Look, up in the sky! It's a bird! It's a plane!

It's an unmanned drone helicopter shooting a taco from space down at you and your colleagues during lunchtime!

The Internet is going wild for Tacocopter, perhaps the next great startup out of Silicon Valley, which boasts a business plan that combines four of the most prominent touchstones of modern America: tacos, helicopters, robots and laziness.

Indeed, the concept behind Tacocopter is very simple, and very American: You order tacos on your smartphone and also beam in your GPS location information. Your order -- and your location -- are transmitted to an unmanned drone helicopter (grounded, near the kitchen where the tacos are made), and the tacocopter is then sent out with your food to find you and deliver your tacos wherever you're standing.
Lab Midterms Not Graded Yet

SOON?

SOON
Chapter 8 in book: drawing stuff

Questions?

http://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html
Collections

Collection<MovingCircles> circles;
circles = new ArrayList<MovingCircles>();
MovingCircle c = new MovingCircle( ... );
circles.add( c );

for( MovingCircle c : circles )
{
    c.moveIt();
}
Repainting

myCanvas.setAutoRepaint( false );

myCanvas.repaint();
Arrays of Arrays

An array can represent a collection of any type of object - including other arrays!

The world is filled with examples

• Monthly magazine: we number
  – the monthly editions
  – pages with in each

• Calendars: we number
  – the months
  – days in each month
Creating an Array of Arrays

Array declaration introduces a name, but does not create an array

1. **Declare the Array**
2. **Create the array:**
   - Proceed in two steps
     1. Construct 12-element year
     2. Construct each individual month array
3. **Put values into the array**
1. Construct 12-element year

dailyEvent = new String[12] [ ]
2. Construct months

```java
for (int month = 0; month < 12; month++) {
    int numDays = getDays( month+1 );
    dailyEvent[month] = new String[numDays];
}
```

Assume `getDays` is a private method that returns the number of days in a month.
Indexing an Array of Arrays

Say a user enters the information

1/28 - Spring semester starts

The month is 1
The day is 28
The event is “Spring semester starts”

Since array indexing begins at 0,

dailyEvent[0][27] = “Spring semester starts”;

Traversing a 2-D Array

Do something with every element in an array: *Use for loops!*

- Ex. Initialize all calendar entries to “No event today”
  - to initialize all calendar entries for a single month:
    // Fill all entries for one month with “No event today”
    for (int day = 0; day < dailyEvent[month].length; day++) {
      dailyEvent[month][day] = “No event today”;
    }

- to initialize all 12 months
  // Fill all entries in each month with “No event today”
  for (int month = 0; month < 12; month++) {
    // Fill all entries for one month with “No event today”
    ...
// Fill all entries in each month with “No event today”
for (int month = 0; month < 12; month++) {
    // Fill all entries for one month with “No event today”
    for (int day = 0; day < dailyEvent[month].length; day++) {
        dailyEvent[month][day] = “No event today”;
    }
}

Putting it all Together
Matrices

- two dimensional arrays with rows of same length
  Ex. magnified region of pixels from an image

Each pixel can be described by row and column position, as well as color value
Beyond Two Dimensions
More Practice with Matrices: Digital Image Smoothing

- A digital photo is a matrix of pixel values
- Noise and other artifacts sometimes appear
- can *smooth* an image to eliminate (or decrease) some of these artifacts
A Simple Smoothing Algorithm

• Assume image is black and white
  – pixel values are ints giving brightness levels
  ```java
  private int[][] brightness;
  ```
• Implement smoothing by averaging
  – replace each pixel value with the average of its neighbors
Design of Smoothing Algorithm

- Assume constants WIDTH and HEIGHT give image width and height in pixels
- Declare a matrix for the averaged (smoothed) image

```java
private int[][] avgBrightness;
```
- Traverse row by row

```java
for (int row = 1; row < HEIGHT-1; row++) {
    for (int col = 1; col < WIDTH-1; col++)
        // replace brightness value by average value
        ...
}
```
Calculating Average Brightness

Consider a 3-by-3 square centered at the pixel to be replaced by an average

// Calculate average brightness at position specified by row and col;
// position is not on an edge of the image.
int totalBrightness = 0;
for (int r = -1; r <= 1; r++) {
    for (int c = -1; c <= 1; c++) {
        totalBrightness = totalBrightness + brightness[row+r][col+c];
    }
    avgBrightness[row][col] = totalBrightness/9;
}
Pixels on the Edge

What about pixels on the edges of an image?

– pixels on top edge have no neighbors above them; only to the sides and below.

– pixels on bottom have no neighbors below them; only to the sides and above.

– pixels in the corners have even fewer neighbors.
Handling the Top Edge

// Calculate average brightness at position specified by row and col;
// position is top edge of the image, not in a corner
int totalBrightness = 0;
for (int r = 0; r <=1; r++) {
    for (int c = -1; c <= 1; c++) {
        totalBrightness = totalBrightness + brightness[row+r][col+c];
    }
    avgBrightness[row][col] = totalBrightness/6;
}
Handling the Bottom Edge

// Calculate average brightness at position specified by row and col;
// position is bottom edge of the image, not in a corner
int totalBrightness = 0
for (int r = -1; r <= 0; r++) {
    for (int c = -1; c <= 1; c++) {
        totalBrightness = totalBrightness + brightness[row+r][col+c];
    }
    avgBrightness[row][col] = totalBrightness/6;
}
Generalizing

• Similar loops to handle: left border, right border, corners
• Difference is # neighbors found above, below, to the right, and to the left

// calculate average brightness at position specified by row and col;  
// distLeft, distRight, distUp, & DistDown specify neighborhood  
private int adjustPixel( int row, int col, int distLeft, int distRight, int distUp, int distDown ) {
    int totalBrightness = 0;  
    int neighbors = 0;  
    for (int r = distUp; r <= distDown; r++) {
        for (int c = distLeft; c <= distRight; c++) {
            totalBrightness += brightness[row+r][col+c];  
            neighbors++;  
        }
    }
    return totalBrightness/neighbors;  
}