Discrete Event Simulation

Henry Z. Lo

June 9, 2014

1 Motivation

Suppose you were a branch manager of a bank. How many tellers should you staff to keep customers happy?

With prior knowledge of how often customers come in, how fast it takes to serve a customer, and how long customers wait before leaving, it is possible to answer this question using a simulation:

- Start with k tellers.
- Customers enter the bank at rate λ_1.
  - If a customer enters when a teller is free, process the customer.
  - Otherwise put customer in a queue.
- Customers get processed at rate λ_2, after which the teller is freed and handles the next customer from the dequeue.

See Figure 1 for a diagram. In this simulation, we can measure performance in terms such as average queue length, average wait time, etc.

λ_1 and λ_2 can represent a probability distribution of values (e.g. Poisson) rather than a single rate, but both cases are handled similarly.

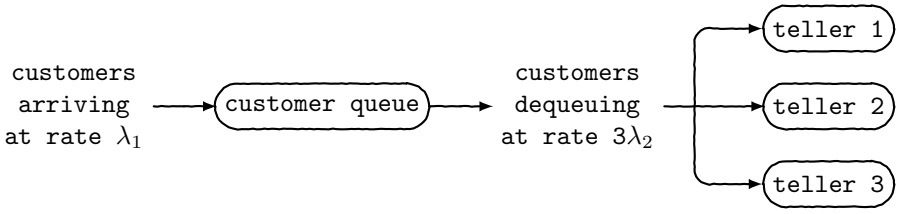

Figure 1: Diagram demonstrating bank teller problem, with 3 tellers.

2 Techniques

In general, there are two methods of performing this type of simulation:

- Simulate the passage of time, then calculate what events happened in the time period.
- Calculate when future events will happen (updating as needed), and then jump from event to event through time.

2.1 Time-driven simulation

Time-driven simulations often need a time jump interval dt to be specified. The general procedure:

1. Move forward dt time units.
2. Calculate all events that happened within that time frame, and take them into account.
3. Loop back to 1 until complete.

For our bank example:

1. Move forward dt time units.
2. Process events in this time span:
    - If there are customers being processed, subtract dt from their process time; finish processing a customer if time exceeds λ_2.
    - Subtract λ_1 from the time until next customer, resetting the time and adding customers as necessary.
    - If a teller is free, dequeue a customer and start processing.
3. Loop back to 1 until complete.

Note that calculating exactly what events occurred in the time step can be fairly difficult (which checks do we do first?). Thus, typically they are synchronized to the end of each leap, i.e. events such as dequeuing and queuing only happen at the end of each time leap.

This can be simplified by a smaller dt, but this means we have to run many more checks. It may make more sense to model event-by-event instead.

2.2 Event-driven simulation

Event-driven simulations can be more difficult to program, but are more accurate and faster. The idea is:

1. Add an initial set of events to happen in the future.

2. Move to the first event to happen in this set.

3. Remove this event, and add new events spawned off by this event.

4. Repeat from step 2 until done.

What data structure is optimal for storing events and when they occur? Priority queues! For our bank teller example:

1. Queue event "new customer" with time $1/\lambda_1$.

2. Dequeue soonest event from priority queue, and subtract that time from all event times.

   - If event is "new customer" and
     - if a teller i is free, mark teller i as occupied, then add event "customer done at teller i" in $1/\lambda_2$.
     - if no teller is free, put customer into customer queue.
   - If event is "customer done at teller i" and
     - if the customer queue is not empty, then dequeue a customer, mark teller i as occupied, then add event "customer done at teller i" in $1/\lambda_2$.
     - if the customer queue is empty, then mark i as free.

3. Repeat step 2 until done.

As events are precomputed in the past, they do not need to be checked at every time step as in time-driven simulations. This makes the simulation process much quicker.

If we want to see the changes between events, e.g. for visualization purposes, then we can run the simulation without having to check for events, until a scheduled event occurs. We demonstrate this with another example.

3 Particle Collisions

Given a bunch of particles (assumed to be spheres), how can you simulate them bouncing off of each other?

Assume each point is a square of size $2s$, a starting position x, y, and a starting velocity v_x, v_y. There are two types of events in this system: collisions between points, and collisions between a point and a wall. Suppose for the sake of simplicity, when two points collide sideways, they reverse horizontal direction, and when they collide vertically, they reverse vertical direction. There is no gravity or air resistance.

The procedure goes as follows:

1. Delete the soonest event, subtract its time from all events.

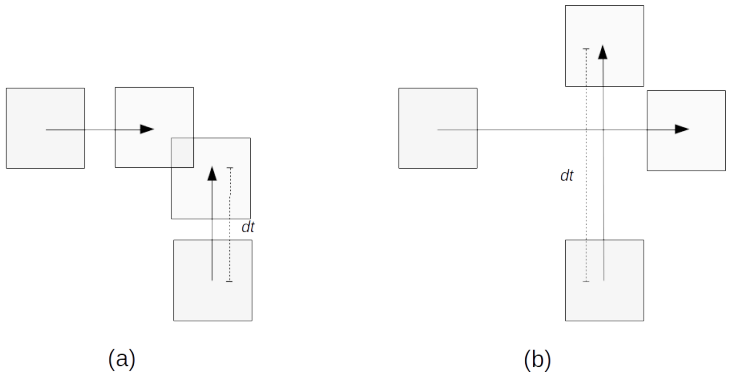

Figure 2: A collision between two particles (a) and two particles passing through each other due to a large dt (b).

2. Disregard the event if one of the particles have been involved in a collision since the event has been added.

3. Process the event, and add others if needed.

   - If two particles are colliding, update velocities of colliding particles (reverse horizontal direction if horizontal collision, etc.)
   - Calculate future collisions for the particles with new velocities.
   - Perform similar calculations for particles colliding with walls.

3.1 Collision detection

The time-driven solution is to perform collision detection at each time step or frame. If particles overlap, then there was a collision.

In order to avoid error, the simulation must be rewound to when the collision occurred and reran in order. Alternatively, we can simply bounce the particles when we detect the collision (rather than when the collision actually happens), and accept the error that comes with this.

There is also the problem of the length of timestep dt, as before. When dt is too large, we may miss some collisions altogether (see Figure 2). When dt is too small, then we have to check collisions many times. In either case, if we have n particles, we need to do $O(n^2)$ checks every time step.

3.2 Event-driven simulation

At the beginning of the simulation, we first determine all the collisions that each particle will have - with the wall, and with each other. As these collisions happen, we move through time and add more collisions to the priority queue, and invalidate some of the future reactions that were previously predicted to happen.

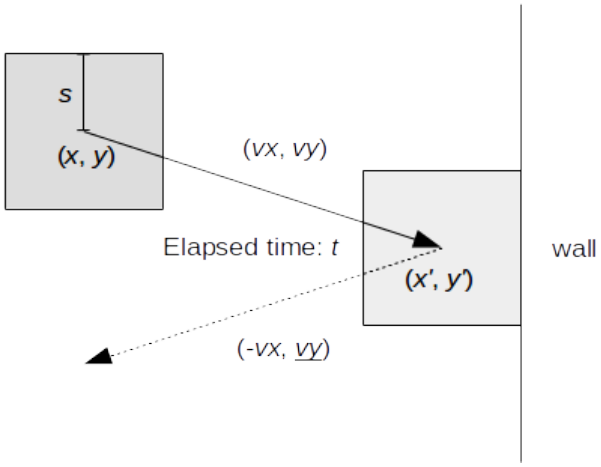

Figure 3: Collision of a particle with wall.

3.2.1 Collisions with walls

Collisions with walls are easiest to calculate, because the wall doesn't move.

Suppose the walls exist at $x = 0$, $y = 0$, $x = 1$, and $y = 1$. Then the time until the particle's edge collides with a wall is t, where:

$$\begin{aligned} x + s + v_x t &= 1 \text{ if } v_x > 0 \\ x - s + v_x t &= 0 \text{ if } v_x < 0 \\ y + s + v_y t &= 1 \text{ if } v_y > 0 \\ y - s + v_y t &= 0 \text{ if } v_y < 0 \end{aligned}$$

Collisions in x don't affect collisions in y. Solving for t:

$$t = \begin{cases} \frac{1-x-s}{v_x} & \text{if } v_x > 0 \\ \frac{-x-s}{v_x} & \text{if } v_x < 0 \\ \frac{1-y-s}{v_y} & \text{if } v_y > 0 \\ \frac{-y-s}{v_y} & \text{if } v_y < 0 \end{cases}$$

See Figure 3 for a visual.

3.3 Collisions between particles

To simplify computations, let s be the same for every particle. There are two ways in which particles can collide - horizontally, after which they reverse horizontal directions, and vertically, in which case they reverse vertical directions.

Since s is the same, one of the corners of one square must touch the edge of the other square. In other words, we have a horizontal collision at time t:

- if $v_x > 0$, $x + s + v_x t = x' - s + v'_x t$ (there is horizontal overlap) AND
  - $y + s + v_y t$ is between $y' + s + v'_y t$ and $y' - s + v'_y t$ OR
  - $y - s + v_y t$ is between $y' + s + v'_y t$ and $y' - s + v'_y t$
- if $v_x < 0$, $x - s + v_x t = x' + s + v'_x t$ AND
  - $y + s + v_y t$ is between $y' + s + v'_y t$ and $y' - s + v'_y t$ OR
  - $y - s + v_y t$ is between $y' + s + v'_y t$ and $y' - s + v'_y t$

The first case describes the first square going right, and the second square going left; the second case describes the opposite situation. Similar equations need to be defined for vertical collisions.

We can solve either of these equations for the time of collision t:

$$\begin{aligned} x + s + v_x t &= x' - s + v'_x t \\ x - x' + s + s &= v'_x t - v_x t \\ \frac{x - x' + 2s}{v'_x - v_x} &= t \end{aligned}$$