Expressive Data
Ira Greenberg

As a painter, artist Ira Greenberg studied nature, visually searching for patterns and forms that he translated into expressive strokes of paint. As a creative coder, he continues to search, but within motifs that extend beyond the physical, natural world. In this talk, Ira will present an overview of his creative journey, visually telling the story of how his medium transmuted from paint to code. He will also feature some of his recent work, including Protobytes (studies in artificial life) and visualization in the digital humanities.

**Time:** Wednesday 9/28 12:30-2pm  
**Place:** Thomas Library 224 (refreshments served)
Review

- Arrays – Declaring and Using
- Built-in Array Functions
- Arrays and loops
- Converting single variable-based programs to array-based programs
- Objects
- Object-Oriented Programming (OOP)
- Objects
  - Fields (Variables)
  - Methods (Functions)
- PImage Object
  - Fields: width, height, pixels[], ...
  - Methods: loadPixels(), updatePixels(), get(x, y), save(path), ...
- String Object
  - Fields: ...
  - Methods: length(), toUpperCase(), ...
Defining Your Own Object with Classes

• Classes are blueprints or prototypes for new objects

• Classes encapsulate all field and method declarations

  ... which are repeated for each new object created

• Using a class to create a new object is called instantiating an object

  ... creating a new object instance of the class

• Classes often model real-world items
Defining Your Own Objects with Classes

// Defining a new class of object

class MyObjectName {

    // All field variable declarations go here;

    // Define a special function-like statement called
    // the class's Constructor.
    // It's name is same as object class name,
    // with no return value.

    MyObjectName( optional arguments ) {

        // Perform all initialization here

    }

    // Declare all method functions here.

}
// A Ball Class

class Ball {
    // Fields
    float ay = 0.2; // y acceleration (gravity)
    float sx;     // x position
    float sy;     // y position
    float vx;     // x velocity
    float vy;     // y velocity

    // Constructor
    Ball() {
        sx = random(0.0, width);
        sy = random(0.0, 10.0);
        vx = random(-3.0, 3.0);
        vy = random(0.0, 5.0);
    }

    // Methods
    void update() {
        // Move ball
        sx += vx;
        sy += vy;
        vy += ay;

        // Bounce off walls and floor
        if (sx <= 10.0 || sx >= (width-10.0)) vx = -vx;
        if (sy >= (height-10.0) && vy > 0.0) vy = -0.9*vy;
    }

    void draw() {
        ellipse( sx, sy, 20, 20);
    }
}
Creating New Objects with Classes

• To create a new instance of an object, use the `new` keyword and call the object Constructor

```java
MyObjectName ob = new MyObjectName(42);
String s = new String("Blah");
String s = "Blah";
Ball b = new Ball();
```

Same result
Use the Ball class

Treat in a manner very similar to a primitive data type.

// bounce4
Ball[] balls = new Ball[20];  

void setup() {
  size(500, 500);
  fill(255, 0, 0);
  smooth();
  ellipseMode(CENTER);

  // Create all new Ball objects
  for (int i = 0; i < balls.length; i++) {
    balls[i] = new Ball();
  }
}

void draw() {
  background(255);

  for (int i = 0; i < balls.length; i++) {
    balls[i].update();
    balls[i].draw();
  }
}
Comparing Declarations and Initializers

```java
int i;
int j = 3;
float fac = 0.1;
float[] Xs;
float[] Ys = new float[10];
float[] Zs = new float[] {1.2, 2.3, 3.4};
String s1 = "abc";
String s2 = new String("abc");
String[] s3 = new String[50];
String[] s4 = new String[] {"moe", "larry", "curly"};
Ball b = new Ball();
Ball[] bs = new Ball[200];
```