Antecedentes

Dentro del ecosistema de Rust ya existen librerías dedicadas a la simulación por eventos discretos, algunas de estas siguen un formalismo especifico mientras que otras buscan ser una librería de uso general para simulaciones, este ultimo es el caso de la librería desarrollada en este documento por lo tal se estudiaran a fondo las librerías similares, como funcionan y por que son diferentes a la desarrollada.

Ecosistema de Rust respecto a librerías de Simulación por eventos discretos.

Las de uso general | Las que siguen un formalismo

Nosotros somos de uso general por lo tanto esas se estudiarán a fondo

Mostrar como funcionan y el cómo difieren de nosotros.

Solo existen 2 que cumplen los criterios: Desim | Simrs

Simrs: Se define a sí misma como “librería de propósito general de simulaciones que provee mecanismos como planificador, estado, colas, etc.”. Desim en cambio se define como “Una librería que implementa un *framework* de simulación por eventos discretos inspirado por la librería SimPy de Python”.

Hablar de las similitudes y diferencias, que tienen en común y que hacemos que ellos no para justificar una nueva librería.

Las 3 librerías proveen:

* Una forma de insertar y agendar funciones en la simulación las cuales tienen la capacidad de auto agendarse y agendar otras funciones.
* Una estructura principal, la cual el usuario interactúa para realizar el modelo, comúnmente llamada Simulation.
* Un mecanismo interno para asegurar avanzar la simulación evento por evento, conocida como lista de eventos futuros y comúnmente denominada Scheduler.
* Una llave identificadora entregada al usuario al momento de ingresar una función al simulador con la cual pueden identificarla y agendar eventos en momentos específicos de la simulación.

Simrs: No permite pausar y reanudar las funciones, para poder ejecutar otro evento la función debe retornar, eliminando todo el estado actual de la función teniendo que empezar de nuevo, este problema es mitigado introduciendo una estructura llamada State utilizada para guardar y extraer data y se puede usar tanto para evitar la pérdida de información como para permitir el intercambio de data entre dos o más funciones.

Simrs:

* No permite pausar / reanudar funciones
  + Causa que la función empiece desde el inicio sin guardar el estado anterior
  + Utiliza una estructura State para evitar este problema
* Modelar procesos complejos que involucren funciones que entrelazan acciones se vuelve más difícil ya que la cantidad de código aumenta por cada acción que requiera sincronización entre funciones (Control Flow).
  + Ejemplo: función A solicita un recurso y solo puede avanzar cuando función B cree el recurso.
  + No poder pausar y reanudar cuando se solicita y se recibe el producto causa que el programador deba modelar los estados de transición de manera manual guardando el estado con el cual la función finalizo y utilizándolo para saber dónde debe continuar.