Simulation

COS 323

Simulation

One program variable for each element in the system being simulated,

… as opposed to

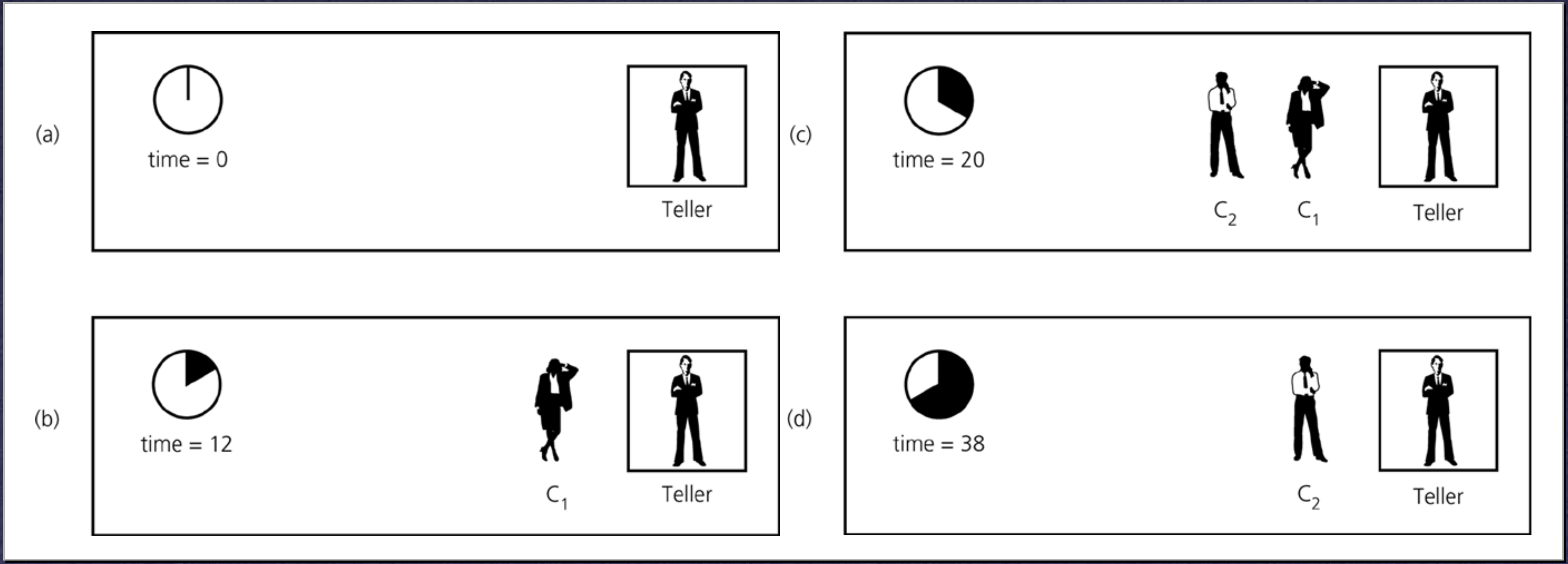
- analytical solution
- formulation of algebraic or differential eqs.

Approaches to Simulation

- Differential equation solvers can be thought of as conducting a *simulation* of a physical system
  - Advance through time
  - “Continuous” equations model change in state
- Some simulations are more “discrete”:
  - Boolean decisions at points in time

Bank Teller

- Simple example: lines at the bank
  - Customers arrive at random times
  - Wait in line(s) until teller available
  - Conduct transaction of random length



Bank Teller

- Simple example: lines at the bank
  - Customers arrive at random times
  - Wait in line(s) until teller available
  - Conduct transaction of random length
- Simulate arbitrary phenomena (e.g. spike in customer rate during lunch)
- Goal: mean and variance of waiting times
  - As a function of customer rate, # tellers, # queues

Bank Teller

- *Time-driven* simulation:
  - Fixed-length time steps
  - Can compute probability of customer(s) arriving, transactions finishing
  - More accurate simulation with shorter time steps, but then have more steps when *nothing* happens

Bank Teller

- *Event-driven* simulation:
  - Variable-length time steps
  - Store sorted list of "events" (customer arrival, transaction finishing)
  - Repeatedly process one event, then fast-forward until scheduled time of next event
  - Good accuracy and efficiency: automatically use time steps appropriate for how much is happening

Time-Driven Simulation: Epidemics

- [Dur95] R. Durrett, "Spatial Epidemic Models," in Epidemic Models: Their Structure and Relation to Data, D. Mollison (ed.), Cambridge University Press, Cambridge, U.K., 1995.

- Discrete-time, discrete-space, discrete-state

Durrett's Epidemic Model

- Time, $t = 0, 1, 2, \ldots$
- Space: orthogonal (square) grid
- State: {susceptible, infected, removed}

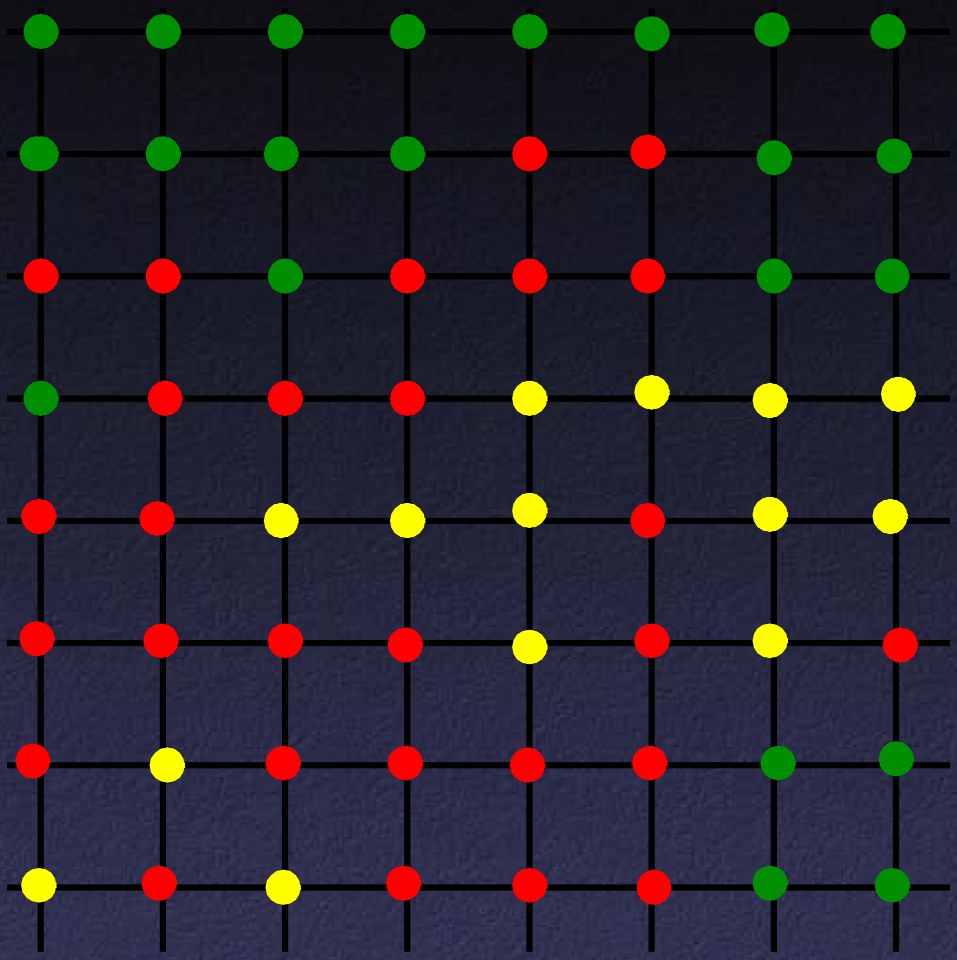
Rules tell us how to get from t to $t+1$ for each spatial location

Each site has 4 neighbors,

contains 0 or 1 individual

Durrett’s Rules (“SIR” Model)

- Susceptible individuals become infected at rate proportional to the number of infected neighbors
- Infected individuals become healthy (removed) at a fixed rate δ
- Removed individuals become susceptible at a fixed rate α



Time, t = 0, 1, 2, …
Space: orthogonal (square) grid
State: {susceptible, infected, removed}

Simulation Results

$\alpha = 0$: No return from removed; immunity is permanent. If δ, recovery rate, is large, epidemic dies out. If δ is less than some critical number, the epidemic spreads *linearly* and approaches a *fixed shape*.

→ Can be formulated and proven as a theorem!

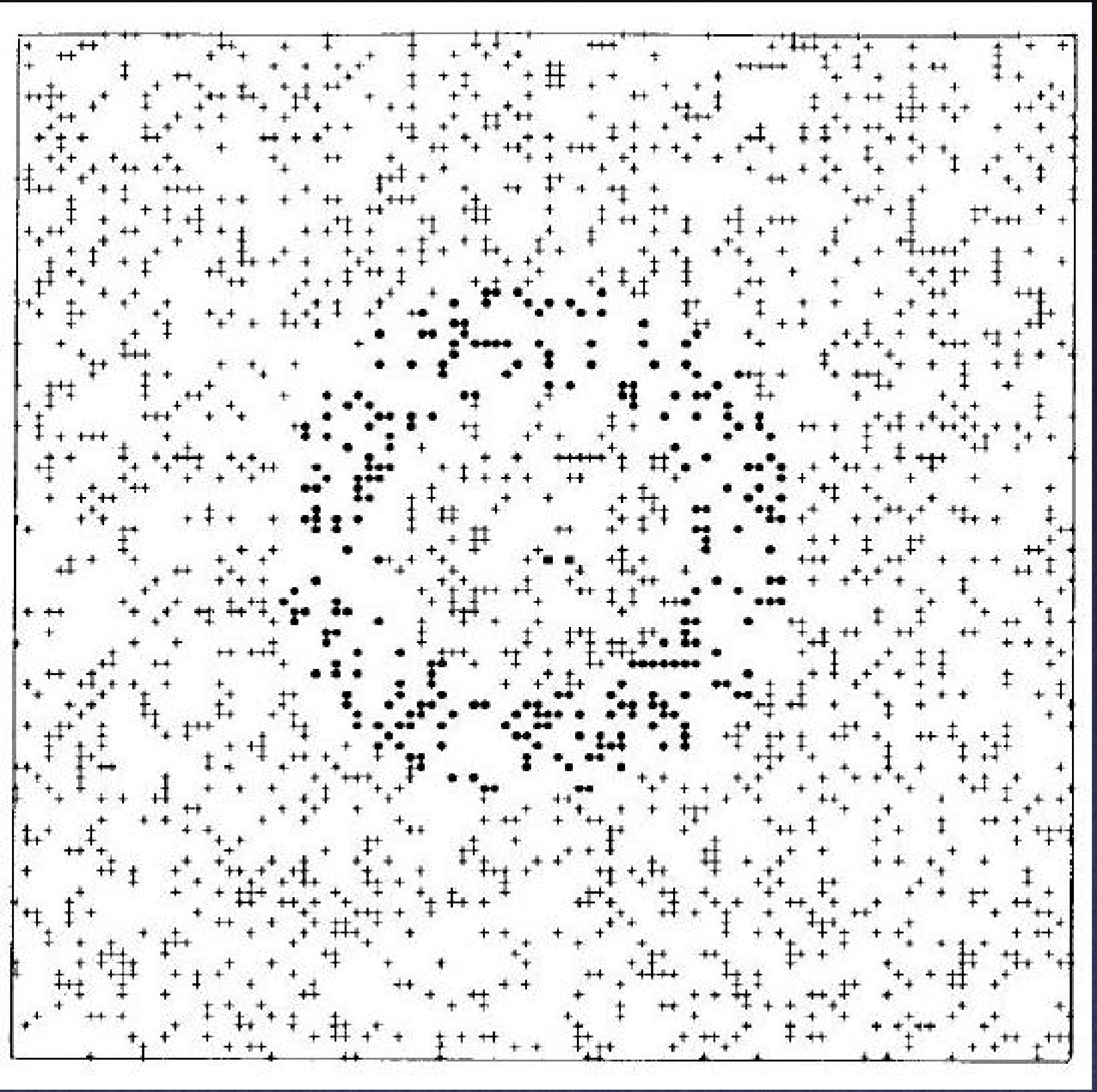
$\alpha > 0$: behavior is more complicated

Empirical Verification

- measles in Glasgow, 1929: 440 ft/week
- Muskrats escape in Bohemia, 1905: square-root of area grows linearly

More recent work:

"Epidemic Thresholds and Vaccination in a Lattice Model of Disease Spread“, C.J. Rhodes and R.M. Anderson, *Theoretical Population Biology* **52**, 101118 (1997) Article No. TP971323.



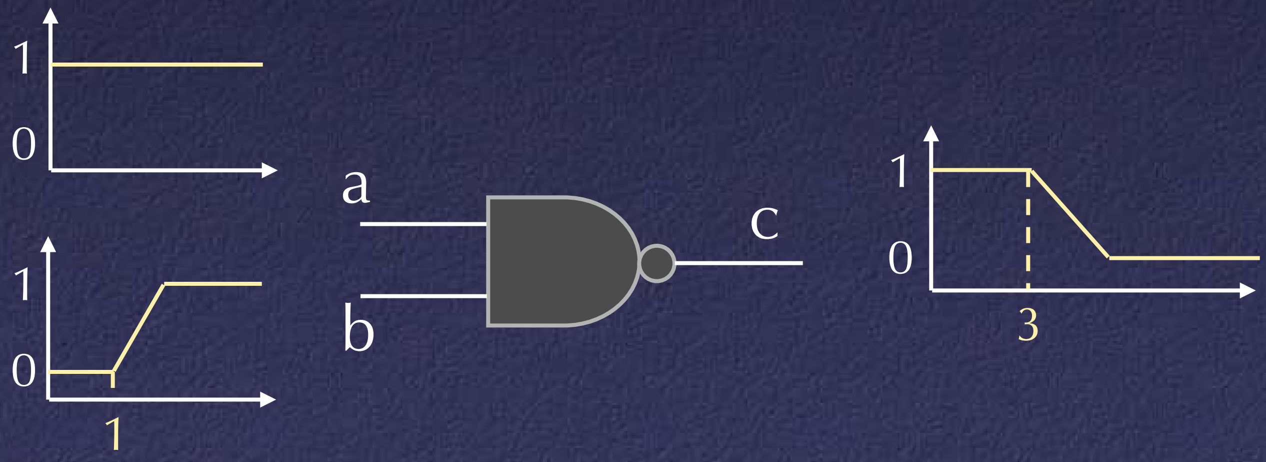
Note ring of vaccinated individuals.

Event-Driven Simulation

- Applications:
  - Traffic during rush hour: effect of different algorithms for controlling traffic lights
  - Load on web server: effect of more machines, scheduling algorithms, etc.

Event-Driven Simulation

- Applications:
  - Circuit/chip simulation: clock rate needed for reliable operation

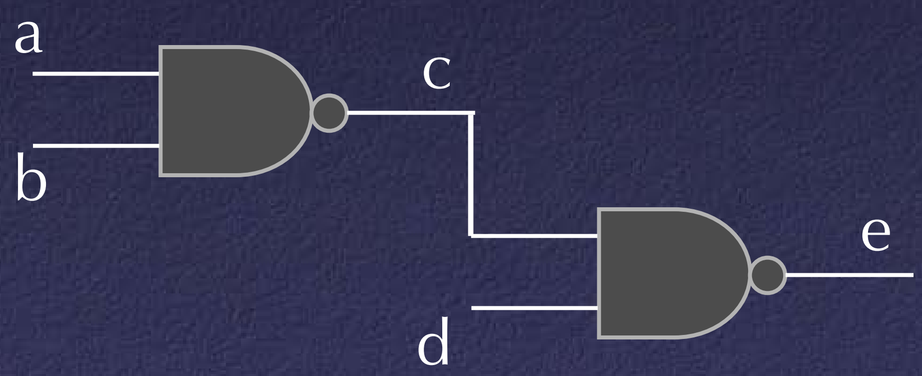


Events:

- Input: b(1)=1
- Output: c(3)=0

Event-Driven Simulation

- Applications:
  - Circuit/chip simulation: clock rate needed for reliable operation



Event-Driven Simulation

- Applications:
  - Multiple code fragments running in parallel, responding to user controls: computer music (e.g., PLOrk)

```
// global gain
gain g => dac;
// set gain
.5 => g.gain;

110.0 => float freq;
6 => int x;

// loop
while( x > 0 )
{
    // connect to gain
    sinosc s => g;
    // change frequency
    freq => s.freq;
    freq * 2.0 => freq;
    // decrement x
    1 -=> x;

    // advance time by 1 second
    1::second => now;
    // disconnect the sinosc
    s =< g;
}
```

Ingredients of Event-Driven Simulations

- Event queue
  - Holds (time, event) tuples
  - Priority queue data structure: supports fast query of event with lowest time
  - Possible implementation: linked list
    O(n) insertion, O(1) query, O(1) deletion
  - Possible implementation: heap, binary tree
    O(log n) insertion, O(1) query, O(log n) deletion

Ingredients of Event-Driven Simulations

- Event loop
  - Pull lowest-time event off event queue
  - Process event
    - Decode what type of event
    - Run appropriate code
    - (Compile statistics)
    - Insert any new events onto queue
  - Repeat.

Ingredients of Event-Driven Simulations

- How are new events scheduled?
  - Some are a direct result of current event.
    Example: when teller finishes a transaction, takes next customer and schedules new transaction completion event
  - Some are background events.
    Example: new customer arrives
  - Some are generated via real-time user input

Using Random Numbers

- Simulation: accounts for uncertainty: biology (large number of individuals), physics (large number of particles, quantum mechanics), human behavior, etc.
- Testing (large number of cases)
- Monte Carlo evaluation
- Run experiments with humans

Sources of “Randomness”

- “Digital Chaos”: Deterministic, complicated. Examples: pseudorandom RNGs in code, digital slot machines.
- “Analog Chaos”: Unknown initial conditions. Examples: roulette wheel, dice, card shuffle, analog slot machines.
- “Truly random”: Quantum mechanics. Examples: some computer hardware-based RNGs

"Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin."

--- John von Neumann (1951)

What Did von Neumann Mean?

- Distinguish between "random" and "pseudorandom"
- Big advantage of pseudorandom: repeatability
- Big disadvantage: not really random

Linear Congruential Generator

- Most common and popular --- simple, fast, pretty good most of the time

$$X = aX_{n-1} + c \quad (\mathrm{mod}\, M)$$

$$X_n / M \quad \text{approx. unif. distr. in } [0,1)$$

$$X_n / (M - 1) \quad \text{approx. unif. distr. in } [0,1]$$

Choosing Good *a*, *c*, *M*

- Maximal period is M
  - Get all the integers in $\{0, 1, \ldots, (M-1)\}$ in some order before repetition, then periodic
- Achieved if and only if:
  - $\gcd(c,M) = 1$
  - a-1 divisible by all prime factors of M
  - if M is a multiple of 4, so is a-1
- That's not all to the story: consecutive numbers clustered in small # of hyperplanes

Using RNGs

How would you…

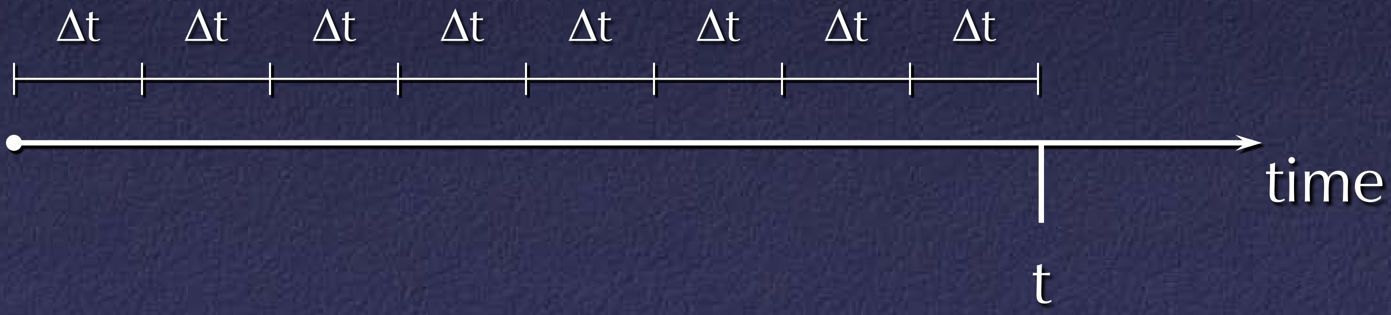
- Choose an integer i between 1 and N randomly
- Choose from a discrete probability distribution; example: $p(heads) = 0.4$, $p(tails) = 0.6$
- Pick a random point in 2-D: square, circle
- Shuffle a deck of cards

Bank Simulation: Scheduling Arrival Events

- Given time of last customer arrival, how to generate time of next arrival?
- Assume arrival rate is uniform over time: *k* customers per hour
- Then in any interval of length Δt, expected number of arrivals is $k\,\Delta t$

Scheduling Arrival Events

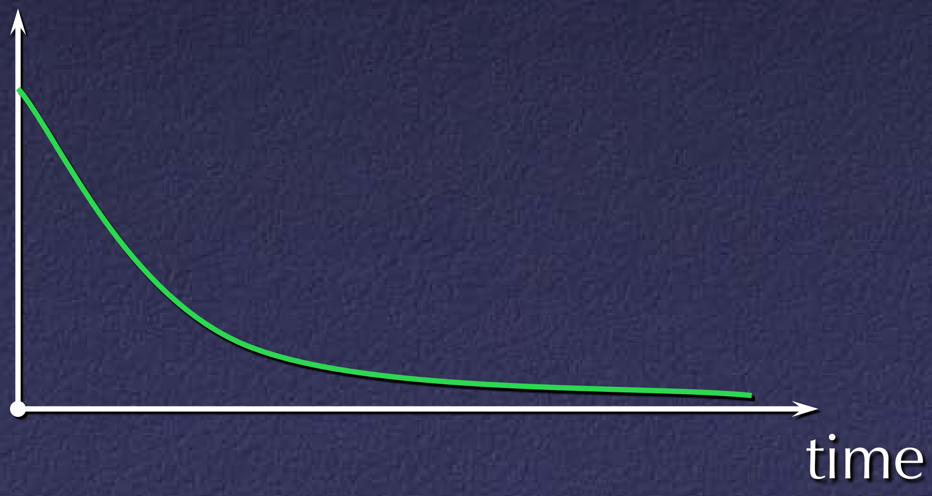
- Probability distribution for next arrival?
  - Equal to probability that there are no arrivals before time t
  - Subdivide into intervals of length Δt



p(no arrivals before t) = p(no arrival between 0 and Δt) *
p(no arrival between Δt and $2\Delta t$) * …

Scheduling Arrival Events

- p(no arrival in interval) = 1 – $k\,\Delta$t
- So, p(no arrivals before t) = $\lim_{\Delta t \to 0} \left(1 - k\,\Delta t\right)^{\frac{t}{\Delta t}} = e^{-kt}$



Scheduling Arrival Events

- Normalize, integrate, invert: time to next arrival event can be found from uniform random variable $\xi \in [0..1]$ via

$$t = -\frac{\ln \xi}{k}$$

Ingredients of Event-Driven Simulations

- How are events handled?
  - Need to run different piece of code depending on type of event
  - Code needs access to data: which teller? which customer?
  - Original motivation for Object-Oriented Programming languages: encapsulate data and code having a particular interface
  - First OO language: Simula (developed in 1960s)

Summary

- Insert events onto queue
- Repeatedly pull them off head of queue
  - Decode
  - Process
  - Add new events

Simulating population genetics (assignment 5)

- review of basic genetics: genes, alleles
- If there are two possible alleles at one site, say A and a, there are in a diploid organism three possible genotypes: AA, aa, Aa, the first two homozygotes, the last heterozygote
- Question: How are these distributed in a population as functions of time?

Why study this?

- Understanding history of evolution, human migration, human diversity
- Understanding relationship between species
- Understanding propagation of genetic diseases
- Agriculture

Approaches, pros and cons

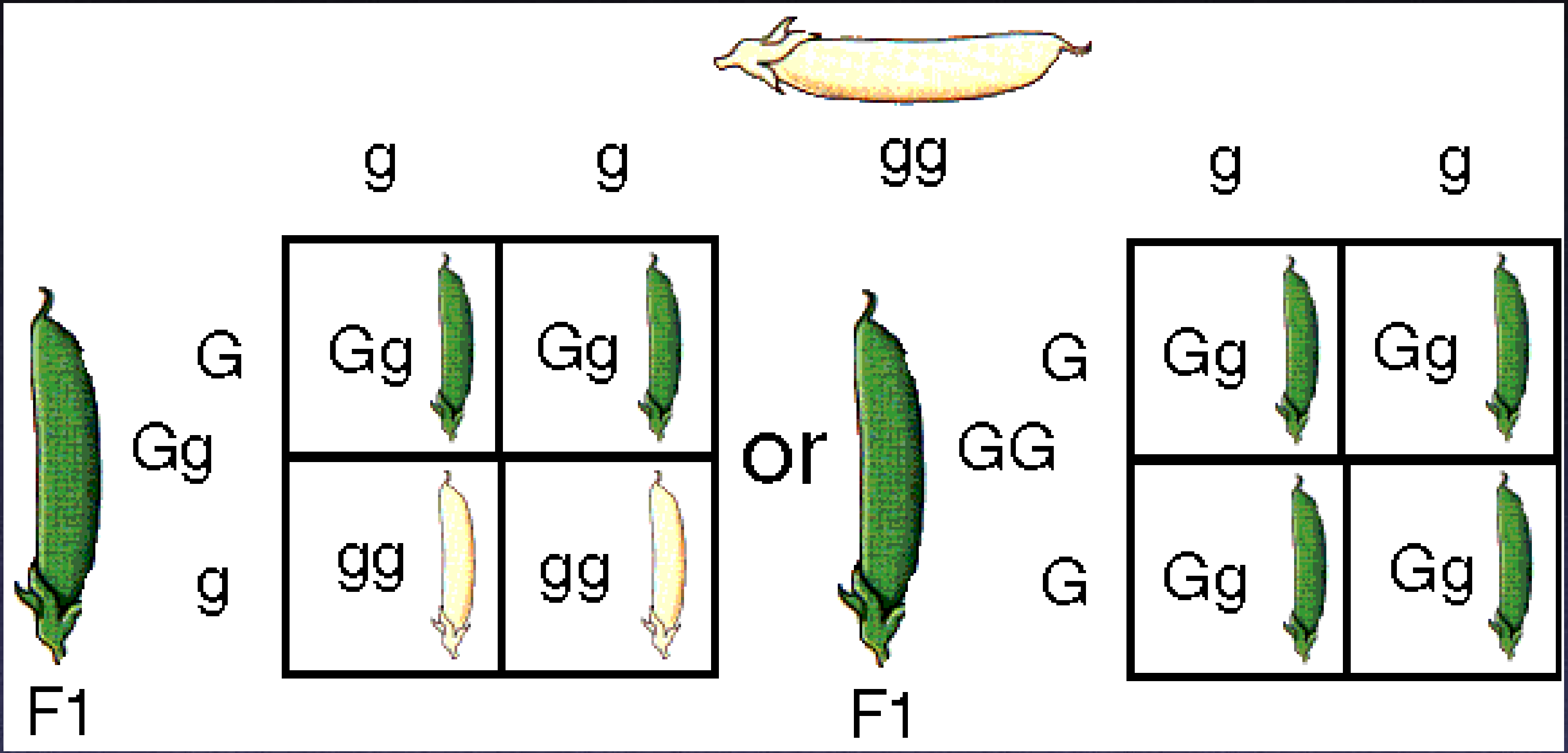
- Field experiment
  - \+ realistic
  - \- hard work for one particular situation
- Mathematical model
  - \+ can yields lots of insight, intuition
  - \- usually uses very simplified models
  - \- not always tractable
- Simulation
  - \+ very flexible
  - \+ works when math doesn't
  - \- not easy to make predictions

19th Century: Darwin et al. didn't know about genes, etc., and used the idea of *blended inheritance*

→But this requires an unreasonably large mutation rate to explain variation, evolution

Enter Mendel…

Gregor Mendel (1822 - 1884)



http://bio.winona.edu/berg/241f00/Lec-note/Mendel.htm

Steven Berg, Winona State

Simplest Model

- Hardy-Weinberg equilibrium
  - If probability of allele A is p, of a is $q=1-p$
    p(AA) = p^2, p(Aa) = $2pq$, p(aa) = q^2
- Not always observed
  - Wahlund effect: fewer heterozygotes if multiple isolated subpopulations
  - Differences in viability, mating preference
- Assignment 5: limitations of theoretical model
  - Finite population, others