CSE 12:

Basic data structures and object-oriented design

Jacob Whitehill
jake@mplab.ucsd.edu

Lecture One
1 Aug 2011

Welcome

- CSE 12 is a programming-oriented introduction to fundamental data structures of computer science.
- By the end of this course, you will hopefully know:
  - What data structures are.
  - Why selecting the right data structure is important.
  - When to use a particular type of data structure.
  - How the most common data structures are implemented in code and in memory.
  - How object-orientation can facilitate good program design.

Administrivia.

Course structure

- 4 lectures/week:
- M, T, W, Th
  11:00a - 12:20p
  Here
- 4 programming assignments (45%)
- 1 midterm (20%), 1 final (30%)
- Class participation (5%)
  - CSE 12 Moodle web forum
  - In-class unannounced quizzes (yay!)

Teaching staff

- Lecturer:
  - Me
- Teaching assistant (TA):
  - Vineet Kumar
- Tutors/graders:
  - Stephanie Yeh
  - Anthony Dang
  - Kerwin Azares

Course website

- http://ieng6.ucsd.edu/~cs12v

Moodle forum

- http://csemoodle.ucsd.edu
- Appropriate contributions:
  - **Questions** about programming projects, data structures, or anything else in computer science.
  - **Answers** to the above.
  - **Suggestions** for topics you want to hear about during lecture and/or discussion section.

Discussion section

- Go to CSE 12 Moodle web forum and list your availability!

Warning on grammar

- “Data” is technically a plural (“your data are so lovely”)
- BUT:
  - Data in plural form can sound very pretentious.
- In this course I will alternate between them inconsistently.

Storing information in a river of 1's and 0's.

Consider the following request:

- ```
  Dear you,

  Please email me your phone
  number.

  Thanks,
  Someone else
  ```
  ```

Transmitting your phone

- To transmit your phone number by email, the 10 digits must be converted into a binary sequence of 1's and 0's.
  - **That's all you (ever) have to work with**.

Your computer

Someone else's computer

Transmitting your phone

- Phone numbers in USA: 10 decimal digits, e.g., (858) 822-5241.
- **Step 1**: concatenate 10 digits into one simple string.
  - 8588225241
- **Step 2**: encode each digit using a few bits (how many?).

Transmitting your phone

- For example, to encode digit 8 using 4 bits, we write `1000`.
- To encode 1 in binary, we write `0001`.
- Given the binary codes for each decimal digit, we concatenate all the codes together (in order), e.g.,

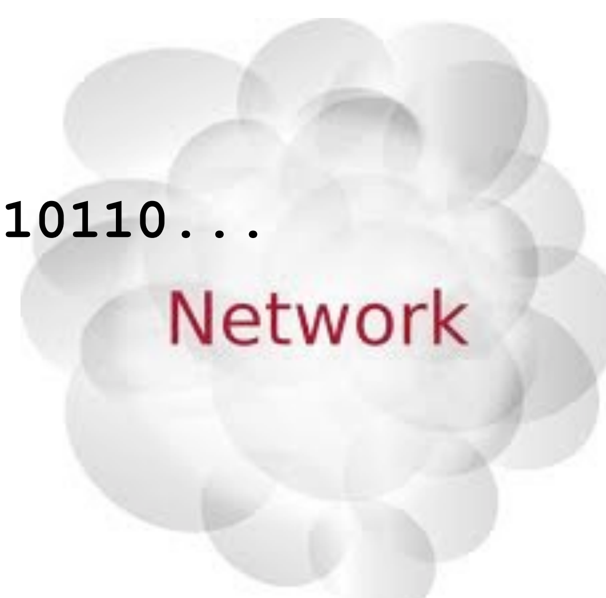
  `100001011000100000100010010100100100001`

Transmitting your phone

- We then send this bit sequence over the network to Someone else.

Your computer

1000010110...

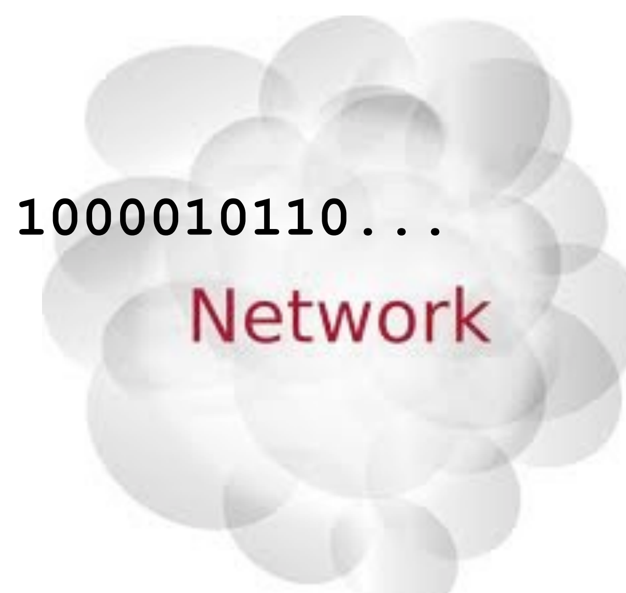


Someone else's computer

Transmitting your phone

- We then send this bit sequence over the network to Someone else.

Your computer

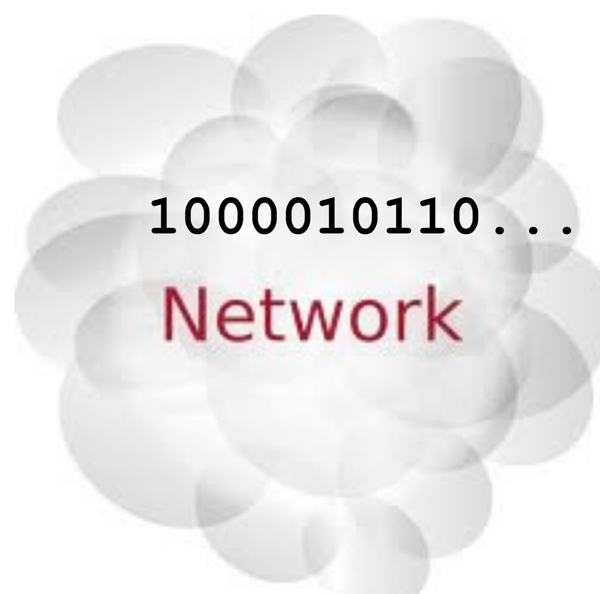


Someone else's computer

Transmitting your phone

- We then send this bit sequence over the network to Someone else.

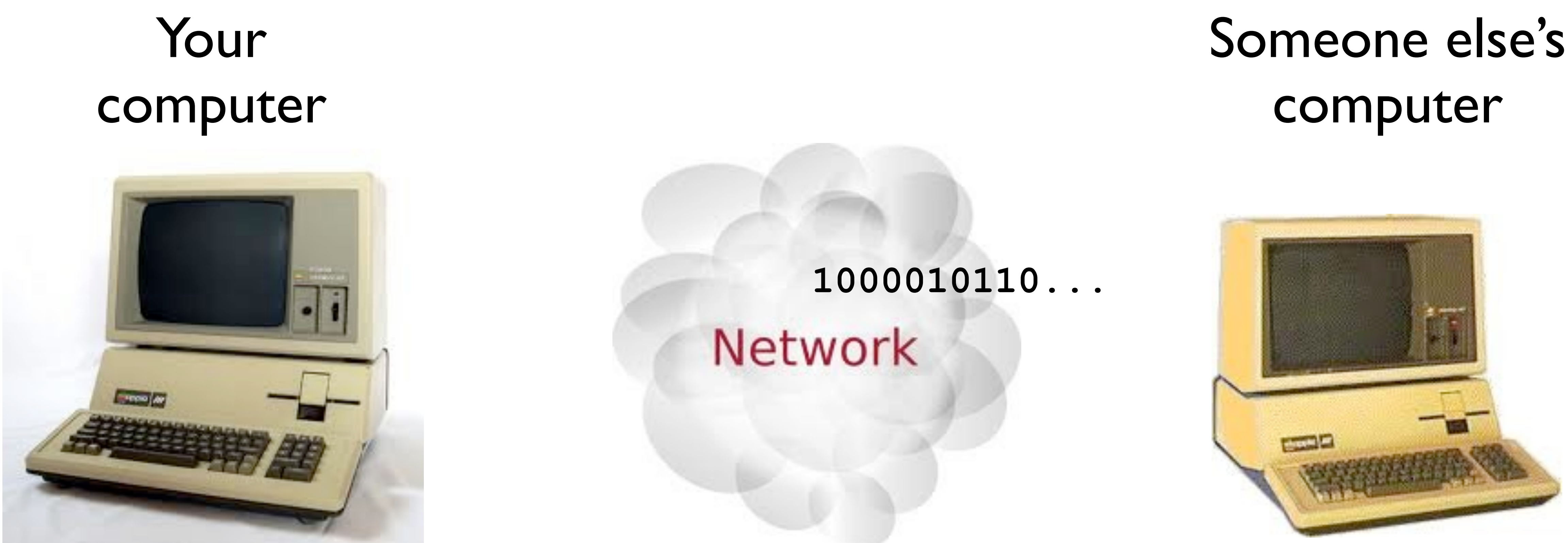
Your computer



Someone else's computer

Transmitting your phone

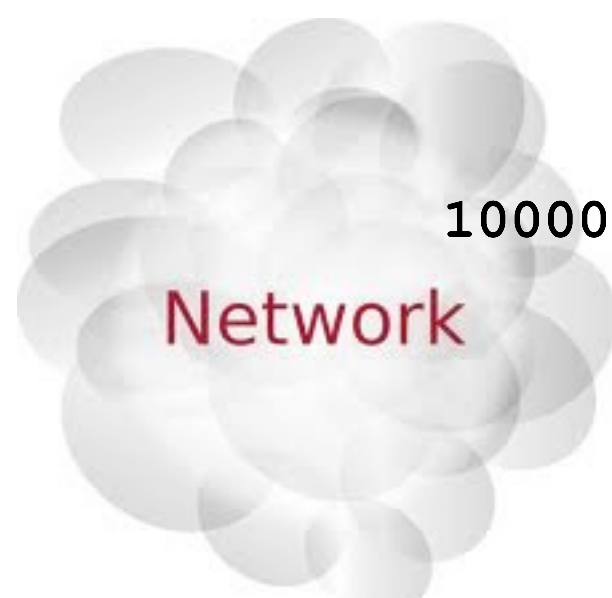
- We then send this bit sequence over the network to Someone else.



Transmitting your phone

- We then send this bit sequence over the network to Someone else.

Your computer



1000010110...

Someone else's computer

Decoding your phone

- Someone else then has to **decode** the bit sequence you sent:

  `1000010110001000001000100101001001000001`

- How?

Decoding your phone

- Questions:
  - How does Someone else know when to stop reading bits?
  - Consider the binary code for decimal digit 1:
    - `0001`
  - Isn't this wasteful? Why can't we just encode 1 as `1` (with no 0's)?

Decoding your phone

- Consider the number 515...:
  - `0101   0001   0101 . . .`
- Without the leading 0's, we have:
  - `101   1   101 . . .`
- The problem is that Someone else doesn't see the spaces -- all they see is `1011101`.
- No way to infer where each digit starts/ends.
- We need to **structure** the **data** by making each decimal digit have the same length.

A slightly more ambitious task...

How would you handle this?

- Dear Google,

  Please send me all
  of your Google Earth
  data.

  Thanks,
  Some Other Company

Data transmission

- To handle such a request, we could either:
  - Transmit the data over a network using a very long sequence of 1's and 0's.
  - Write the entire Google Earth database to a large number of hard disks (containing 1's and 0's) of high capacity.
- **Ultimately, we need to encode a huge amount of information as 1's and 0's.**

1's and 0's

- What we would *like* to do is send separate "groups" of bits for different parts of the data:

1101010000110011010
1001100101011101010
0110100011001000101
1101000101000101001
0110000111010110100
1010100011001001011
1010001...

Images of Australia

1011011010
0000001111
0010000001
1100101110
...

Marker of Eiffel Tower

1001001110
0010101110
11011...

Marker of Mt. Everest

0000001000000110100
0011100001110110011
1001110001101100000
1001011001010110001
0110101011111010010
00111...

Images of Europe

1's and 0's

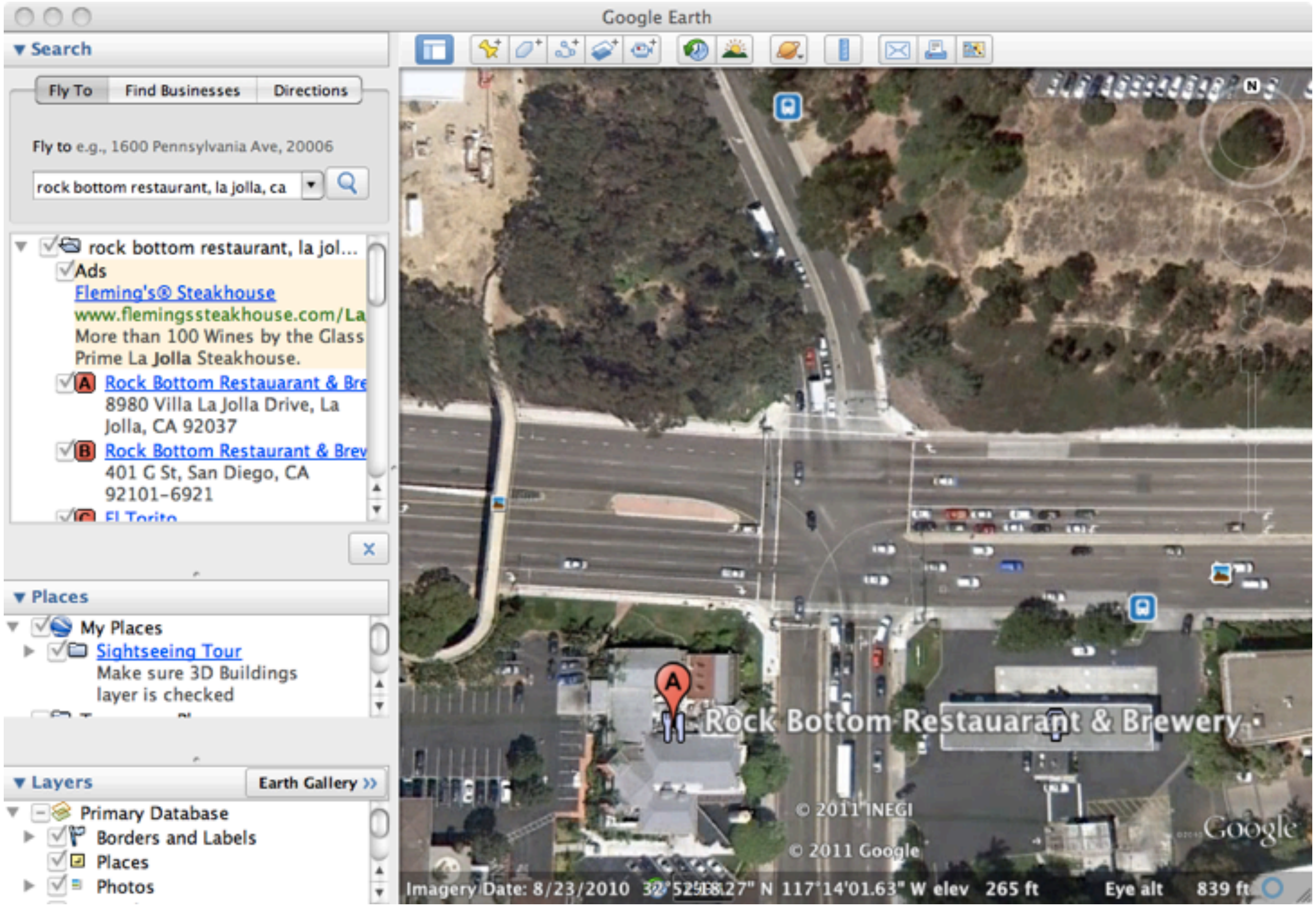
- In the real world, we must unfortunately encode everything in a *single stream of 1's and 0's*.
- We must somehow *structure* our *data* (1's and 0's) to allow meaningful information to be extracted.
- First, how many bits are we dealing with?

How much data is there in Google Earth?

- Satellite imagery:
  - Surface area of earth: 510,072,000 km^2
- Markers:
  - Landmarks, campgrounds, museums, restaurants...
- (and much more)

Satellite imagery

- Estimate resolution at highest zoom:



How much data is there in Google Earth?

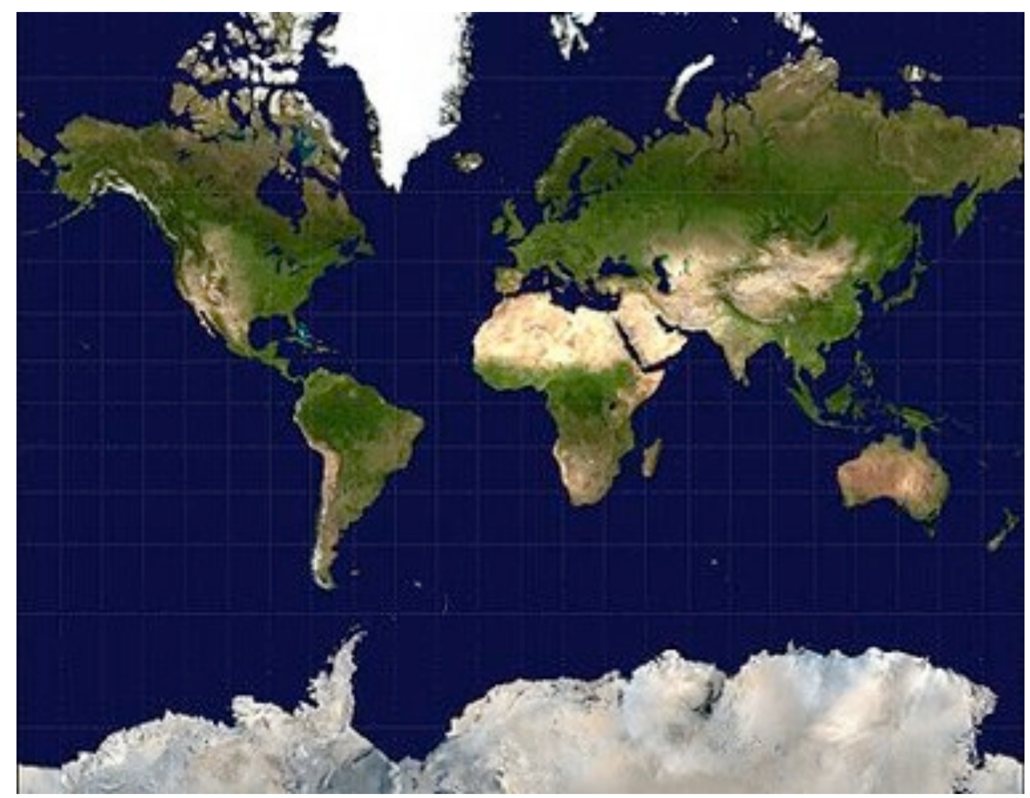
- Back-of-the-envelope calculations:
  - Satellite imagery: 510,072,000 km^2 * X pixels/km^2 * 3 bytes/pixel * 8 bits/byte = ?? bits

...0111001001101100111000100111110100010 11...

Satellite imagery

- How might we store the satellite imagery?
    1. Convert image of 3-D spherical surface to 2-D image.

Mercator projection



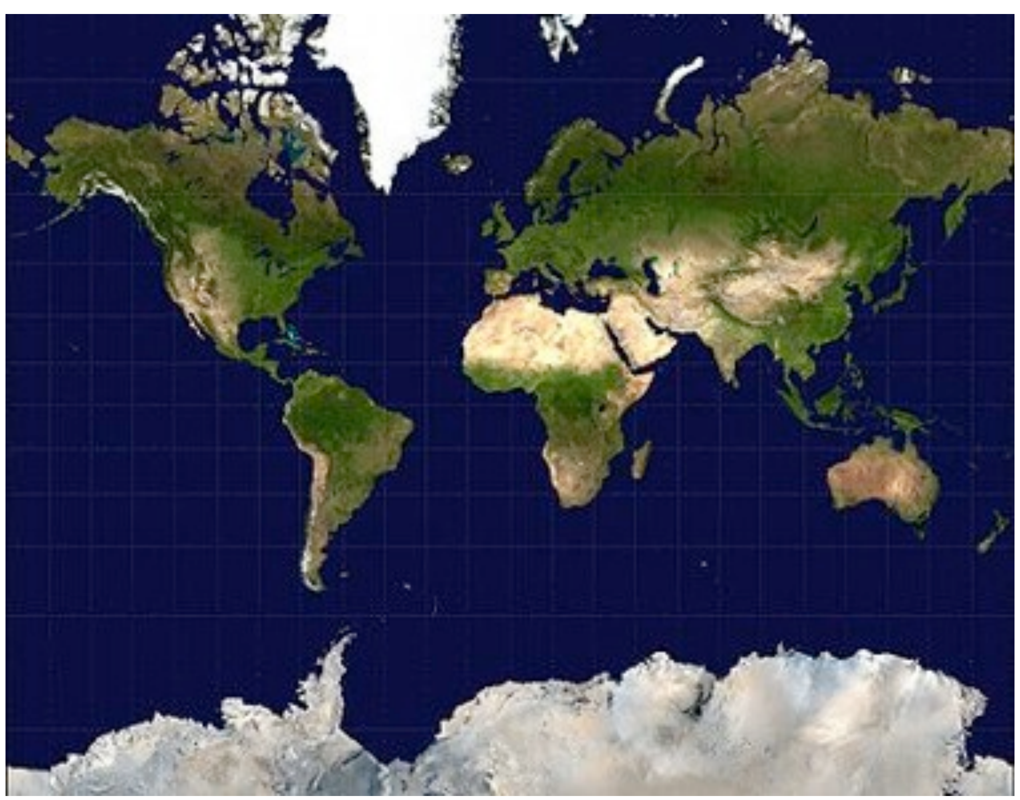
3-D sphere

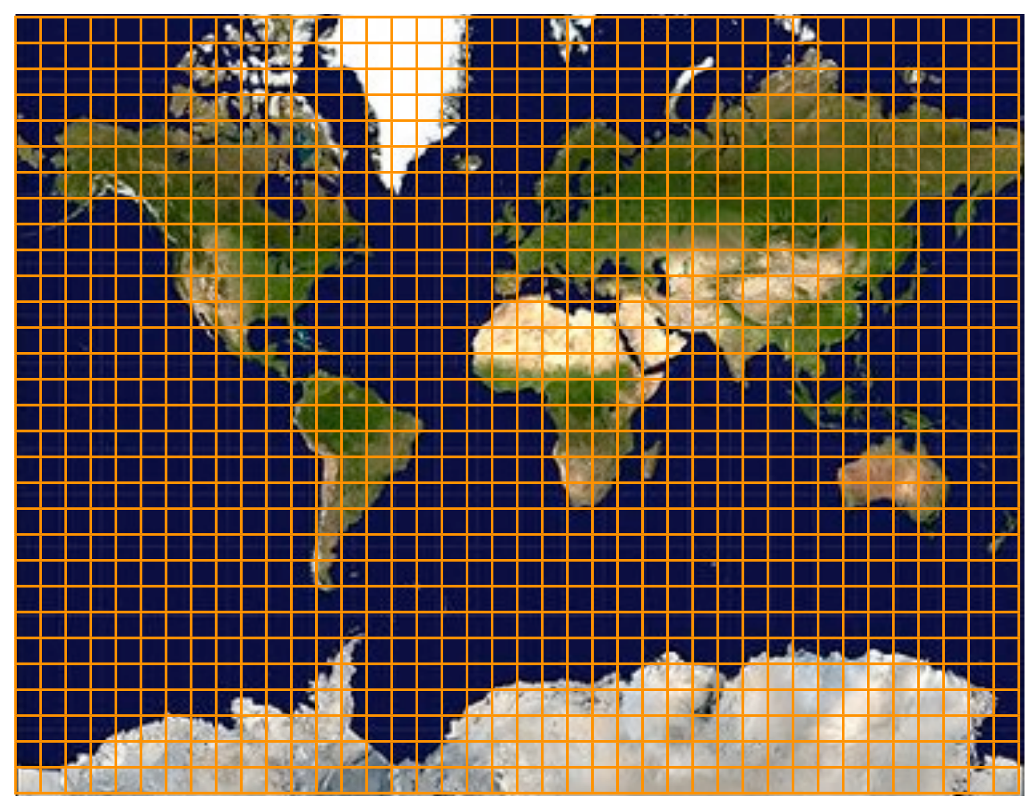
2-D image

Satellite imagery

- What is an image in terms of 1's and 0's?

  1. Image is a 2-D grid of pixels.

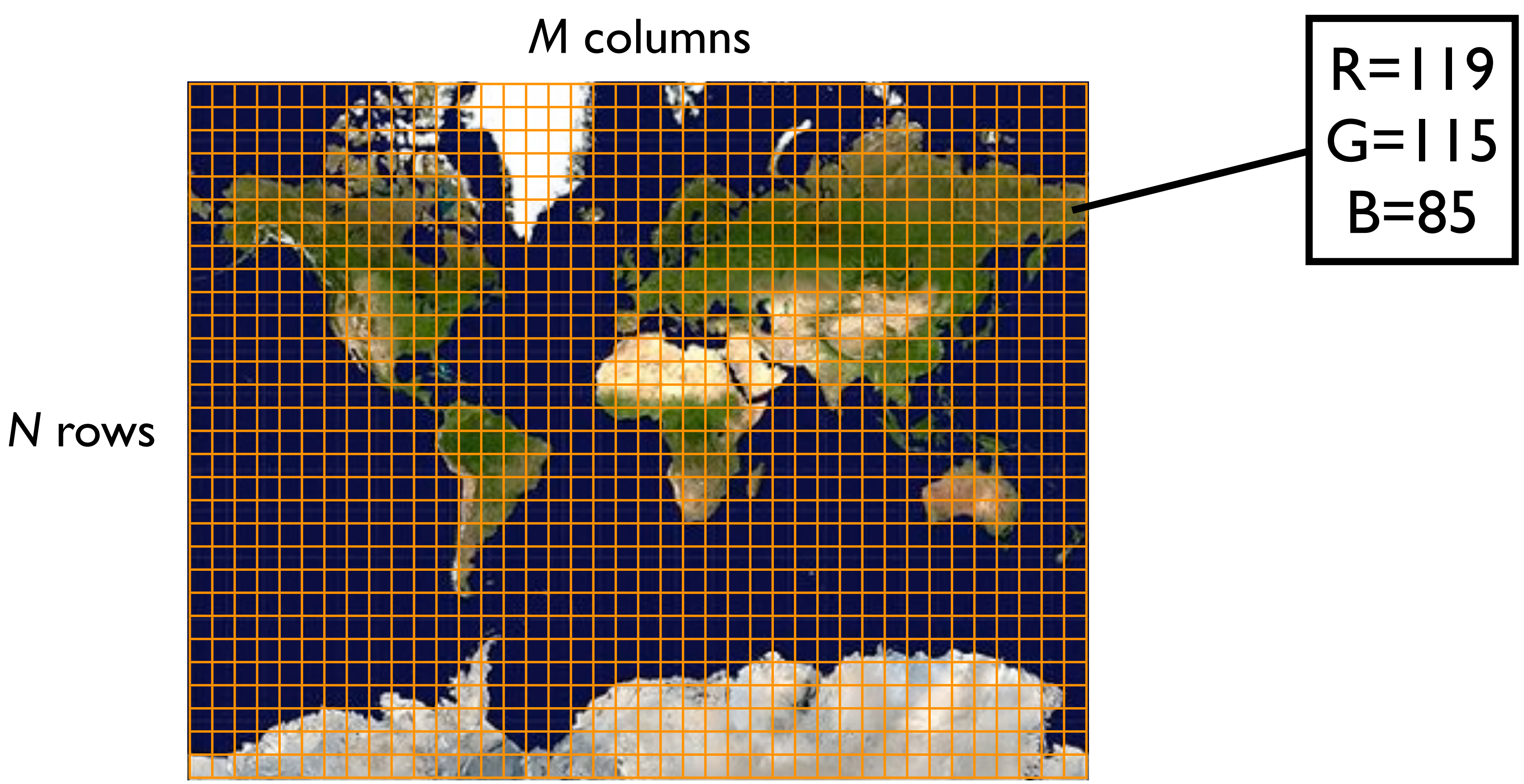




Satellite imagery

- What is an image in terms of 1's and 0's?

2. Each pixel consists of red, blue, and green color channels.
   Each color channel is between 0-255.



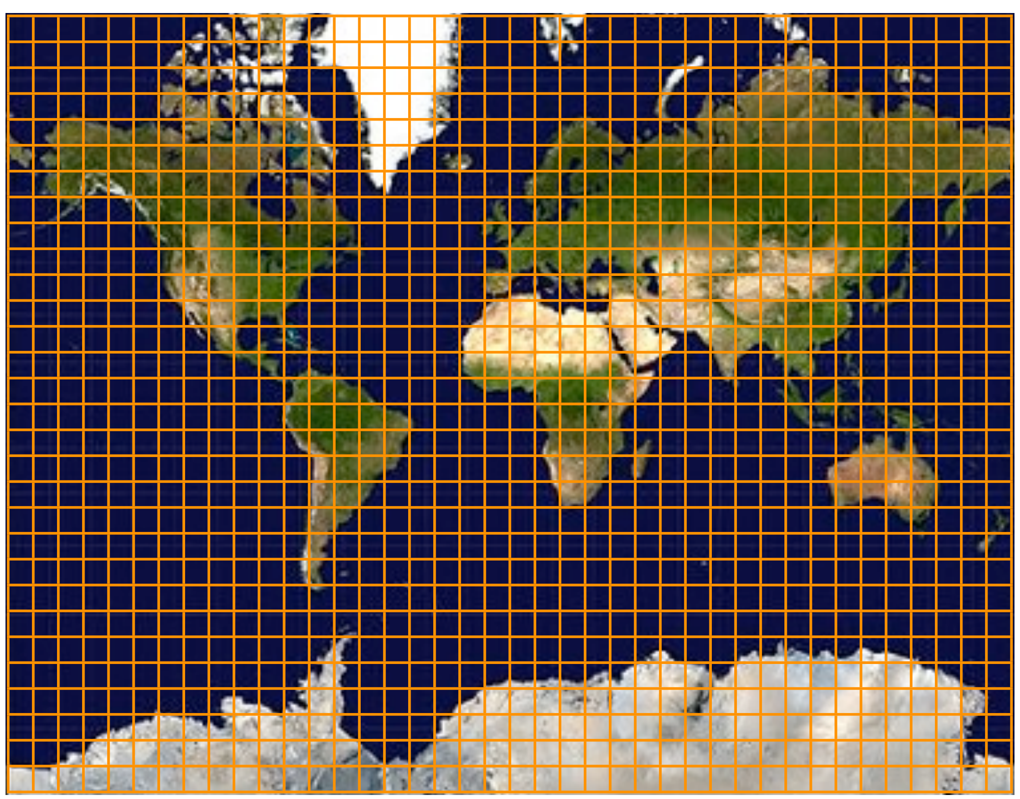
Satellite imagery

- What is an image in terms of 1's and 0's?

  2. We can represent a 2-D image as a 3N x M array of numbers.

M columns

N rows



→

(R,G,B) for one pixel

M columns

255	255	255	255	255	255	...
255	14	23	24	24	23	
255	17	25	27	25	24	
255	255	255	255	255	255	
255	255	255	255	64	64	
255	53	61	63	64	64	
...						...

3N rows

Satellite imagery

- What is an image in terms of 1's and 0's?

3. We can concatenate the 3N * M array elements into one large vector of numbers.

M columns

3N rows

255	255	255	255	255	255	...
255	14	23	24	24	23	
255	17	25	27	25	24	
255	255	255	255	255	255	
255	255	255	255	64	64	
255	53	61	63	64	64	
...						...

2-D array

→

255	255	255	255	14	17	255	23	25	255	...

1-D vector of numbers

Satellite imagery

- What is an image in terms of 1's and 0's?

4. We convert each element of the vector (0-255) to binary representation.

255	255	255	255	14	17	255	23	25	255	...

$\longrightarrow$

1	1	1	1	1	1	1	1	1	1	...

1-D vector of integers

1-D binary vector

Done!

Storing the markers

- Each marker consists of:
  - Location (latitude & longitude)
  - Name
- Example:
  - **Belem, Brazil**:
    1 deg 28’ S latitude,
    48 deg 29’ W longitude

Storing the markers

- Location, e.g.,
  1 deg 28' S, 48 deg 29' W
  - Integers (4 bytes) for degrees/minutes of latitude/longitude.
  - Single characters (1 byte) for South/North/West/East.
- Name:
  - String of characters (bytes), e.g., "Belem, Brazil"

Storing the markers

- To encode a marker we simply concatenate the name and the location: *

```
Aberdeen, Scotland00570009N00020009W
```

- We then concatenate *all* the markers:

```
Aberdeen, Scotland00570009N00020009WAdelaide,
Australia00340055S01380036EAlgiers,
Algeria00360050N00030000EAmsterdam,
Netherlands00520022N00040053EAnkara, Turkey...
```

- We then convert the string of characters into a binary sequence:

```
10000000000000001000000000000000000000000100000001000000000000000000000000000000001000000010000000100000000
00000000000000010000000000000000000000000000000000000000100000001000000000000000000000000000000001000000000
00000000000000000000000000000000000000000000000000000000000000001000000010000000100000000000000001000000000
00000100000001000000000000000100000000000000000000000001000000010000000100000000000000000000000001000000000
00001000000010000000000000000000000001000000000000000000000000000000000000100000001000000010000000000000000000
00010000000100000000000000000000000000000001000000000000000100000001000000000000000100000001000000010000000
00000000010000000000000000000000010000000000000001000000010000000000000000100000001000000000000000000000000
0100000000000000000000000000000000010000000000...
```

* This is slightly fictitious -- the integers would actually appear quite differently if printed as text.

Google Earth data: One huge binary sequence

- We concatenate the satellite imagery and markers into one huge binary sequence (**serialization**).

Satellite image

11111110011100111000010111110000111110101001010101010
01110011000001100000111110010110111011000001110001
101000101110010111111101101110000111111101000110111
1110100010110111101010001110001011001000...

Markers

- The serialized data can then be easily:
    - Loaded into memory.
    - Written to disk/DVD.
    - Transmitted over a network.

Google Earth data: One huge binary sequence

- After serializing the Google Earth data, we can send it to Some Other Company.
- But...

Google

Send data

...011010001011...

Some Other Company

Some other company: How do I parse the 0's and 1's?

- How will Some Other Company know how to decode the data?
  - In order for the binary sequence to be of any value, we must know how the data are **structured**.
    - *What* is stored *where* in the binary sequence?

Some other company: How do I parse the 0's and 1's?

- Where does satellite imagery *end*, and each marker *start*?

```
...
111111100111001110000101111100001111101010010101010
011100110000011000001111100101101110110000011110001
101000101110010111111101101110000111111101000110111
1110100010110111101010001110001011001000...
```

Satellite image

Marker 1

Marker 2

- It is necessary to give Some Other Company information on how to separate the individual data fields.

Extracting satellite image data

- How can we let Some Other Company know how long the satellite image data subsequence is?

Extracting satellite image data

- How can we let Some Other Company know how long the satellite image data subsequence is?
  - Encode M and N as integers just before the image data?
  - Encode M and N as integers just after the image data?
  - Encode 3*M*N*8 as one integer just before the image data?

Extracting satellite image data

- Encode M and N as integers just **before** the image data?
- M*N pixels, 3 colors each, 8 bits for each color channel and pixel = 3*M*N*8
-

M (320)

N (200)

```
<start>
0000000101000000000000011001000111111100111001
1100001011111000011111010100101010100111001100
0001100000111110010110111011000001111000110100
0101...
<end>
```

Satellite image
(3*M*N*8 bits)

Extracting satellite image data

- Encode M and N as integers just **after** the image data?

Satellite image
(3*M*N*8 bits)

```
<start>
...
11111110011100111000010111110000111110101001010
01010011100110000011000001111100101101110110000
00111100011010001010000000101000000000000001100
1000
<end>
```

M (320)

N (200)

Extracting satellite image data

- Encode M and N as integers just **after** the image data?

Satellite image
(3*M*N*8 bits) ?

```
<start>
...
1111111001110011100001011111000011111010100101
0101001110011000001100000111110010110111011000
0011110001101000101000000010100000000000001100
1000
<end>
```

? M (320)

? N (200)

Extracting satellite image data

- Encode 3*M*N*8 as one integer just **before** the image data?

3*M*N*8

Satellite image
(3*M*N*8 bits)

```
<start>
...
0000000000010111011100000000000011111110011100
1110000101111100001111101010010101010011100110
0000110000011111001011011101100000111100011010
00101
<end>
```

Extracting marker data

- Assume:
  - 1st marker starts immediately after satellite data.
  - 2nd marker starts immediately after 1st, etc.
- But how do we know the length (in bits) of each marker?
  - **<start>**

Satellite image

Marker 2

...
11111110011100111000010111110000111110101001010
10100111001100000110000011111001011011101100000
11110001101000101110010111111101101110000111111
10100011011111101000101101111010100011100010110
01000...
<end>

Marker 1

Monday, August 1, 2011

Extracting marker data

- One possible solution:
  - At beginning of binary sequence for marker each, we encode its length.

- Marker 1 length   Marker 1

```
<start>
...
1111111001110011100001011111000011111010100101010
1001110011000001100000111110010110111011000001111
0001101000101010000001100101111111011011100001111
1110100011011111101000101101111010100011100010110
01000...
<end>
```

Extracting marker data

- Once we have extracted each marker, we must then extract the name and location of each marker.
    - We can use a similar scheme as above -- encode the length of each field.
- Finally, we must encode the number of markers.
    - Put this integer in the bit stream just before the first marker.

Extracting Google Earth data

- We can now extract both the satellite image data and the markers.
- How might this look in Java code...?

Data structures and object-orientation.

Google Earth data extraction in Java

- We can use the Image class for the satellite image.
- Let's assume there's some nice Location class to represent latitude+longitude.
- Let's create a Marker class:

```
class Marker {
  private String _name;
  private Location _location;
}
```

Google Earth data extraction in Java

- To extract satellite image and markers from the bit sequence, let's define 2 "pseudo-Java" methods: *

- ```
  // Should be called at beginning of entire Google
  // Earth bit sequence.
  public Image extractSatelliteImage (bit[] sequence)
  { ... }
  ```

- ```
  // Should be called on the bit sequence just after
  // the satellite data.
  public Marker[] extractMarkers (bit[] sequence)
  { ... }
  ```

* Type "bit" doesn't actually exist in Java.

```
// Assume that integers such as “width” and “height”
// are encoded as 16-bit integers.
public Image extractSatelliteImage (bit[] sequence) {
  int width = ... // read first 16 bits
  int height = ... // read next 16 bits
  Image image = ... // create width-by-height image

  for (int i = 0; i < height; i++) {
    for (int j = 0; j < width; j++) {
      int r = ... // read red channel
      int g = ... // read green channel
      int b = ... // read blue channel
      image[i][j] = // set (i,j)th pixel to (r,g,b)
    }
  }

  return image;
}
```

```
public Marker[] extractMarkers (bit[] sequence) {
  int numMarkers = ... // read 16 bits
  for (int i = 0; i < numMarkers; i++) {
    Marker marker =
      new Marker(/* read from bit sequence */);
    // Add marker to array markers.
  }

  return markers;
}
```

Google Earth data extraction in Java

- Danger 1...
- Danger 2...

Google Earth data extraction in Java

- Danger 1 -- **wrong bits**: the caller calls a method on the wrong part of the Google Earth data bit sequence.
- Danger 2 -- **mismatched image/markers**: if there are multiple planets (Google Earth, Google Mars, etc.), then the caller might mismatch the set of markers with the wrong planet. (This happens to me all the time!)

In come the objects...

- One of the purposes of objects in Java is to prevent these problems from occurring.
- Objects **encapsulate** related pieces of data.
- Example: define a class `GooglePlanet`.

class `GoogleEarth`

- ```
  class GooglePlanet {
    Image _satelliteImage;
    Marker[] _markers;

    // Should start at the beginning of entire
    // Google Planet bit sequence.
    GooglePlanet (bit[] sequence) { ... }
  }
  ```
- Now, the constructor of `GooglePlanet` handles the initialization.
  - Fewer opportunities for caller to mess up -- only one bit sequence to pass in.
  ```

class `GoogleEarth`

- ```
  class GooglePlanet {
    Image _satelliteImage;
    Marker[] _markers;

    // Should start at the beginning of entire
    // Google Planet bit sequence.
    GooglePlanet (bit[] sequence) { ... }
  }
  ```

- Also, the satellite image and markers are eternally coupled (how romantic) -- there is no danger of mismatching markers and images.
  ```

Object-orientation and data structures

- These are two benefits of data encapsulation. (There are others.)
- Data encapsulation is a benefit of object-orientation.
- Other benefits include:
  - **Polymorphism**
  - **Abstraction**
  - (More on these later in the course...)

Time complexity and space complexity.

What do we use class `GoogleEarthData` for?

- How is the Google Earth data used in practice? Common use case:
  - User is navigating somewhere on Earth, and wants to fetch a list of markers nearby (e.g., country-western bars).

Finding local markers

- To implement this "query" functionality, let's add a method to class `Marker`:
- (For simplicity, a marker is either "close to" the user's location, or "not close to" her/him.)

```
class Marker {
   ...
   public boolean isCloseTo (Location location) {
      ...
   }
}
```

Finding local markers

- We also add a method to `GooglePlanet`:

```
class GooglePlanet {
  Image _satelliteImage;
  Marker[] _markers;

  public Marker[] findLocalMarkers
                     (Location location) {
    ...
  }
}
```

- How is this method implemented?

Finding local markers

- Algorithm:
  - Create empty list `localMarkers`.
  - For each `Marker` **`i`** in `_markers`:
    - If `_markers[i].isCloseTo(location)`:
      - Add `_markers[i]` **to** `localMarkers`.
  - Return `localMarkers`.

Time cost

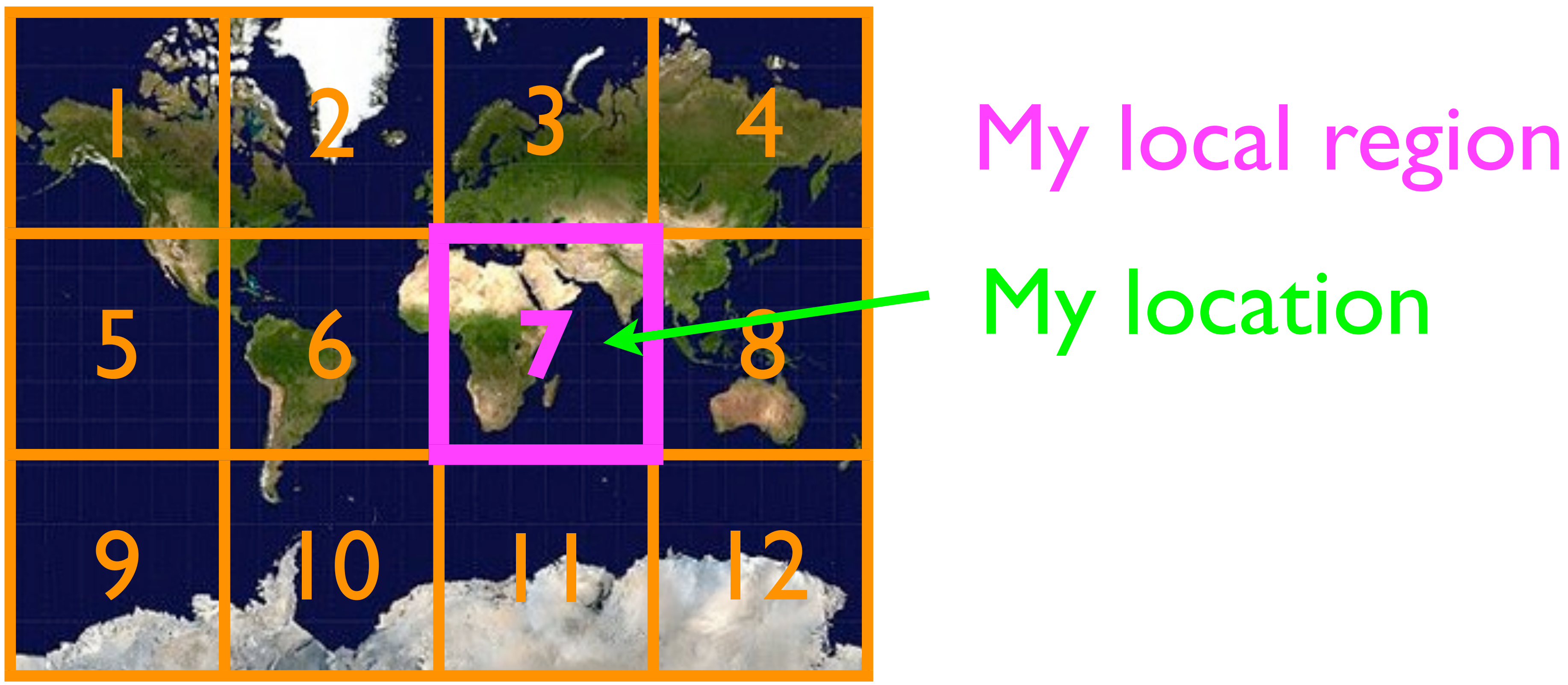
- If there are 10,000,000 markers in `_markers`, how many times will the loop iterate?
- What if location is not close to any of the markers in `_markers`?
- Problem -- we have to call `isCloseTo()` on markers that are very far away from `location`.

Time cost

- Implementing `_markers` as a simple array causes `findLocalMarkers` to run fairly slowly ("linear time" in this case).
- The running time of `findLocalMarkers` can be estimated from the **time complexity** of that method.
- The time complexity of an algorithm depends on the **data structures** it uses.

Finding local markers more quickly

- How can we speed up the search for local markers?
- Simple approach: divide the markers into *regions*.
- When looking for local markers, we search only within our *local region*.



Finding local markers more quickly

```
class GooglePlanet {
  Image _satelliteImage;
  Marker[] _region1, _region2, ..., _region12;
  ...
}
```

Finding local markers more quickly

- New algorithm for finding local markers:
  - Create empty list `localMarkers`.
  - Determine which `localRegion` contains `location`.
  - For each `Marker` **`i`** in `localRegion`:
    - If `localRegion[i].isCloseTo(location)`:
      - Add `localRegion[i]` to `localMarkers`.
  - Return `localMarkers`.

Finding local markers more quickly

- If there are 12 regions, then this algorithm will run about 12x faster than our first one.
  - **Time cost** has been reduced.
- BUT -- there is a penalty.
- Instead of just one Marker[]:

```
Marker[] _markers;
```

  we now have 12 Marker[]'s:

```
Marker[] _region1, _region2, ..., _region12;
```

Array overhead

- Each array of type `Marker[]` incurs some overhead.
  - In Java, the length of an array is stored in its `length` field. This takes up space!
- So...we have *decreased* the **time cost** at the expense of *increasing* **space cost**.
- **There is an inherent tension between minimizing time cost and minimizing space cost.**
- The space cost of a data structure can be estimated from its **space complexity**.

Finding local markers more quickly

- Our “grid” of local regions is still not great in terms of time cost.
- A **tree** data structure could yield much better performance (more later in the course...).

“Code complexity”

- Sometimes, it may be reasonable to sacrifice some time/space costs to make the code simpler.
  - Especially on small amounts of data, an “easy to implement” data structure may often be the best solution.

Choosing the right data structure.

Choosing the right data structure

- When writing a program, very often you will be solving the same kinds of problems over and over again:
  - How do I store a collection of addresses?
  - How do I sort these numbers?
  - How can I find the largest object quickly?
  - How can I fetch a person's profile picture from a dataset quickly given just her name?

Choosing the right data structure

- Rather than having to rediscover the solution every time, you should learn how the **fundamental data structures** of computer science work.
- Data structures covered in this course:
  - List
  - Stack
  - Queue
  - Heap
  - Tree
  - Hash table
  - Graph

The rest of this course.

CSE 12

- In this course you will study the **properties** of and practice **implementing** the data structures listed above (list, stack, queue, heap, tree, hash table, graph).

CSE 12

- **Question**: "Why should I spend time implementing a data structure that has been implemented literally millions of times before, when superbly written, highly efficient, thoroughly tested, standardized library versions exist for free?"

CSE 12

- My answer:
    - Once you thoroughly understand the basic data structures, by all means use library code.
    - BUT: There is no better way of gaining a thorough understanding than than having to implement those structures yourself.

Programming Project #1

- In your zeroth programming project you will write a Hello Whirled program.
- In your first programming project you will implement a doubly-linked list.
- In your second programming project you will **test** your linked list implementation -- and the implementations of your classmates.

Getting help

- If you need help on the programming project, you can come to:
  - Me during office hours or in the lab.
  - The TA during discussion section, his office hours, or in the lab.
  - The tutors in the lab.
  - The web forum (csemoodle).
    - Each programming project will be allocated its own thread.

Getting help

- You may also get help from:
  - Your peers.
    - This is natural.
    - This is beneficial.
    - This is slightly dangerous...

Obtaining help from peers

- It is ok to talk to your peers about CSE 12 programming assignments *without writing any notes on paper or on the computer*.
  - Equivalent to having a phone call.
- It is ok to discuss a programming assignment while using paper and pencil as a visual aid.
  - BUT: *you must destroy these notes before returning to your computer*.
- You may *not* look at someone else's code on any computer screen.

Obtaining help from the Internet

- Feel free to consult general texts on data structures on the Internet:
  - Online textbooks
  - Wikipedia
- You are *not* permitted to download anyone's source code to complete an assignment.

Enforcement

- We will be using automatic code comparison programs to identify copied code.
- In a previous course I taught, I caught one student cheating; he failed the course and his graduation was delayed by 1 year :-(.

Participation in class

- Please ask questions during class if you are curious about or do not understand something.
- It is not a bother to answer questions.
  - Answering questions is my job.
  - Answering "stupid" questions is my job.
- Every student (and instructor) sometimes makes mistakes.
  - Please show respect to classmates (and me) at all times.