INTRO TO PROCESSING

Hitesh Raju

University of Utah

slide acknowledgements:
http://processing.org/
administrivia . . .
- data exploration assignment due today!
- time series assignment out today
- open lab in class on Thurs.
SCI Visualization Journal Club

Meeting every Tuesday at 2:30pm is the Halvorsen Conference Room (WEB 4640)

This week’s paper: Illustrative Visualization of Molecular Reactions using Omniscient Intelligence and Passive Agents” from EuroVis 2014

For more information, e-mail Paul (prosen@sci.utah.edu) or visit http://www.sci.utah.edu/the-institute/events/vjc-fall2014.html
last time . . .
(human) visible light
Color != Wavelength

but rather, a combination of wavelengths and energy
terms

- hue (chroma)

- saturation (chromaticity)

- luminance (lightness / brightness / value)
perceptual color spaces

change in amount of a color value should produce an equivalent visual change.
size & color

“the smaller the mark, the less distinguishable are the colors”

-Jacques Bertin
simultaneous contrast
 categorical

- color is great for categorical quantities!
distinguishability

- only good at 6 - 12 simultaneous colors
today . . .
Processing
what is it?

- programming environment
- visually oriented applications
- targets artists, designers, etc.
what is it?

- Processing Development Environment (PDE)
what is it?

**Processing API**

**Reference.** The Processing Language was designed to facilitate the creation of sophisticated visual structures.

<table>
<thead>
<tr>
<th>Structure</th>
<th>Shape</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>( ) (parentheses)</td>
<td>createShape()</td>
<td>Setting</td>
</tr>
<tr>
<td>, (comma)</td>
<td>loadShape()</td>
<td>background()</td>
</tr>
<tr>
<td>. (dot)</td>
<td>PShape</td>
<td>clear()</td>
</tr>
<tr>
<td>/* */ (multiline comment)</td>
<td>2D Primitives</td>
<td>colorMode()</td>
</tr>
<tr>
<td>/** */ (doc comment)</td>
<td>arc()</td>
<td>fill()</td>
</tr>
<tr>
<td>// (comment)</td>
<td>ellipse()</td>
<td>noFill()</td>
</tr>
<tr>
<td>; (semicolon)</td>
<td>line()</td>
<td>noStroke()</td>
</tr>
<tr>
<td>= (assign)</td>
<td>point()</td>
<td>stroke()</td>
</tr>
<tr>
<td>[] (array access)</td>
<td>quad()</td>
<td></td>
</tr>
<tr>
<td>{} (curly braces)</td>
<td>rect()</td>
<td></td>
</tr>
<tr>
<td>catch</td>
<td>triangle()</td>
<td></td>
</tr>
<tr>
<td>class</td>
<td>Curves</td>
<td></td>
</tr>
<tr>
<td>draw()</td>
<td>bezier()</td>
<td>Creating &amp; Reading</td>
</tr>
<tr>
<td>exit()</td>
<td>bezierDetail()</td>
<td>alpha()</td>
</tr>
<tr>
<td>extends</td>
<td>bezierPoint()</td>
<td>blue()</td>
</tr>
<tr>
<td>false</td>
<td>bezierTangent()</td>
<td>brightness()</td>
</tr>
<tr>
<td>final</td>
<td>curve()</td>
<td>color()</td>
</tr>
<tr>
<td>implements</td>
<td>curveDetail()</td>
<td>green()</td>
</tr>
<tr>
<td>import</td>
<td>curvePoint()</td>
<td>hue()</td>
</tr>
<tr>
<td>loop()</td>
<td>curveTangent()</td>
<td>lerpColor()</td>
</tr>
<tr>
<td>new</td>
<td>curveTightness()</td>
<td>red()</td>
</tr>
<tr>
<td>noLoop()</td>
<td>3D Primitives</td>
<td>saturation()</td>
</tr>
<tr>
<td>null</td>
<td>bowf()</td>
<td></td>
</tr>
<tr>
<td>popStyle()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>private</td>
<td></td>
<td></td>
</tr>
<tr>
<td>public</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

20
what is it?

- open-source, online community

- http://forum.processing.org/

- https://github.com/processing
why Processing?
why Processing?

- difficulty to sketch with other languages
- complicated setup
- not easy to learn
- repetitive code
why Processing?

- based on:
  - Logo
  - Design by Numbers
why Processing?

- program = sketch
why Processing?

- programming syntax

```java
void setup() {
  size(480, 120);
}

void draw() {
  if (mousePressed) {
    fill(0);
  } else {
    fill(255);
  }
  ellipse(mouseX, mouseY, 80, 80);
}
```
why Processing?

- Java-based
  - complexity
  + Big standard library
  + lots of user-contributed libraries

- similar syntax & portability

3 Billion Devices Run Java
Computers, Printers, Routers, Cell Phones, BlackBerry, Kindle, Parking Meters, Public Transportation Passes, ATMs, Credit Cards, Home Security Systems, Cable Boxes, TVs...
public class Hello
{
    public static void main(String args[])
    {
        System.out.println("Hello, world!");
    }
}

javac Hello.java
java Hello
why Processing?

- `println("Hello, World!");`
why Processing?

- active development
graphics
monitors

grid of pixels
shape

point(x, y);
shape

\texttt{line(x1, y1, x2, y2);}
shape

\texttt{rect}(x, y, width, height);
shape

ellipseMode(CENTER);

ellipse(x, y, width, height);

Example:
ellipseMode(CENTER);
ellipse(4,4,5,7);
shape

\texttt{triangle}(x_1, y_1, x_2, y_2, x_3, y_3);

\texttt{quad}(x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4);

\texttt{arc}(x, y, \text{width}, \text{height}, \text{start}, \text{stop});
color

-luminance

background(255);
color

- RGB (default)

```
  color c1 = color(r, g, b);

  color c2 = #RRGGBB;
```
color

- RGBA:
  - \( a = \text{alpha} / \text{transparency} / \text{opacity} \)
  - \( 0 = \text{transparent}; \ 255 = \text{opaque (solid)} \)

\[
\text{color } c1 = \text{color}(r, g, b, a);
\]
color

- color modes

- custom range

  colorMode(RGB, 100);

- HSB

  colorMode(HSB);
properties

noStroke();
fill(c1);
rect(...);
fill(c2);
stroke(c3);
ellipse(...);
properties

noFill();

noStroke();

elipse(...);

rect(...);

You get NOTHING!
order

- shapes are painted one at a time
- overlap can occur
- some shapes are not supported
animation

void setup() {
    ...
}

void draw() {
    ...
}
// in setup()
PFont myFont;
myFont = createFont("Georgia", 32);

// in draw()
textFont(myFont);
textAlign(CENTER, CENTER);
text("Hello, World!", width/2, height/2);
programming
interaction

-mouse

```cpp
void mouseClicked() {
    if (mouseButton == LEFT)
        fill(0);
    else if (mouseButton == RIGHT)
        fill(255);
    else
        fill(126);
}
```

void mousePressed()
void mouseReleased()
void mouseClicked()
void mouseDragged()
void mouseMoved()
void mouseWheel()

mouseX
mouseY
pmouseX
pmouseY
interaction

- keyboard

```cpp
void keyTyped()
{
    if (key == 'b')
        fill(0);
    else if (key == 'w')
        fill(255);
    else
        fill(126);
}
```

```
void keyPressed()

void keyReleased()

void keyTyped()

keyPressed
key
keyCode
```
structure

- comments, variables, arrays, loops

- **ArrayList** (also FloatList, IntList, StringList)

- **HashMap** (dict: also FloatDict, IntDict, StringDict)

- Table, XML, JSON
object-oriented

- with classes

```java
class oAnimal{
    boolean brain;
    int legs;

    oAnimal(){
        brain = true;
        legs = 0;
    }
}
```
folder structure

- folder [NAME] & [NAME].pde must match

- optional data folder (for images, input)
modes

- Java (default), JavaScript, Android, etc.
libraries
documentation

- available online
- also in the PDE

http://processing.org/reference/
exporting

creating applications is simple
examples

- variety of samples
instructions

- download Processing

- download time series project

- follow along in chapter
lab

- no lecture on Thurs.

- bring in laptops or email if you will be in a lab

- work on time series assignment; ask questions
L8. Tasks

REQUIRED READING
Chapter 3

Why: Task Abstraction

3.1 The Big Picture

Figure 3.1 breaks down into actions and targets the reasons why a vis tool is being used. The highest-level actions are to use vis to consume or produce information. The cases for consuming are to present, to discover, and to enjoy; discovery may involve generating or verifying a hypothesis. At the middle level, search can be classified according to whether the identity and location of targets are known or not: both are known with lookup, the target is known but its location is not for locate, the location is known but the target is not for browse, and neither the target nor the location are known for explore. At the low level, queries can have three scopes: identify one target, compare some targets, and summarize all targets. Targets for all kinds of data are finding trends and outliers. For one attribute, the target can be one value, the extremes of minimum and maximum values, or the distribution of all values across the entire attribute. For multiple attributes, the target can be dependencies, correlations, or similarities between them. The target with network data can be finding paths, and with spatial data the target can be understanding shape.

3.2 Why Analyze Tasks Abstractly?
The need for justifying 3D for abstract data is covered in Section 6.3.

6.7 Overview First, Zoom and Filter, Details on Demand

Ben Shneiderman’s influential mantra of **Overview First, Zoom and Filter, Details on Demand** [Shneiderman 96] is a heavily cited design guideline that emphasizes the interplay between the need for overview and the need to see details, and the role of data reduction in general and navigation in particular in supporting both.

A visual idiom that provides an **overview** is intended to give the user a broad awareness of the entire information space. Using the language of the what–why–how analysis framework, it’s an idiom with the goal of **summarize**. A common goal in overview design is to show all items in the dataset simultaneously, without any need for navigation to pan or scroll. Overviews help the user find regions where further investigation in more detail might be productive. Overviews are often shown at the beginning of the exploration process, to guide users in choosing where to drill down to inspect in more detail. However, overview usage is not limited to initial reconnaissance; it’s very common for users to interleave the use of overviews and detail views by switching back and forth between them many times.

When the dataset is sufficiently large, some form of **reduce action** must be used in order to show everything at once. Overview creation can be understood in terms of both filtering and aggregation. A simple way to create overviews is by zooming out geometrically, so that the entire dataset is visible within the frame. Each object is drawn smaller, with less room to show detail. In this sense, overviews are created by removing all filtering; an overview is created by changing from a zoomed-in view where some items are filtered out, to a zoomed-out view where all items are shown. When the number of items in a dataset is large enough, showing an overview of the entire dataset in a single screen using one mark per item is impossible, even if the mark size is decreased to a pixel.