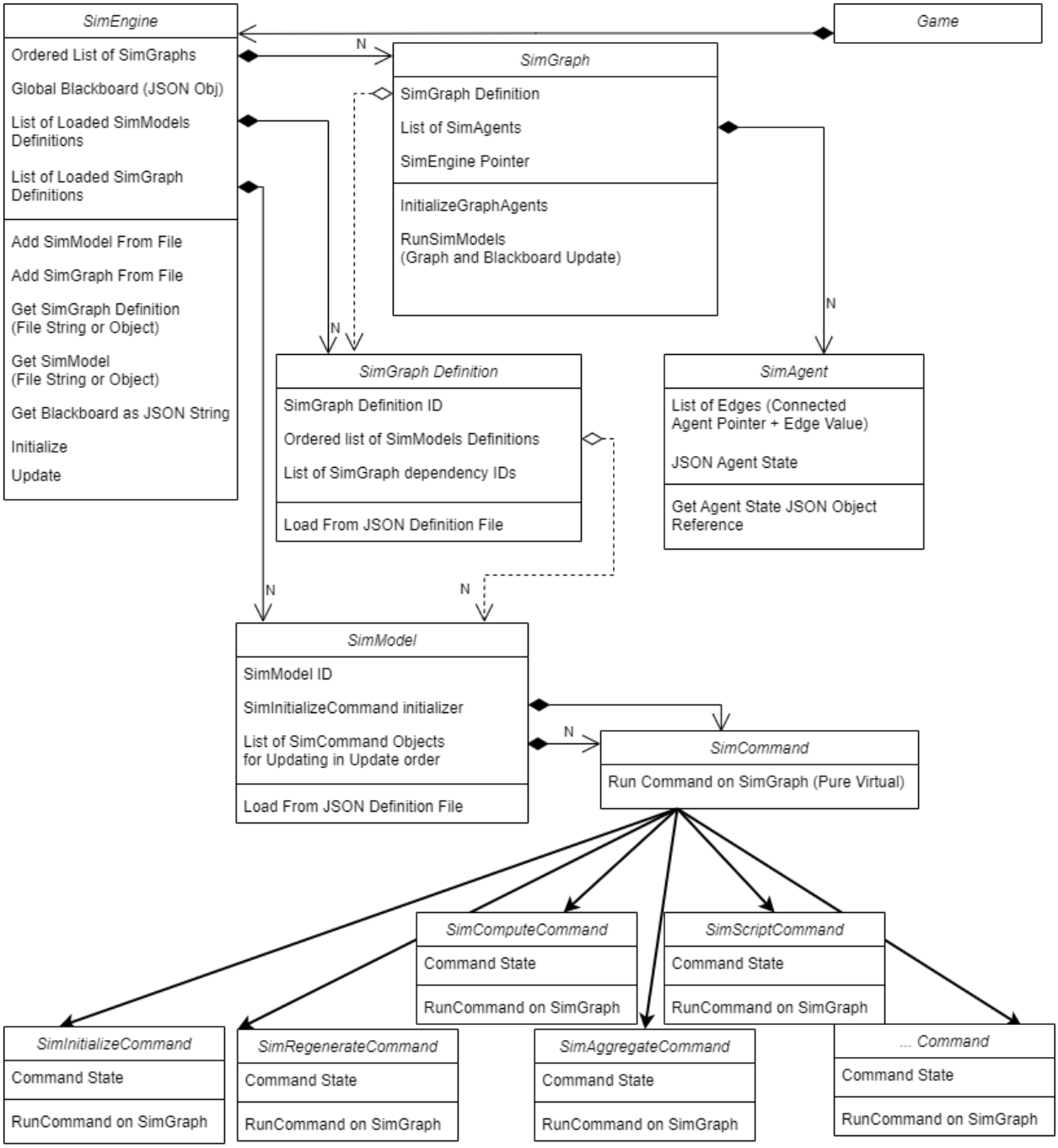
- JSON For Modern C++: https://github.com/nlohmann/json
    - Will be used to encode the runtime determined state of the Global Blackboard and agents.

Architecture Diagrams:




Simulation Graphs
SimGraph A
SimGraph B
SimGraph C
Hospital A
Hospital C
Hospital B
Person A
Person E
Person C
Person F
Person B
Person D
Person G
Pathogen
SimProcessor Queue
SimProcessor Queue
SimProcessor Queue
SimProcessor A
SimProcessor B
SimProcessor C
SimProcessor D
SimProcessor E
SimProcessor F
Global Blackboard

UML and Class Structures



SimEngine Class:

Properties:

- Ordered List of SimGraphs
- Global Blackboard
- List of Loaded SimModel
- List of Loaded SimGraph Definitions

Functions

- Add SimGraph from SimGraph Definition file
    - Stores file path, and defers loading to initialization time
- Add SimModel from SimModel file
    - Stores file path, and defers loading to initialization time
- Initialize Simulation Engine
    - Loads all queued SimGraph Definitions and SimModels. Generates all graphs and Initializes them from SimGraph Definitions. Each stored SimGraph Definition will become an individual SimGraph ordered in the SimEngine graph queue. Graphs are ordered according to the dependencies species in their definition files.
    - Validates SimGraph dependency tree.
- Get Global Blackboard
    - Retrieves the Blackboard JSON Object or JSON String.
- Update Simulation
    - Updates the Simulation. This will update each graph in list order

SimGraph Class:

Properties:

- SimGraph ID
- SimGraph Definition
- List of SimAgents in graph
- SimEngine Pointer

Functions

- InitializeGraphAgents
    - Initializes a SimGraph according to its Definition. This includes:
        - Creating all agents objects
        - Loop over SimModels in SimGraph Definition in order and Run the Initialize Command object on this SimGraph to Initialize the graph's edge networks and agents and the initial values of the blackboard.
        - Validates SimModel dependency tree,
- Run Sim Processors
    - Loop over SimModels in the SimGraph Definition, passing this SimGraph into each SimModelCommand object in the SimModel command list.

SimAgent Class:

Properties:

- Agent State JSON Object
- List of Edges (edge is a pointer to an agent and a value)

Functions

- Get Agent State
    - Retrieves the Agent State JSON Object or JSON String.

SimGraph Definition Class:

Properties:

- SimGraph Definition ID
- Ordered list of SimModels
- List of SimGraph dependencies by ID

Functions

- Load From JSON Definition File
    - Loaded when the SimEngine is Initialized, after all SimModels have been loaded. SimGraph looks up SimModels specified by ID in the definition file and stores them in a list of the SimGraph Definition. Also loads the ID's of all SimGraph dependencies, so that the SimGraph may be ordered in dependency order on the SimEngine graph list.

SimModel Class:

Properties:

- SimModel ID
- InitializeCommand object
- List of SimModelCommand objects

Functions

- Load From JSON Definition File
    - Loaded when the SimEngine is Initialized. Runs through command definitions in the definition file, and creates matching command objects with the proper state. Update Commands are parsed, stored, and used in-order.

SimCommand

Functions

- Run Command on SimGraph
    - Takes in a SimGraph and runs this command object using it. This function is pure virtual.

SimInitializeCommand

Functions

- Run Command on SimGraph
    - Takes in a SimGraph and runs this command object using it. Initializes graphs agent state, initializes relevant blackboard state, and generates initial network of graph edges on the passed in graph.

SimScriptCommand

Functions

- Run Command on SimGraph
    - Takes in a SimGraph and runs this command object using it. Runs and external script to perform graph updates.

SimAggregateCommand

Functions

- Run Command on SimGraph
    - Takes in a SimGraph and runs this command object using it. Specifies how to aggregate values on a SimGraph and the name under which they will be posted to the global blackboard.

SimRegenerateCommand

Functions

- Run Command on SimGraph
    - Takes in a SimGraph and runs this command object using it. Specifies how to regenerate a SimGraph based on the current state of the graph and blackboard.

SimComputationCommand

Functions

- Run Command on SimGraph
    - Takes in a SimGraph and runs this command object using it. Performs a computation using graph/agent/blackboard state and uses it to update some aspect of each agent's state in a SimGraph