15110 Principles of Computing, Carnegie Mellon University

UNIT 10A
Discrete Simulation

1

Last Time

- How to generate pseudo-random numbers
- Using randomness in interesting applications
- Monte Carlo simulations: running many experiments with random inputs to approximate an answer to a question when an analytical solution is impossible or to hard to obtain

Waking Up

- Understanding systems
- Data Visualization
  - http://www.ted.com/talks/hans_rosling_shows_the_best_stats_you_ve_ever_seen?language=en
  - http://www.ted.com/talks/deb_roy_the_birth_of_a_word?language=en (4:15, 9:30)
  - http://www.carbonmap.org/
- Simulations

Understanding Systems

- **Data Visualization** and Simulations are different
- We try to visualize the results of simulations to make it easy to see/understand the systems

because generally what we try to see / understand or predict is complicated because of the nature of systems.

Systems

- Collection of tracks and railway cars → railroad system
- Collection of HW and SW → computer System
- Collection of teachers, students → school system

Dynamic, Interactive, Complicated

How Can we Study a System?

- Experiment with the **actual system**
- Experiment with a **model of the system**
  - **Physical model**
    May not exist, may be unsafe to work with, expensive to build and modify, some change too slowly over time
  - **Mathematical model**
    - **Analytical solution:** Equations or systems may be too complex for closed-form or analytical solution
    - **Simulation:** The imitative representation of the functioning of one system or process by means of the functioning of another, for example a computer program.

Classification due Law and Kelton Simulation, Modeling and Analysis

Computer simulation is a process of making a computer behave like a cow, an airplane, a battlefield, a social system, a terrorist, a HIV virus, a growing tree, a manufacturing plant, a mechanical system, an electric circuit, a stock market, a galaxy, a molecule, or any other thing. This is done with a specific purpose, mainly in order to carry out some ***"what if" experiments*** *over the computer model instead of the real system.*

Modeling and Simulation,
S. Raczynski

Uses of Simulation

- Performance optimization, safety engineering, testing of new technologies.
- Gaining a better understanding of natural and human systems, and making predictions.
- Providing lifelike experiences in training, education, games.

8

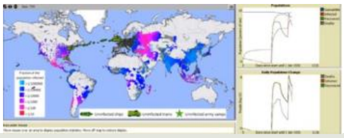
Large Scale Simulations

- Computing power of today enables large scale simulations. For example,
  - Department of Defense: Battle simulations
  - National Center for Atmospheric Research : 1,000 years of climactic changes http://www.youtube.com/watch?v=d8sHvhLvfBo
  - Blue Brain Project at EPFL to reverse engineer the human brain http://www.youtube.com/watch?v=ySgmZOTkQA8

9

Advantages of Using Simulation

- If we use simulation we can
  - Control sources of variation
  - Choose the scale of time
  - Stop and review
  - Replicate results more easily



Models

- A ***model*** is an **abstraction of the real system**.
  It represents the system and the rules that govern the behavior of the system.

- The **model** represents the **system** itself, whereas the **simulation** represents the **operation of the system** over time.

11

Modeling Concerns

Abstraction: In building models a major issue is to achieve a certain level of accuracy while keeping the complexity manageable

- Identify factors that are the most relevant to the functioning of the system.

- How important is time evolution? (Static vs. dynamic models)
- How important is it to capture continuous behavior over time? (Discrete vs. continuous models)
  - Discrete models: essential variables are enumerable such as integers ← our focus
  - Continuous models: essential variables range over non-enumerable sets such as real numbers
- Do parts of the system exhibit random behavior? (Deterministic vs. stochastic models)

12

Computational Science

- Computational sciences use computational models (special kind of mathematical models) as the basis of obtaining scientific knowledge.
- Unifies
  - Modeling, algorithms, simulations
  - Computing environment developed to solve science, engineering, medicine, and humanities problems
- Helps explain and predict phenomena using a mechanistic view

13

Simulation Models are Descriptive

- They **tell us how a system works** under given conditions but not how to set the conditions to make the system work best
- Simulation **does not "optimize"** but it helps us in finding an optimal set of parameter settings.

DISCRETE SIMULATION: A Simple Example

15

Discrete Time and Discrete Events

- **Real time vs. model time**
  - In simulating the movements of a galaxy one hour simulation may cover billions of years
- In discrete simulation we assume time changes in **discrete steps (ticks)** and the states of simulated entities change instantaneously

Experimenting with Models

- **NetLogo** is a programmable modeling environment for simulating natural and social phenomena using discrete simulation.
  - You can learn more about it and download it for free from https://ccl.northwestern.edu/netlogo/.
- It also comes with the **Models Library**, a large collection of pre-written simulations that can be used and modified.
  - In this lecture we will do an example based on *Wilensky, U. (1998). NetLogo Virus model.* http://ccl.northwestern.edu/netlogo/models/Virus. *Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.*

Discrete Simulation of Disease Spread

- We are going to use a dynamic, discrete, stochastic simulation model
  - We want to capture **how the disease spreads over time**
  - We model time discretely as a **sequence of days**, and use discrete variables to capture the **health state** of each person
  - There is **randomness** in how the virus spreads
- Simulate the system execution as a sequence of discrete events that change the state of the system instantaneously at each time step

Example: Flu Virus Simulation

- Goal: Develop a simple simulation that shows graphically how disease spreads through a population.

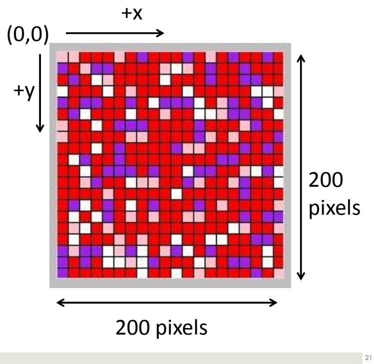
Modeling the Spread of Flu Virus

- Every person is either **healthy, infected, contagious or immune**. We assume that "infected" means infected but not contagious.
- Each day, a healthy person comes in **contact with 4 random people.** If any of those random people is contagious, then the healthy person becomes infected.
- It takes **one day** for the infected person **to become contagious.**
- **After** a person has been contagious for **4 days**, then the person is non-contagious and cannot spread the virus nor can the person get the virus again due to **immunity.**

Displaying the Population

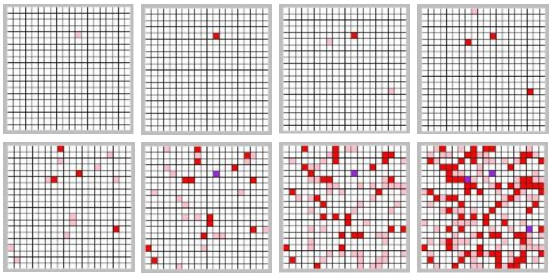
Assumption:
The population consists of 400 people each of which is represented by a cell in the grid.

For simplicity, assume that two cells being a adjacent does NOT mean that they are physically close.



Graphical Simulation

Simulation captures the evolution of the health state of the population over time.
It evolves in discrete steps: change occurs instantaneously as a new day begins.

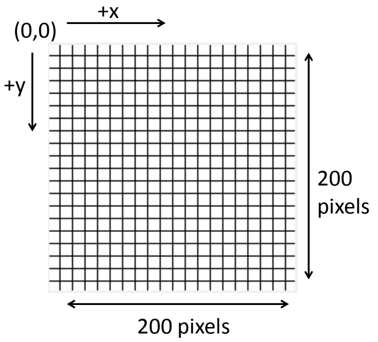


22

Displaying the Population

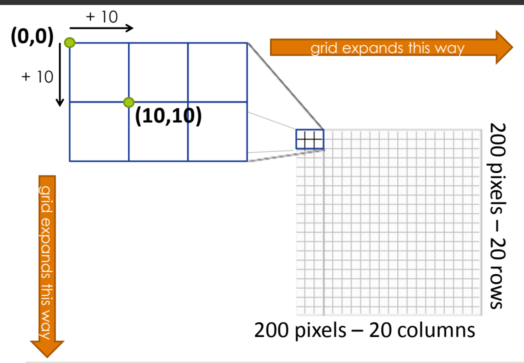
```
Data of 400 people
in 20x20 matrix
   data[0][0]
   data[0][1]
   ...
   data[20][20]


A grid to visualise
200 x 200 pixel
```

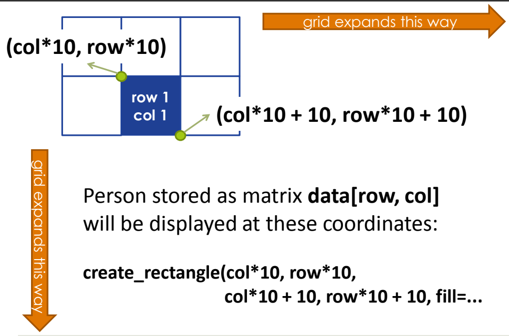


23

Displaying One Person



More Generally For Any Person



Person stored as matrix **data[row, col]** will be displayed at these coordinates:

create_rectangle(col*10, row*10,
col*10 + 10, row*10 + 10, fill=...

Health States

0	white	healthy	**HEALTHY** = 0
1	pink	infected	**INFECTED** = 1
2	red	contagious (day 1)	**DAY1** = 2
3	red	contagious (day 2)	**DAY2** = 3
4	red	contagious (day 3)	**DAY3** = 4
5	red	contagious (day 4)	**DAY4** = 5
6	purple	immune (non-contagious)	**IMMUNE** = 6

The health state of the population will be represented using a 20 by 20 matrix where each entry has one of the values above.

26

Running the Code

```
import tkinter
from tkinter import Canvas
from random import randrange
from time import sleep


# Constants for health states of an individual
HEALTHY = 0
INFECTED = 1
DAY1 = 2
DAY2 = 3
DAY3 = 4
DAY4 = 5
IMMUNE  = 6
```

27

Checking Health State

```
#returns True if person (matrix[r][c]) is healthy
def healthy(matrix, r, c):
    return matrix[r][c] == HEALTHY

#returns True if person (matrix[r][c]) is infected
def infected(matrix, r, c):
    return matrix[r][c] == INFECTED

#returns True if person (matrix[r][c]) is contagious
def contagious(matrix, r, c):
    return matrix[r][c] >= DAY1 and matrix[r][c] <= DAY4

#returns True if person (matrix[r][c]) is immune
def immune(matrix, r, c):
    return matrix[r][c] == IMMUNE
```


Updating the matrix

```
def simNextDay(data):
   nextDayData = []      # create new matrix and initialize
   for i in range(20):
      nextDayData.append([0] * 20)
   for i in range(20):   # create next day
      for j in range(20):
         if immune(data, i, j):
            nextDayData[i][j] = IMMUNE
         elif infected(data,i,j) or contagious(data,i,j):
            nextDayData[i][j] = data[i][j] + 1
         elif healthy(data, i, j):
            nextDayData[i][j] = meetPeople(data, i, j)
   return newMatrix
```

These functions return Boolean value

```
def meetPeople(currMatrix, row, col):
   for counter in range(4):  # repeat 4 times
      if contagious(currMatrix, randrange(20), randrange(20)):
        return INFECTED
   return currMatrix[row][col]
```


Displaying the matrix

```
def display(matrix, c):
   for row in range(len(matrix)):
      for col in range(len(matrix[0])):
         person = matrix[row][col]
         if person == HEALTHY:
            color = "white"
         elif person == INFECTED:
            color = "pink"
         elif person >= DAY1 and person <= DAY4:
            color = "red"
         else:          # non-contagious or wrong input
            color = "purple"
         c.create_rectangle(col*10, row*10,
                            col*10 + 10, row*10 + 10,
                            fill = color)
```

Create_rectangle (topleft_x, topleft_y, bottomright_x, bottomright_y, optional params)

Testing display

```
def test_display():
    window = tkinter.Tk()
    # create a canvas of size 200 X 200
    c = Canvas(window,width=200,height=200)
    c.pack()
    matrix  = []
    # create a randomly filled matrix
    for i in range(20):
        row = []
        for j in range(20):
            row.append(randrange(7))
        matrix.append(row)
    # display the matrix using your display function
    display(matrix,c)
```

32

```
def simulateFlu(numOfDays):
    window = tkinter.Tk()
    # create a canvas of size 200 X 200
    c = Canvas(window,width=200,height=200)
    c.pack()

    # initialize matrix a to all healthy individuals
    population = []
    for i in range(20):
        population.append([0] * 20)

    # infect one random person
    population[randrange(20)][randrange(20)] = INFECTED
    display(population, c)
    sleep(0.3)  # Wait in order to show 1st infection

    # run the simulation for required num of days
    for day in range(0, numOfDays):
        c.delete(tkinter.ALL)
        population = simNextDay(population)
        display(population,c)
        sleep(0.3)   # Wait in order to show change
        c.update()  #Force changes to display -update screen
```

33

What if Our Model Changes?

- If a healthy person contacts a contagious person, she gets sick 40% of the time.

```
def meetPeople(currMatrix, row, col):
   for counter in range(4):  # repeat 4 times
       if contagious(currMatrix, randrange(20), randrange(20))
                                             and randrange(100) < 40 :
          return INFECTED
   return currMatrix[row][col]
```

34

What if Our Model Changes?(cont'd)

- The current model does not capture neighbor relationship. The adjacency of 2 cells does not indicate that they are neighbors.
- What if we used to grid to capture neighbor relationship and assumed that a healthy person gets infected if they have at least one contagious neighbor.

Neighbors

```
cell = matrix[i][j]
north = matrix[i-1][j]        NO!

if i == 0:                    YES!
  north = None
else:
  north = matrix[i-1][j]
```


Next Time

- Continuous simulation