Vectors and Grids

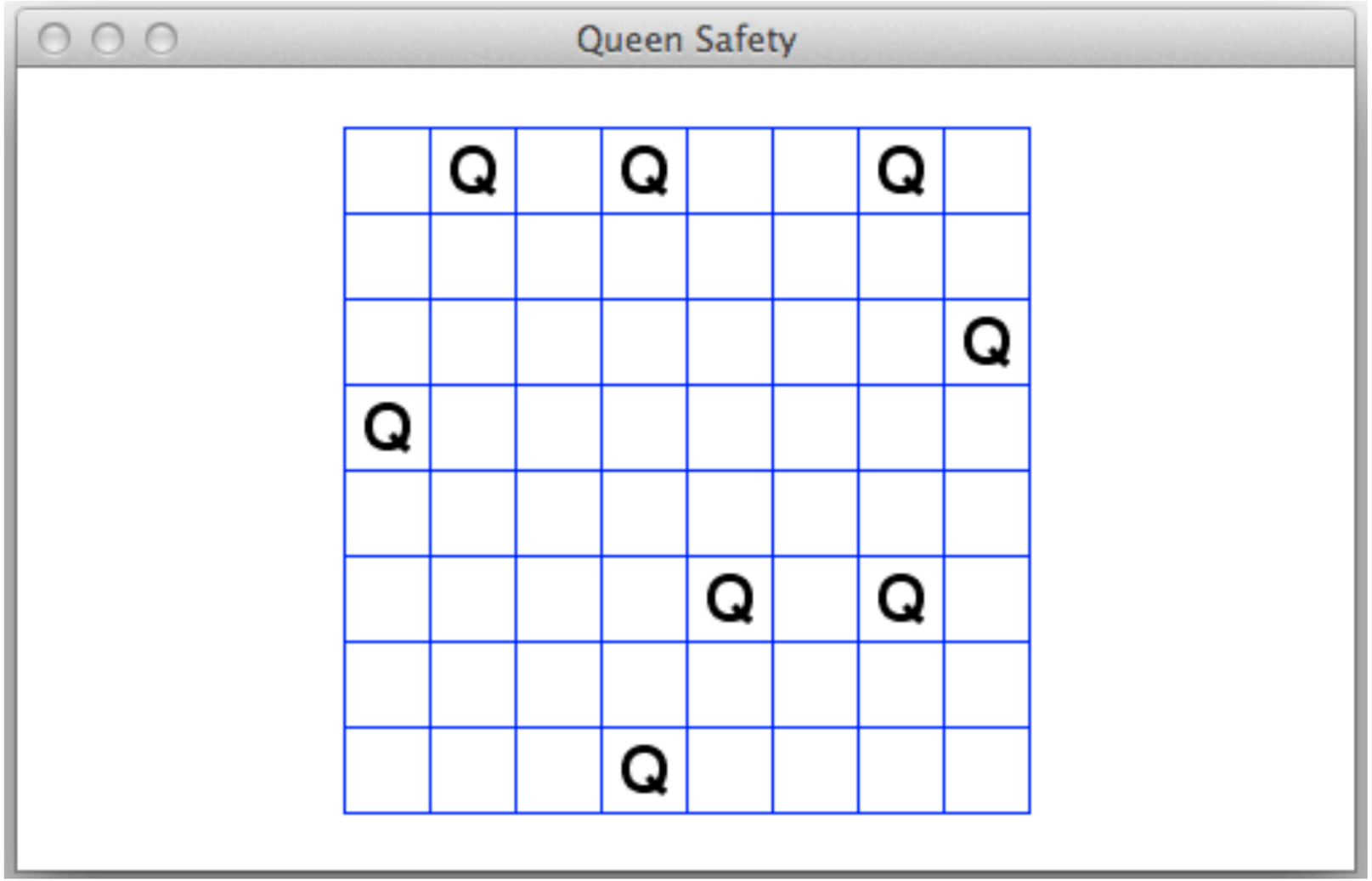
CS 106B

Programming Abstractions
Fall 2016
Stanford University
Computer Science Department

Announcements

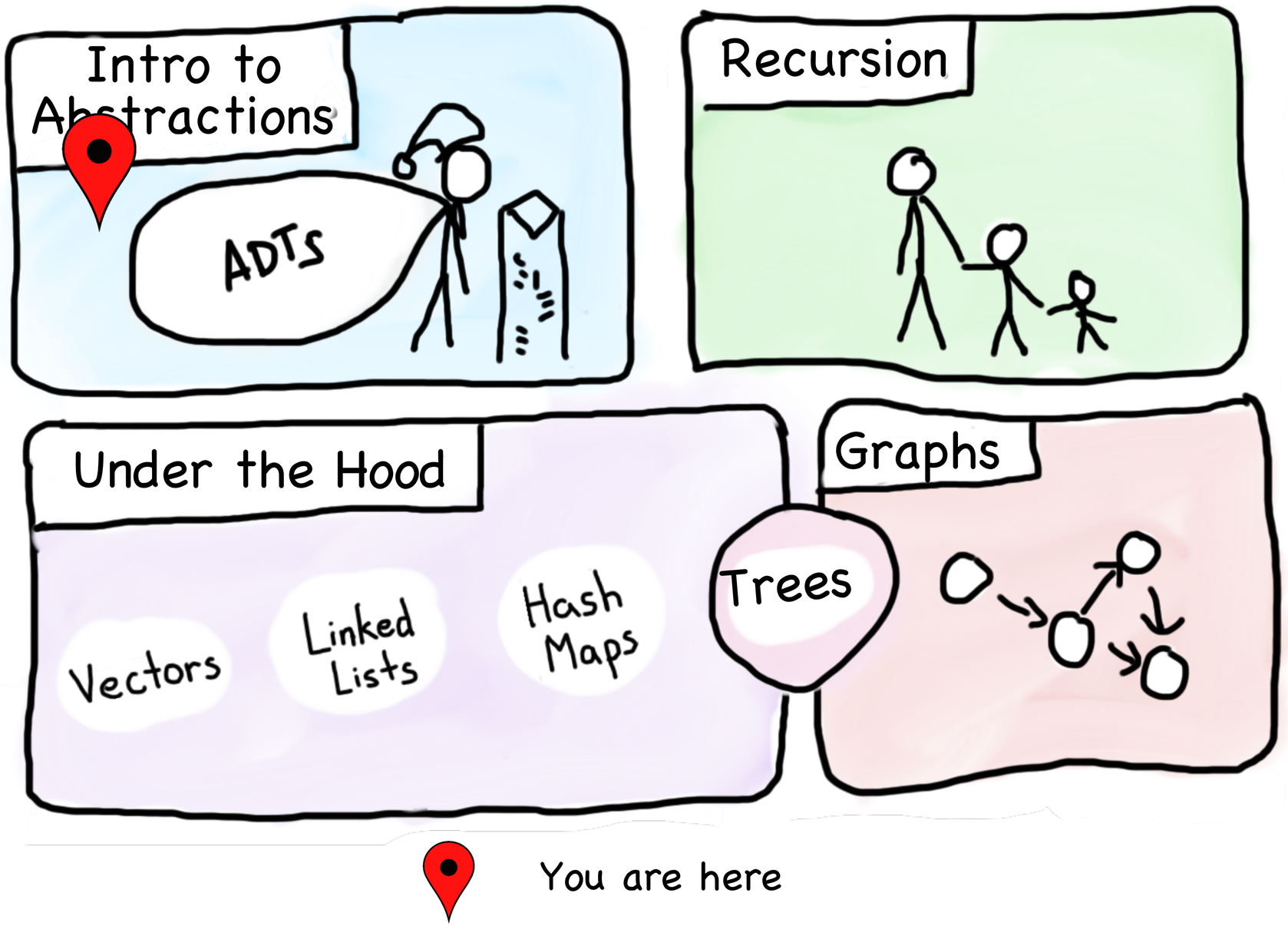
- Assignment 1 is due on Friday at 12:00 noon
- Sections start this Wednesday.
- LaIR is now open. 6pm to Midnight Sun – Thurs.
- Handout to go along with extra problem in lecture
- CS for Social Good

Something to Think About

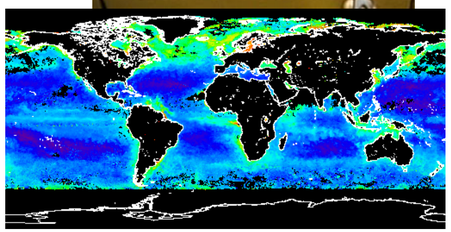


Write a function that returns if a particular square is “safe”

Course Syllabus



Processing Data



Almost all interesting programs process data.
Data comes from many sources:

- files on the local hard disk
- Databases
- downloaded over the internet
- input from hardware devices, e.g. sensors, microphone, gamepad

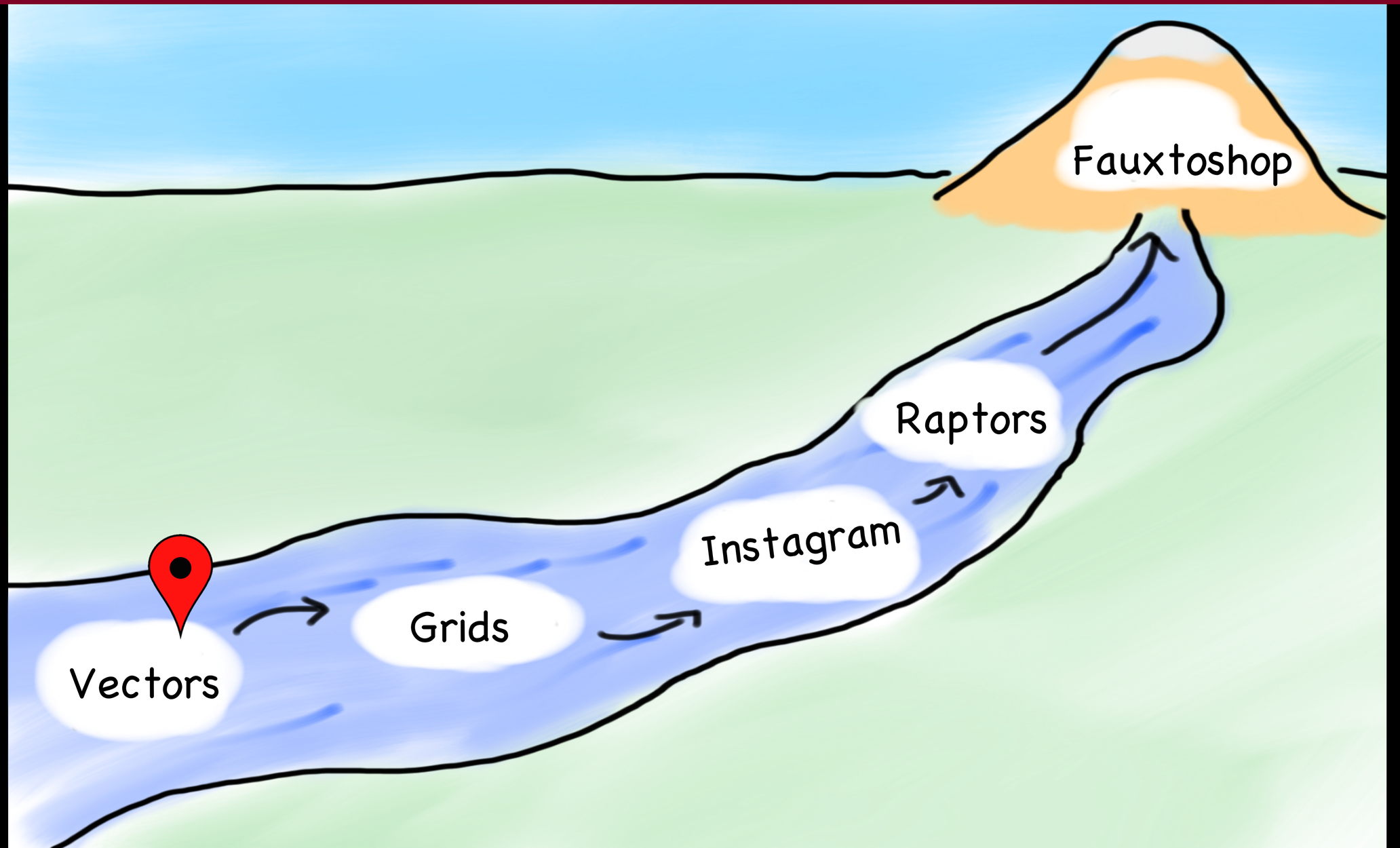
...

In order to effectively store and manage data and perform interesting calculations on it, we must learn about several useful collections (implementing using various data structures).

Today's Goals

1. Learn how to use Vectors
2. Learn how to use Grids
3. Be ready for Fauxtoshop

Today's Plan



Intro to Collections

Intro to Collections

Vector

Grid

Map

Stack

Queue

Set

Vector<type>

What is it?

- ArrayList<*type*>
- A list of elements that can grow and shrink.
- Each element has a place (or index) in the list.
- Advanced array.

Important Details

- Constructor creates an empty list.
- Bounds checks.
- Knows its size.
- Include “vector.h”

Why not use arrays?

Vector Creation

```
Vector<int> vec;
```

or

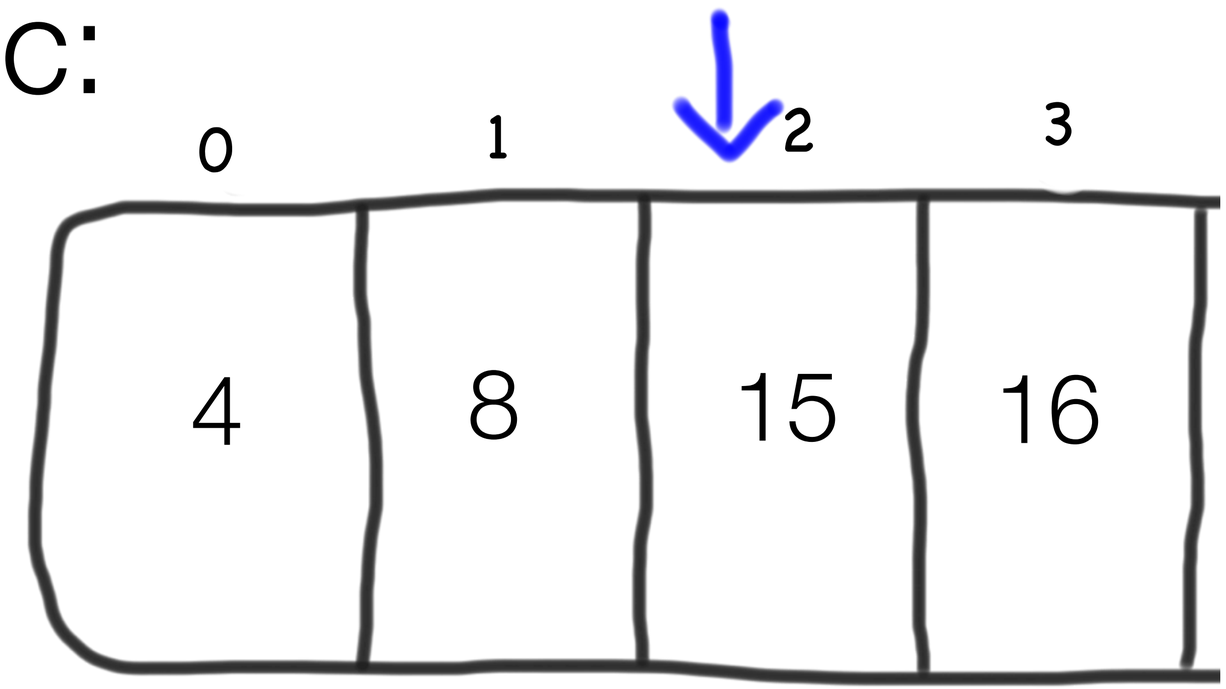
```
Vector<int> vec();
```

You must specify the type of your vector.
When a vector is created it is initially empty.

Vector Example

```
Vector<int> magic;
magic.add(4);
magic.add(8);
magic.add(15);
magic.add(16);
cout << magic[2] << endl;
```

magic:

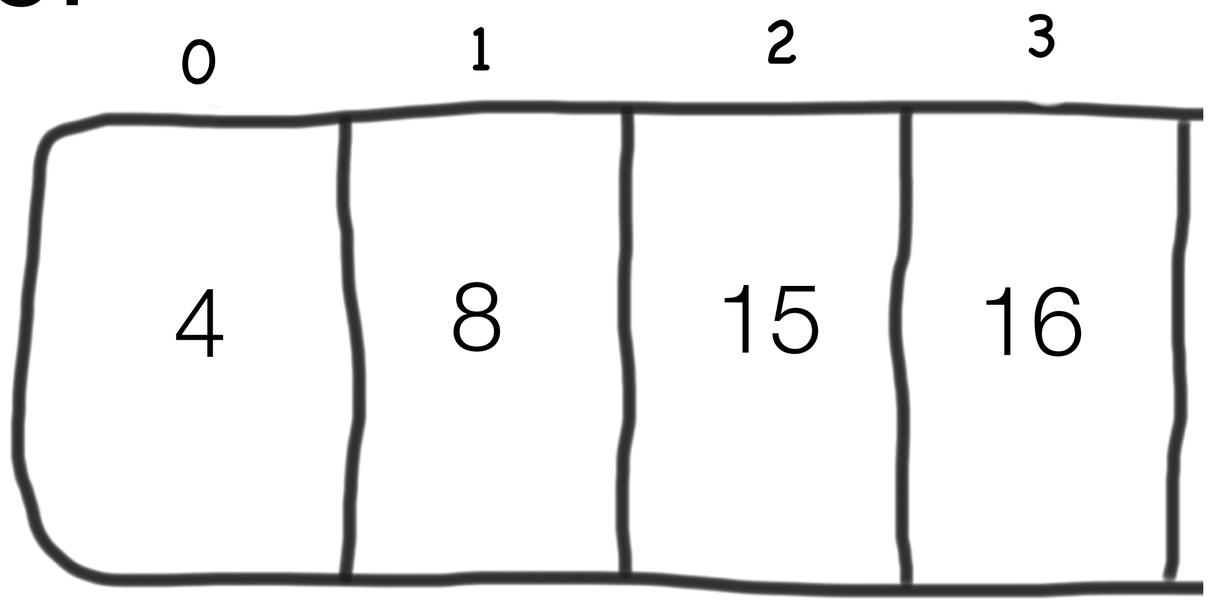


Vector Example

4

```
for(int i = 0; i < magic.size(); i++) {
    cout << magic[i];
}
```

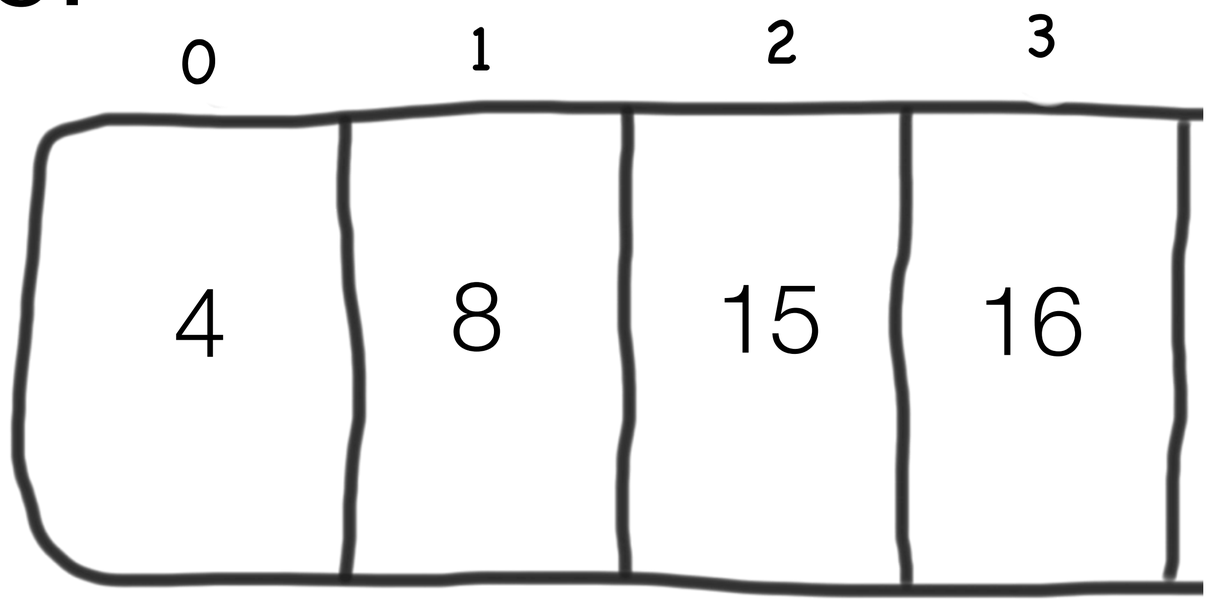
magic:



Equivalently

```
for(int value : magic) {
    cout << value;
}
```

magic:

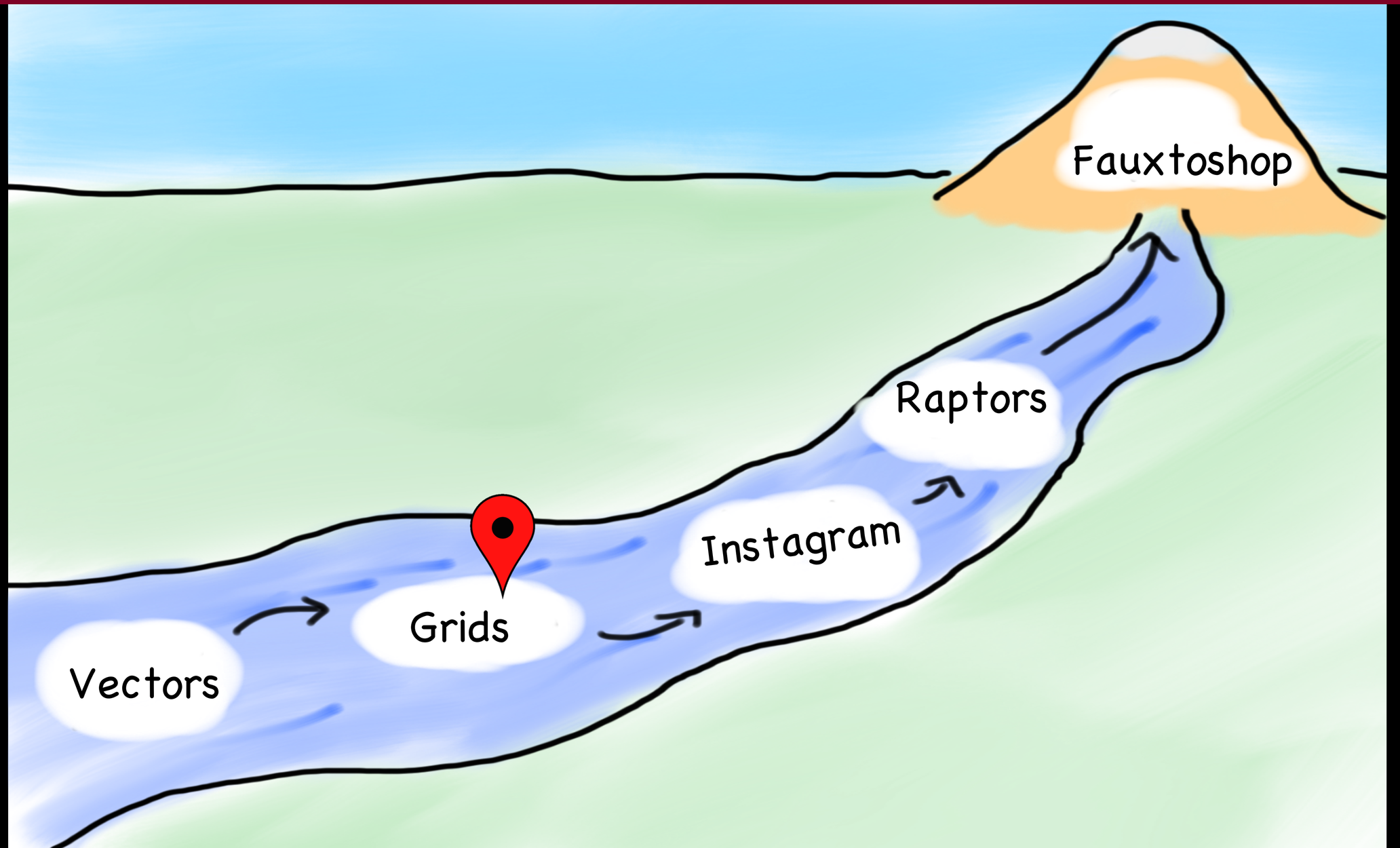


Vector Methods

`vec.size()` Returns the number of elements in the vector.
`vec.isEmpty()` Returns `true` if the vector is empty.
`vec[i]` Selects the i^{th} element of the vector.
`vec.add(value)` Adds a new element to the end of the vector.
`vec.insert(index, value)` Inserts the value before the specified index position.
`vec.remove(index)` Removes the element at the specified index.
`vec.clear()` Removes all elements from the vector.

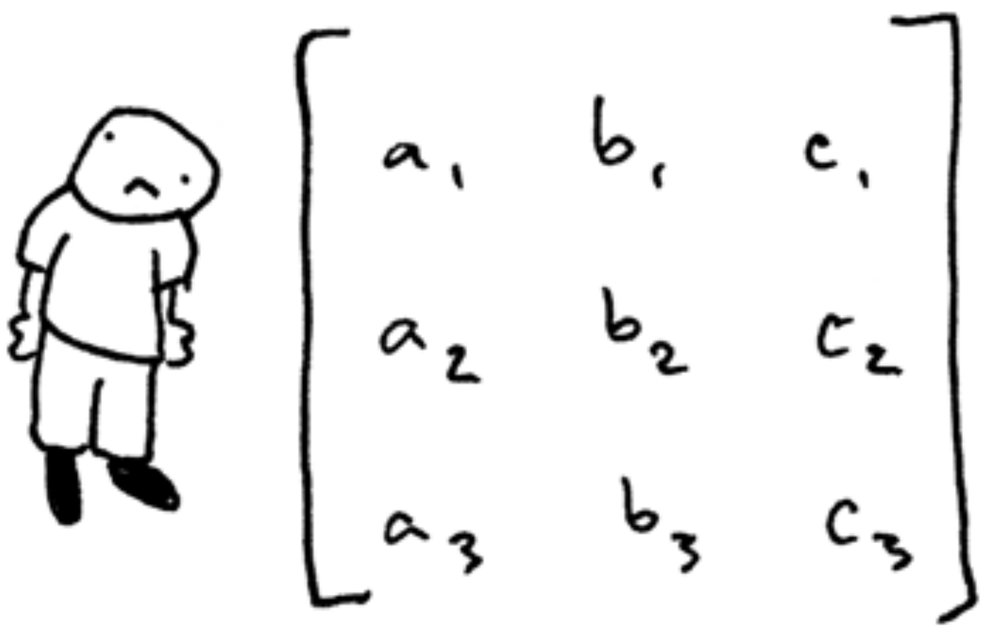
For the exhaustive list check out http://stanford.edu/~stepp/cppdoc/Vector-class.html

Today's Plan



Grid<type>

Grid<type>


$\begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix}$
WELCOME ..... TO
THE MATRIX!!!!!!

Grid Overview

What is it?

- Advanced 2D array.
- Think spread sheets, game boards

Important Details

- Default constructor makes a grid of size 0
- Doesn’t support “ragged right”.
- Bounds checks
- Knows its size.

Grid Creation

```
Grid<string> grid;
```

or

```
Grid<string> grid(3, 4);
```

Grid Example

```
Grid<int> aziz(2,2);
aziz[0][0] = 42;
aziz[0][1] = 6;
aziz[1][0] = aziz[0][1];
cout << aziz.numRows() << endl;
cout << aziz[0][1] << endl;
cout << aziz[1][1] << endl;
cout << aziz[2][3] << endl;
```

Grid Example

```
Grid<int> aziz(2,2);
aziz[0][0] = 42;
aziz[0][1] = 6;
aziz[1][0] = aziz[0][1];
cout << aziz.numRows() << endl;
cout << aziz[0][1] << endl;
cout << aziz[1][1] << endl;
Cout << aziz[2][3] << endl;
```

	0	1
0	42	6
1	6	0

2

```
***
*** STANFORD C++ LIBRARY
*** An ErrorException occurred during program execution:
*** Grid::operator [][]: (3, 2) is outside of valid range [(0, 0)..(2, 1)]
***
```

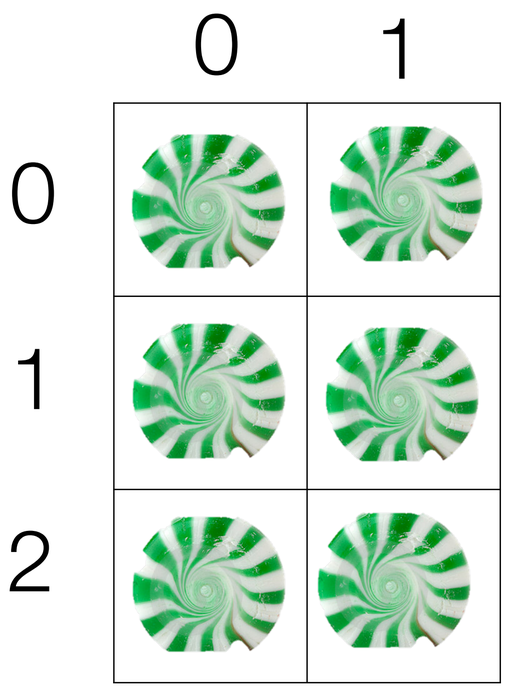
Grid Methods

`grid.numRows()` Returns the number of rows in the grid.
`grid.numCols()` Returns the number of columns in the grid.
`grid[i][j]` Selects the element in the `i`th row and `j`th column.
`grid.resize(rows, cols)` Changes the dimensions of the grid and clears any previous contents.
`grid.inBounds(row, col)` Returns `true` if the specified row , column position is within the grid.

For the exhaustive list check out http://stanford.edu/~stepp/cppdoc/Grid-class.html

Grid Example

```
void printGrid(Grid<Candy> & grid) {
    for(int r = 0; r < grid.numRows(); r++) {
        for(int c = 0; c < grid.numCols(); c++) {
            giveCandy(grid[r][c]);
        }
    }
}
```



Collections

1. Defined as Classes
   This means they have constructors and member functions

2. Templatized
   They have a mechanism for collecting different variable types

3. Deep copy assignment
   Often pass them by reference!

Common Pitfalls 1

```
Vector numbers;
```

Common Pitfalls 1

```
Vector<int> numbers;
```

Common Pitfalls 2

```
void myFunction(Grid<bool> gridParam);
```

Common Pitfalls 2

```
void myFunction(Grid<bool> & gridParam);
```

Common Pitfalls 3

```
void cout(Grid<bool> & grid) {
    for(int i = 0; i < grid.numRows(); i++) {
        for(int j = 0; j < grid.numCols(); j++) {
            cout << grid[j][i];
        }
    }
}
```

Common Pitfalls 3

```
void cout(Grid<bool> & grid) {
    for(int r = 0; r < grid.numRows(); r++) {
        for(int c = 0; c < grid.numCols(); c++) {
            cout << grid[r][c];
        }
    }
}
```

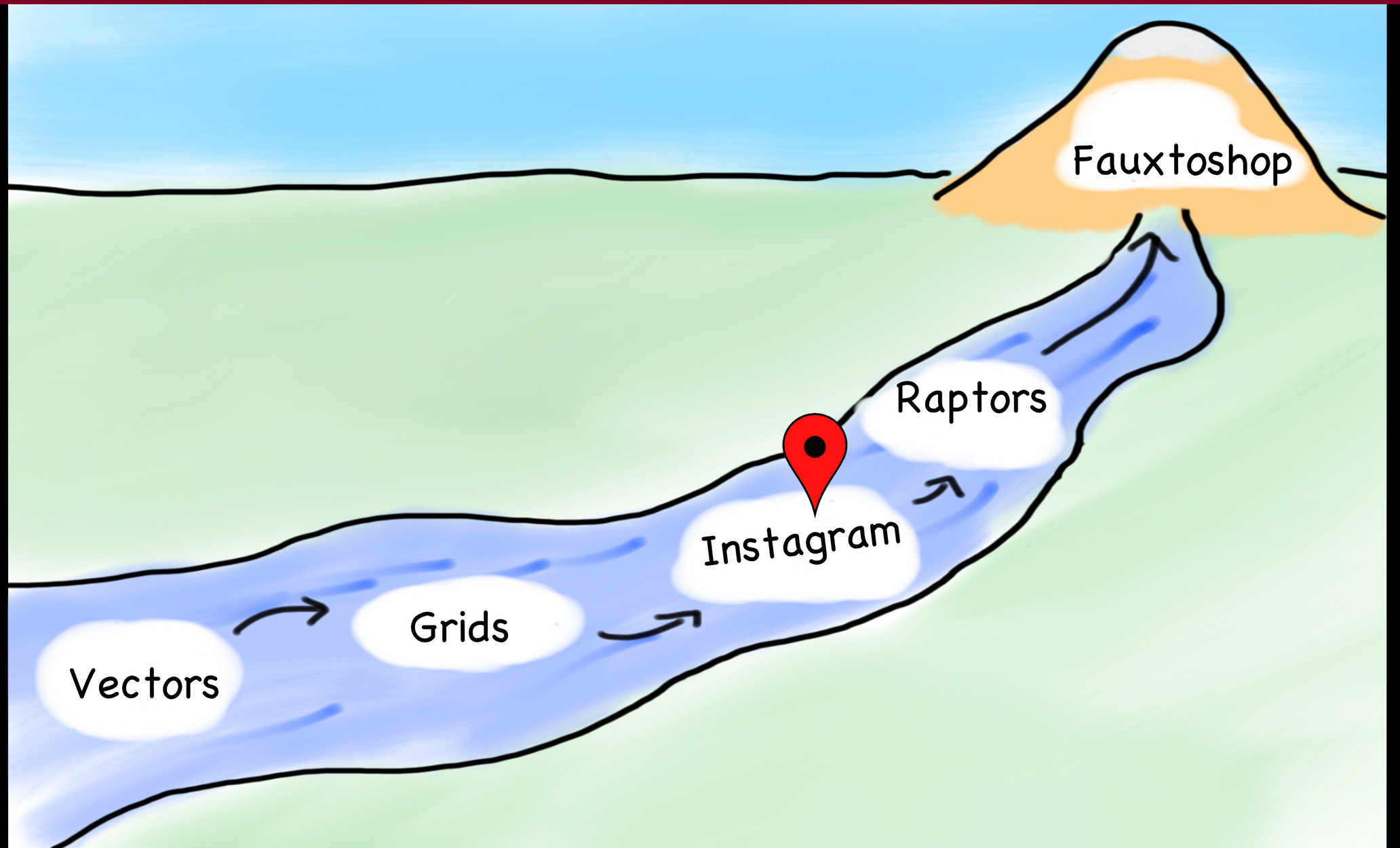
Common Pitfall?

```
Vector<Vector<int>> myVec;
```

Not Anymore!

```
Vector<Vector<int> > myVec;
```

Today's Plan

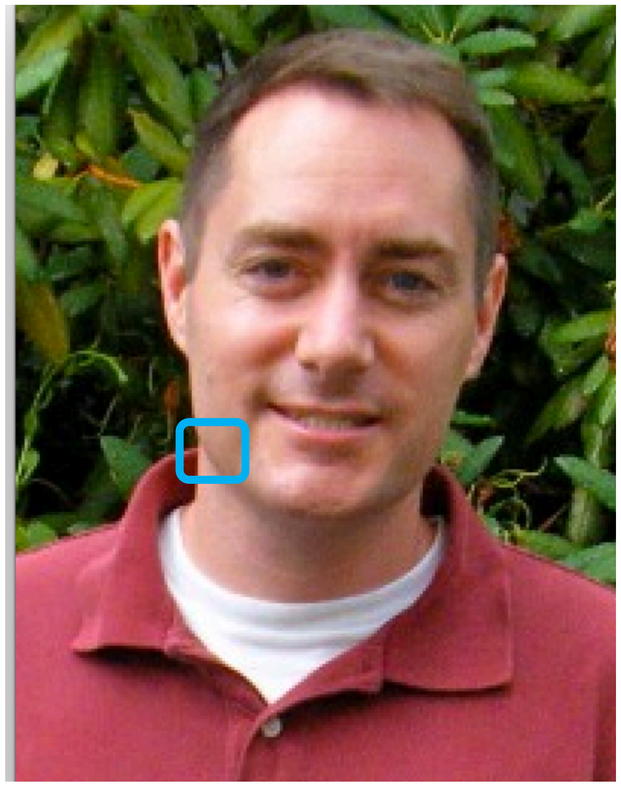


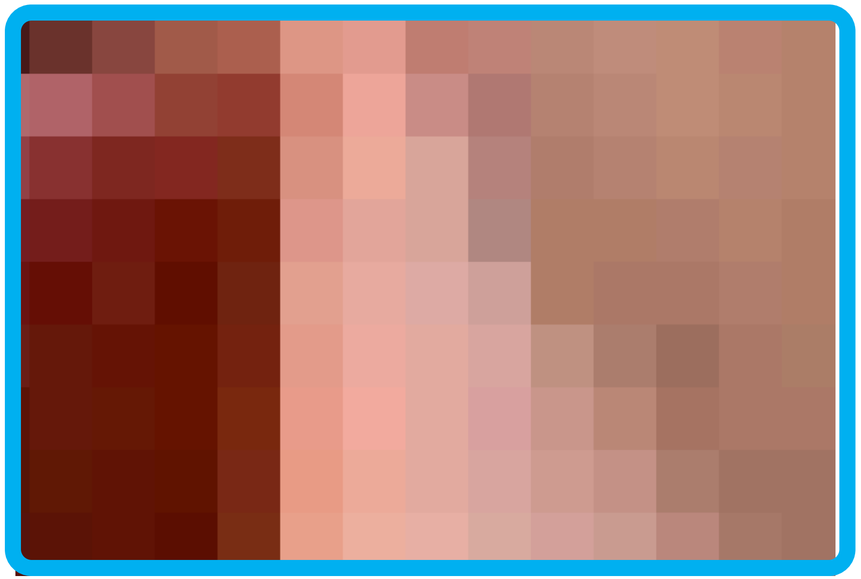
Mike Krieger
From FroSoCo

Problem 1: Instagram

Problem 1: Instagram

Problem 1: Instagram



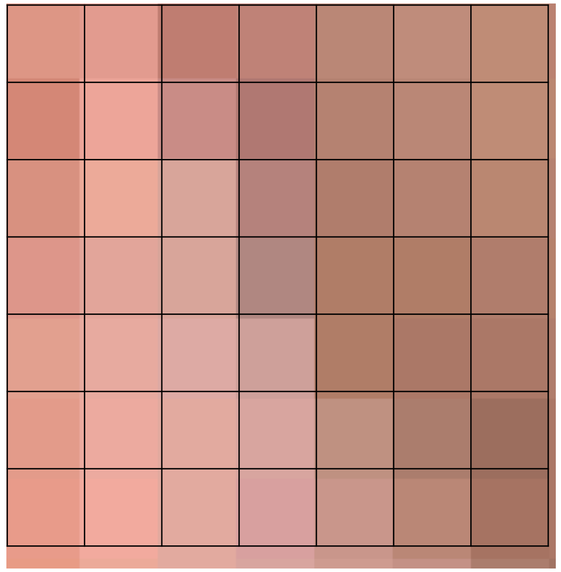


Color is an int!

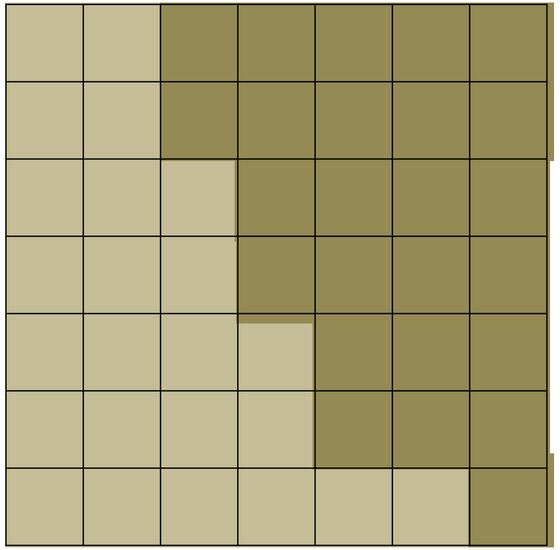
An image is really just a Grid<int>

Problem 1: Instagram

Original



Filtered



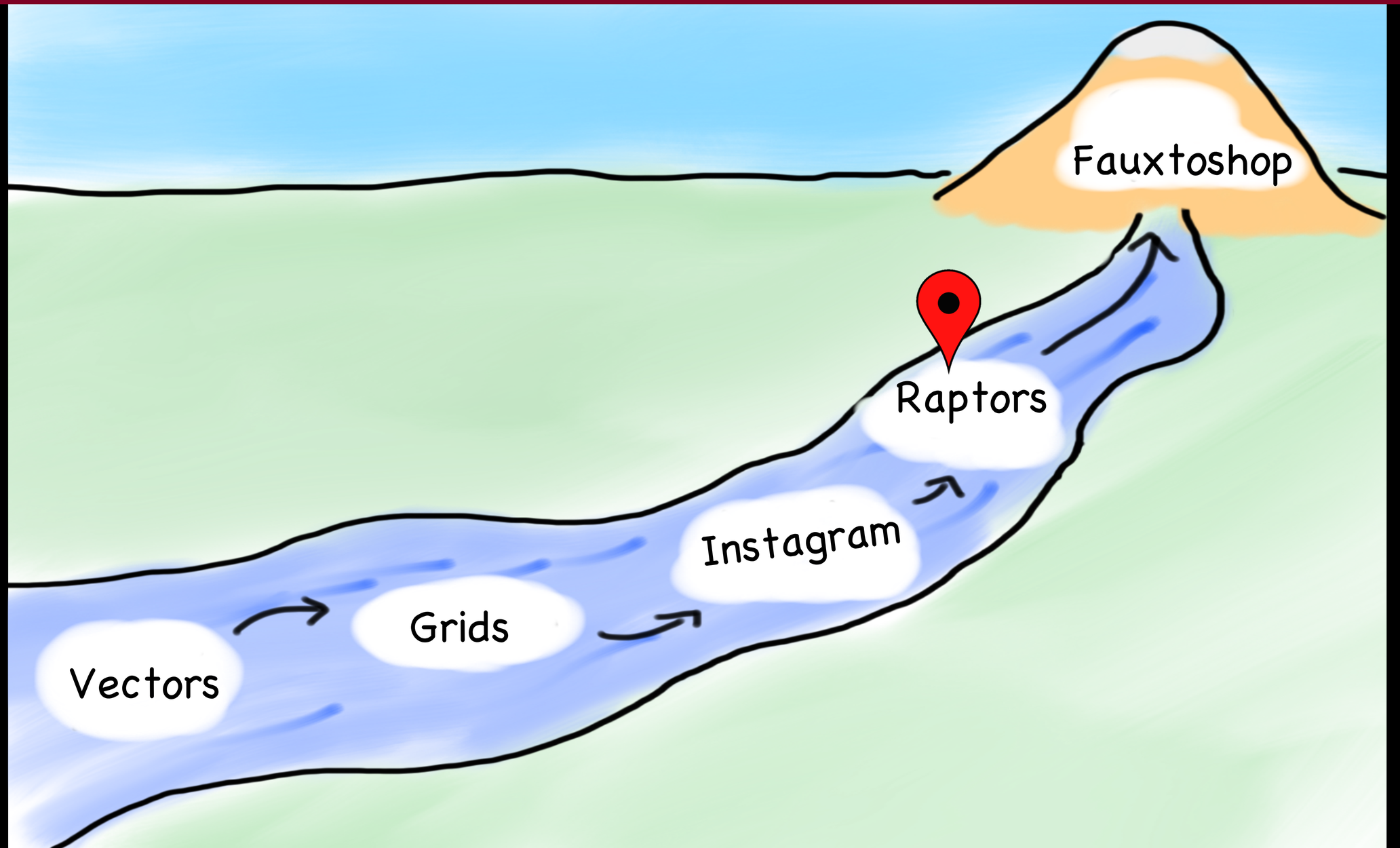
Palette

Let's Code

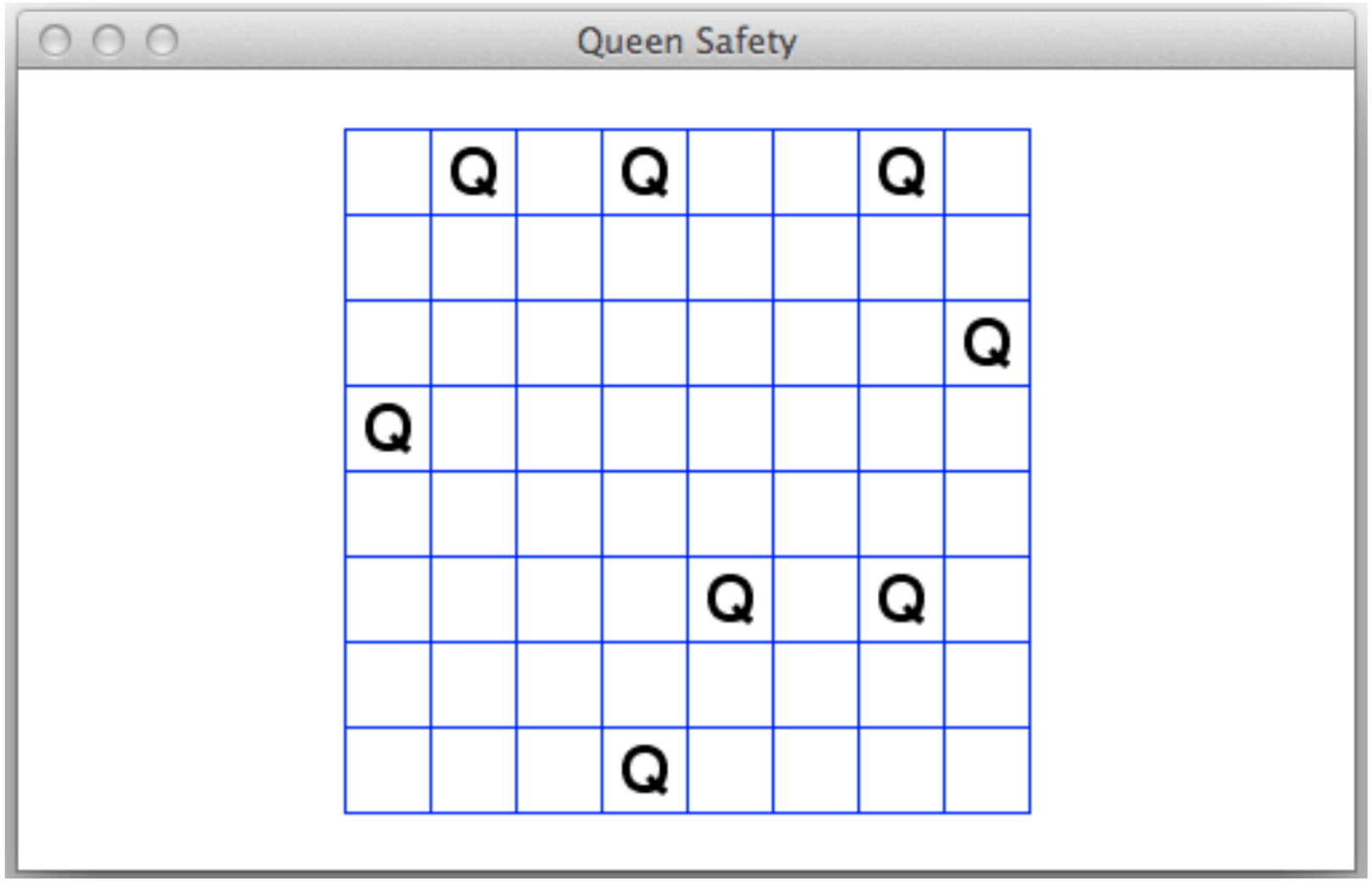

Qt


LELAND STANFORD JUNIOR UNIVERSITY
DIE LUFT DER FREIHEIT WEHT
1891

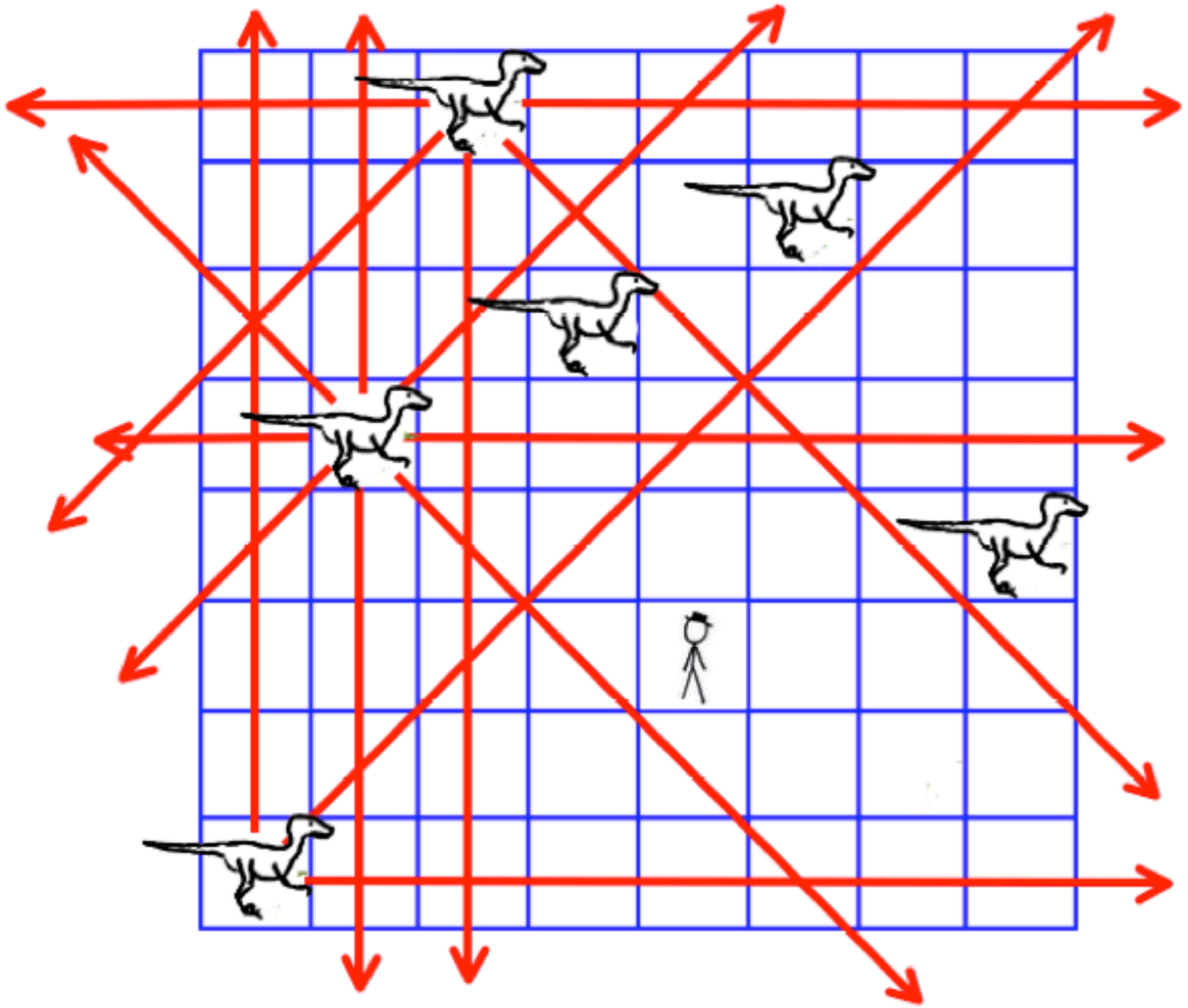
Today's Plan

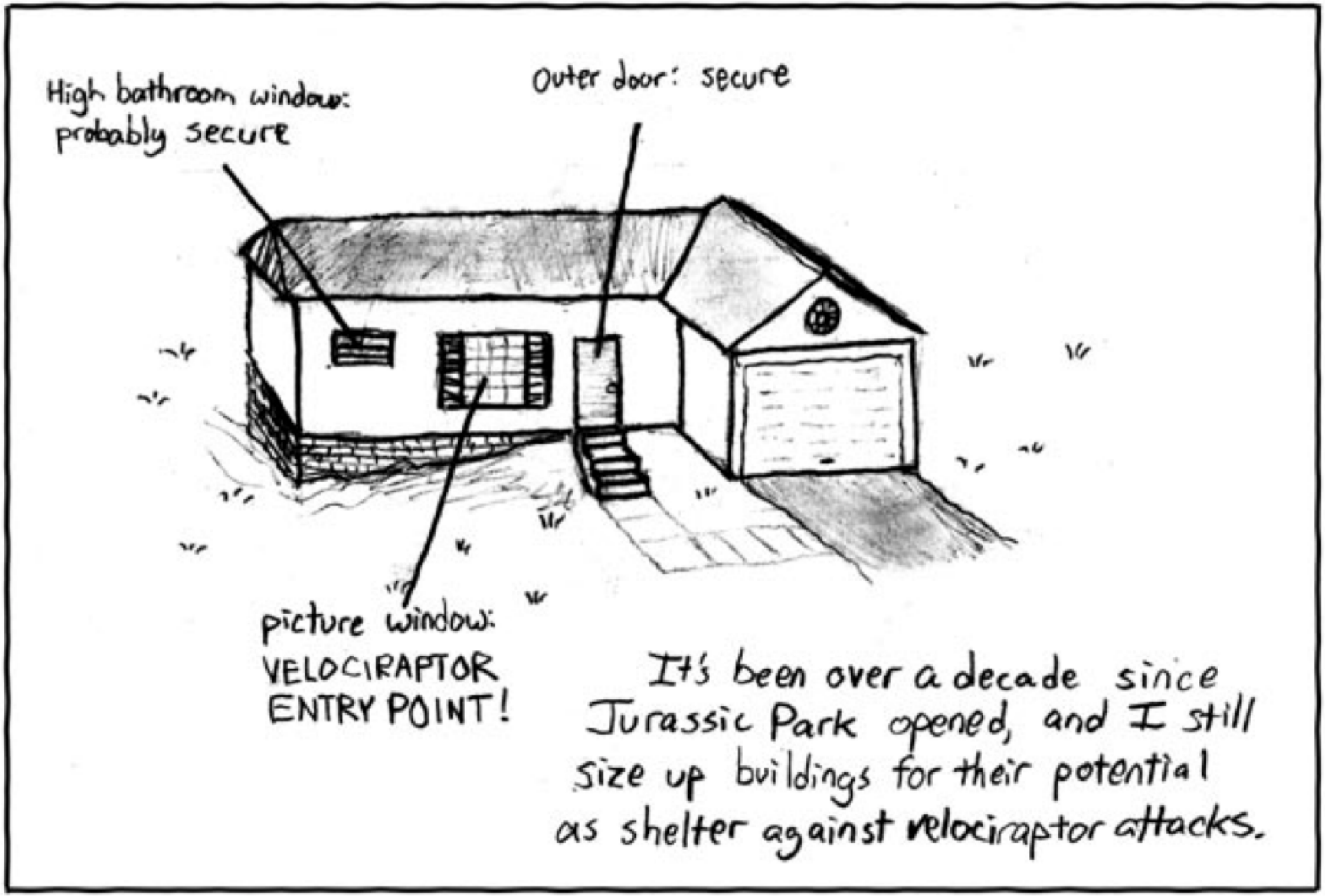


Problem 2: Queen Safety

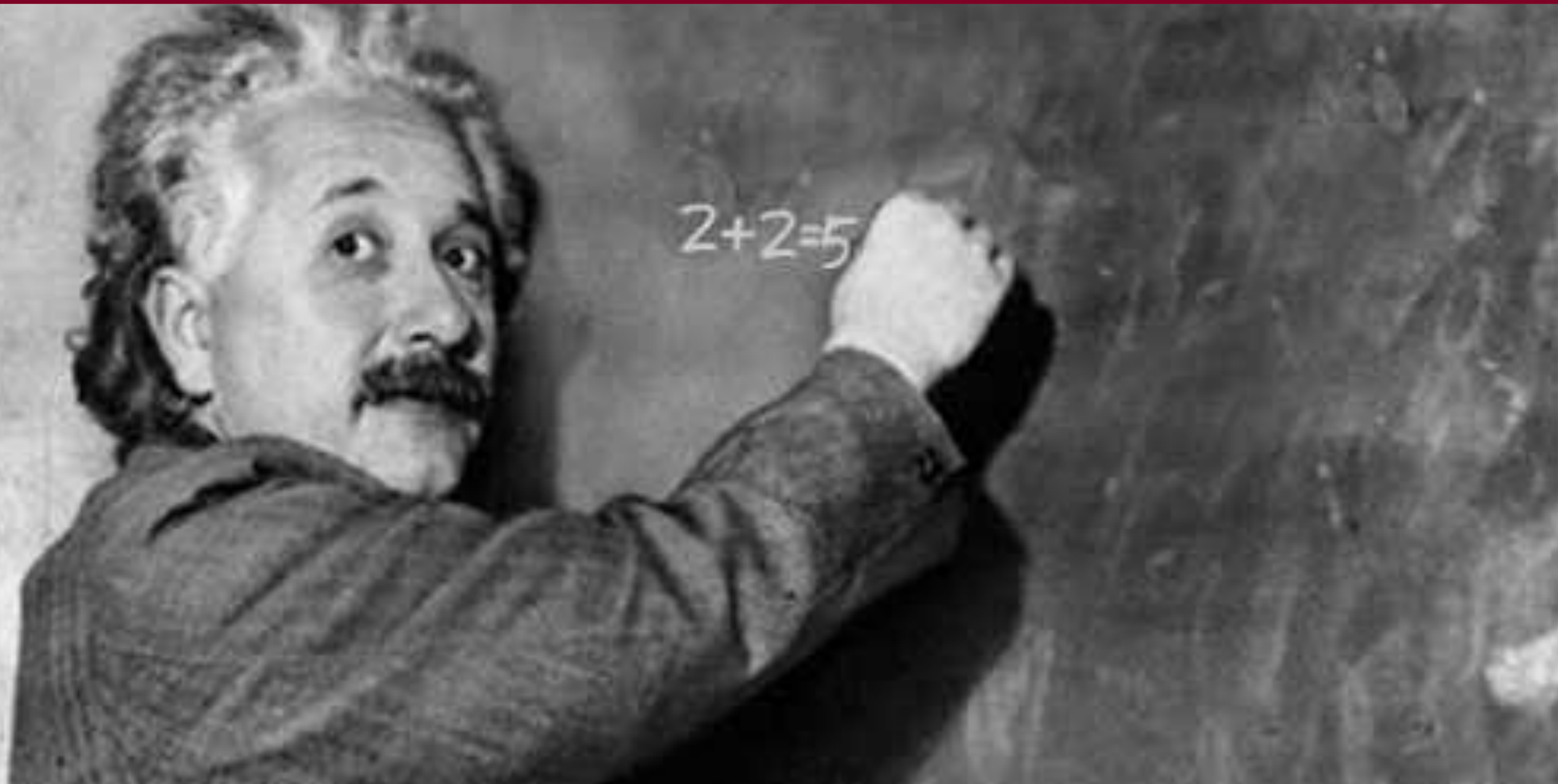


Problem 2: Velociraptor Safety

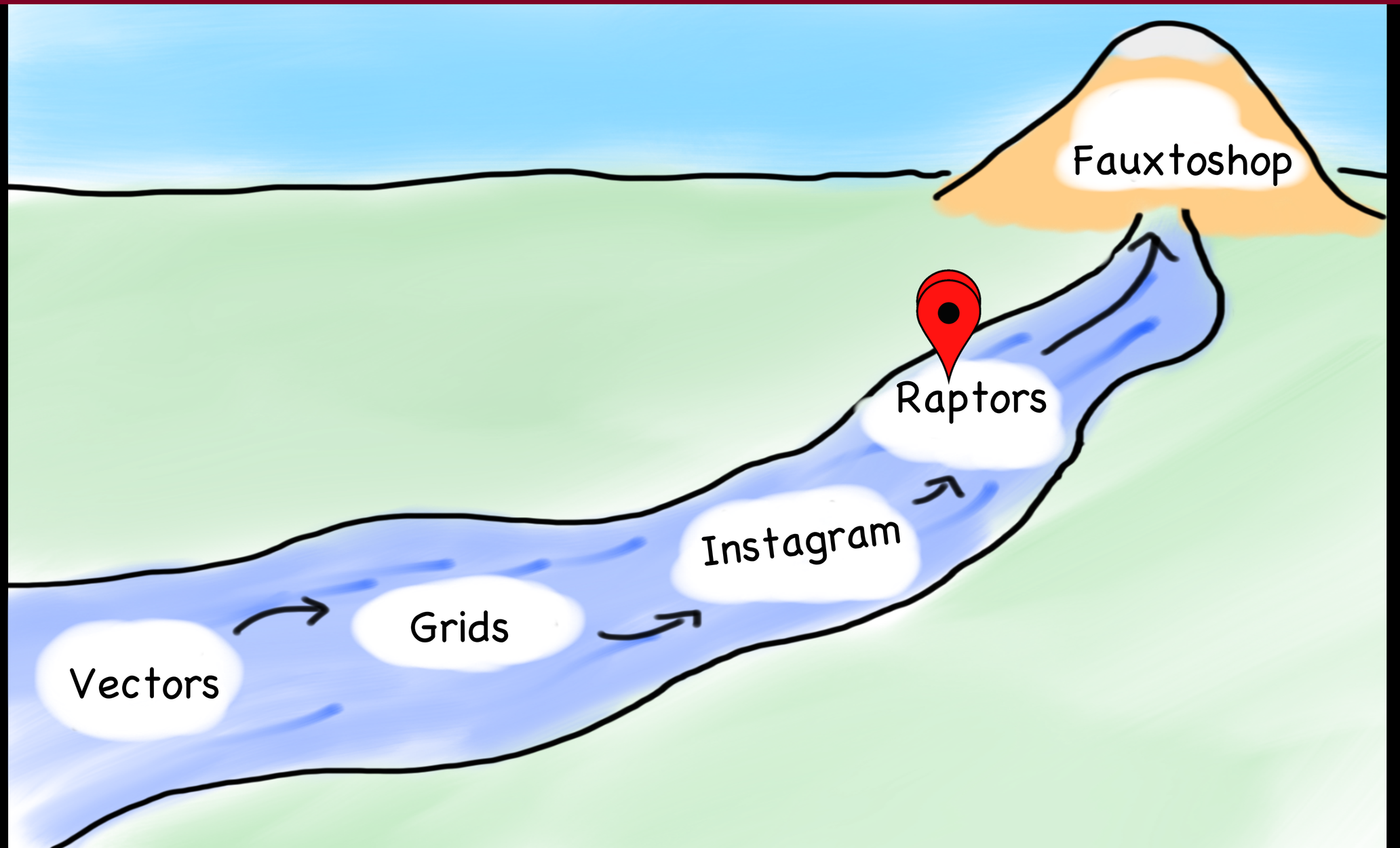



High bathroom window: probably secure
Outer door: secure
picture window: VELOCIRAPTOR ENTRY POINT!
It's been over a decade since Jurassic Park opened, and I still size up buildings for their potential as shelter against velociraptor attacks.

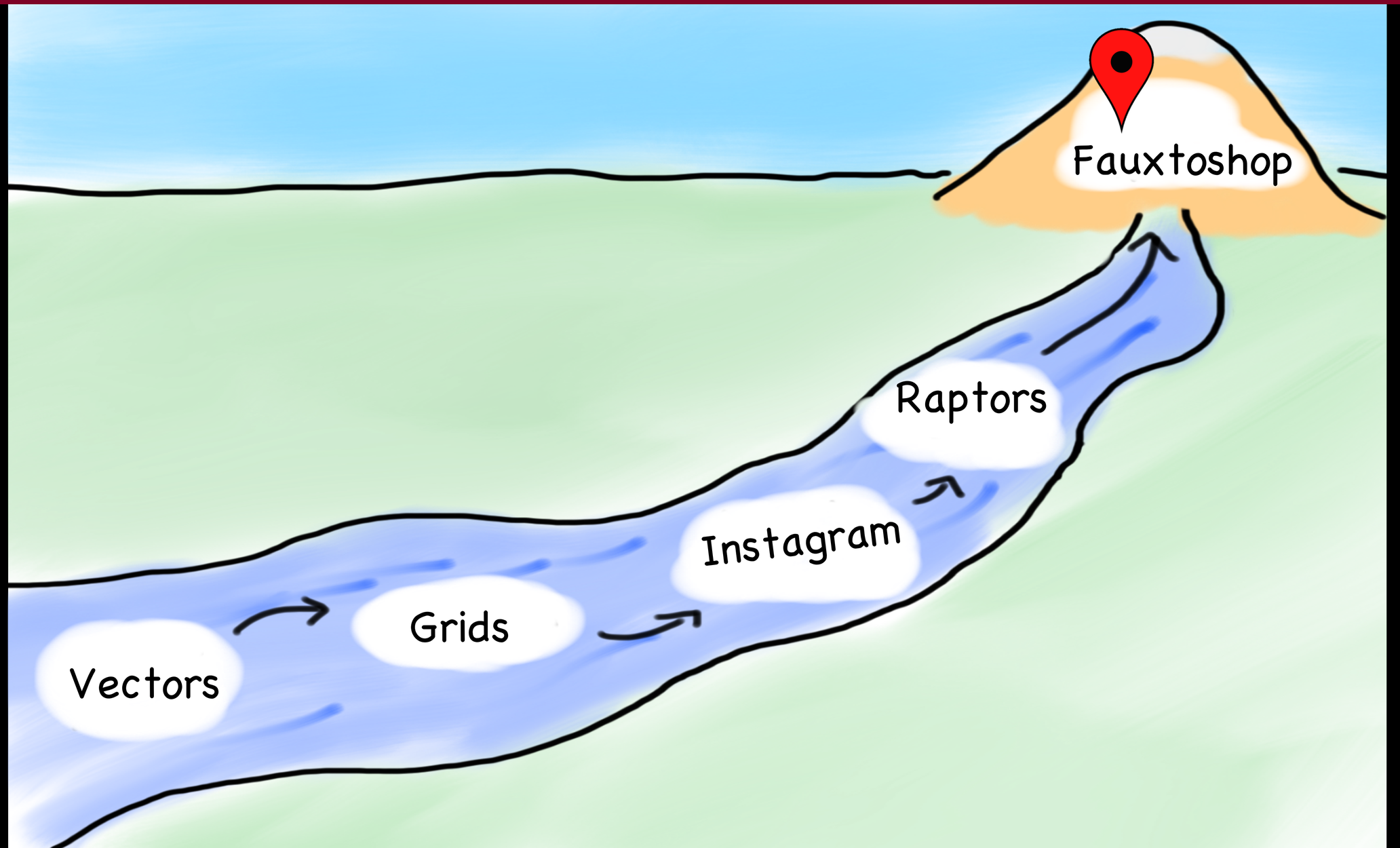
Life Skills on the Board



Today's Plan



Today's Plan



Today's Goals

1. Learn how to use Vectors
2. Learn how to use Grids
3. Be ready for Fauxtoshop

Recap

Vectors and Grids

- Container classes that can store multiple values.
- Vector is an expanding list of values.
- A grid is a 2D matrix of values
- They are classes and have useful member functions.
- They are templatized, meaning you need to specify the type of values.

Instagram and Velociraptor Safety

- Velociraptors are scary. Stay safe!
- The pixels of a GBufferedImage are stored using a Grid<int>
- There are several ways of iterating over a Grid.

Advanced Reading

- Dynamic array: https://en.wikipedia.org/wiki/Dynamic_array
- Sparse matrix: https://en.wikipedia.org/wiki/Sparse_matrix
- Linked lists: https://en.wikipedia.org/wiki/Linked_list