Welcome to the Seminar on Visualization and Graphics with Processing

This is CS 50.

Part I: the building blocks

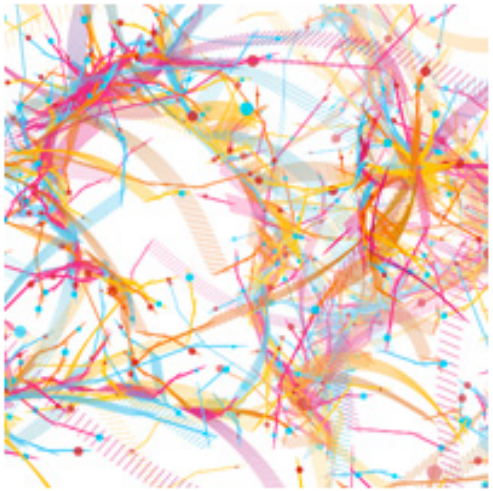
processing.org

Processing

Search

Cover \ Exhibition \ Reference \ Learning \ Download \ Shop \ About

»Feed »Forum »Wiki »Code



Announcing Processing.js 1.0!

Our sister project Processing.js has released its 1.0 version. As they explain, "Processing.js makes your data visualizations, digital art, interactive animations, educational graphs, video games, etc. work using web standards and without any plug-ins. You write code using the Processing language, include it in your web page, and Processing.js does the rest. It's not magic, but almost." They wrote a quick start guide for you.

» Download Processing
» Explore the Exhibition
» Play with Examples
» Browse Tutorials

Processing is an open source programming language and environment for people who want to create images, animations, and interactions. Initially developed to serve as a software sketchbook and to teach fundamentals of computer programming within a visual context, Processing also has evolved into a tool for generating finished professional work. Today, there are tens of thousands of students, artists, designers, researchers, and hobbyists who use Processing for learning, prototyping, and production.

» Free to download and open source

» Interactive programs using 2D, 3D or PDF output

» OpenGL integration for accelerated 3D

» For GNU/Linux, Mac OS X, and Windows

» Projects run online or as double-clickable applications

» Over 100 libraries extend the software into sound, video,

Genome Visualization Project

- From Harvard Pfister Lab http://gvi.seas.harvard.edu/sites/all/files/mizbee.pdf :

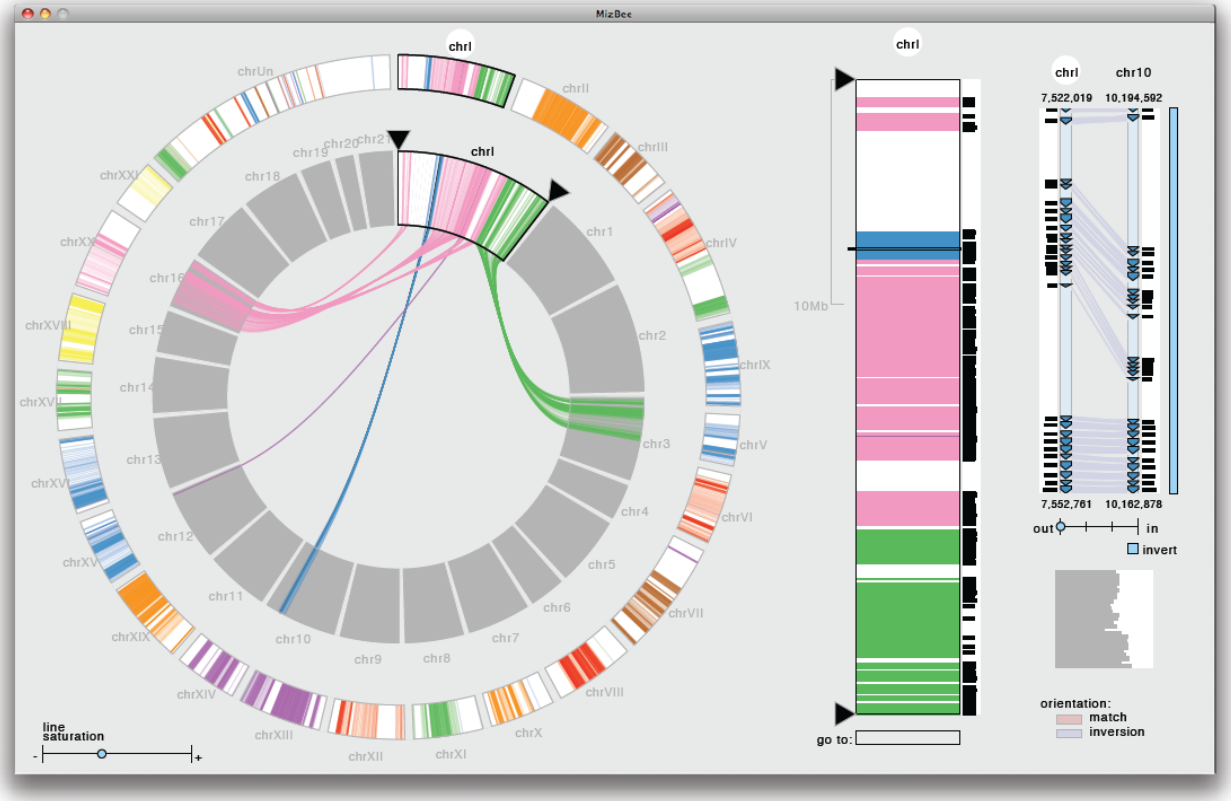
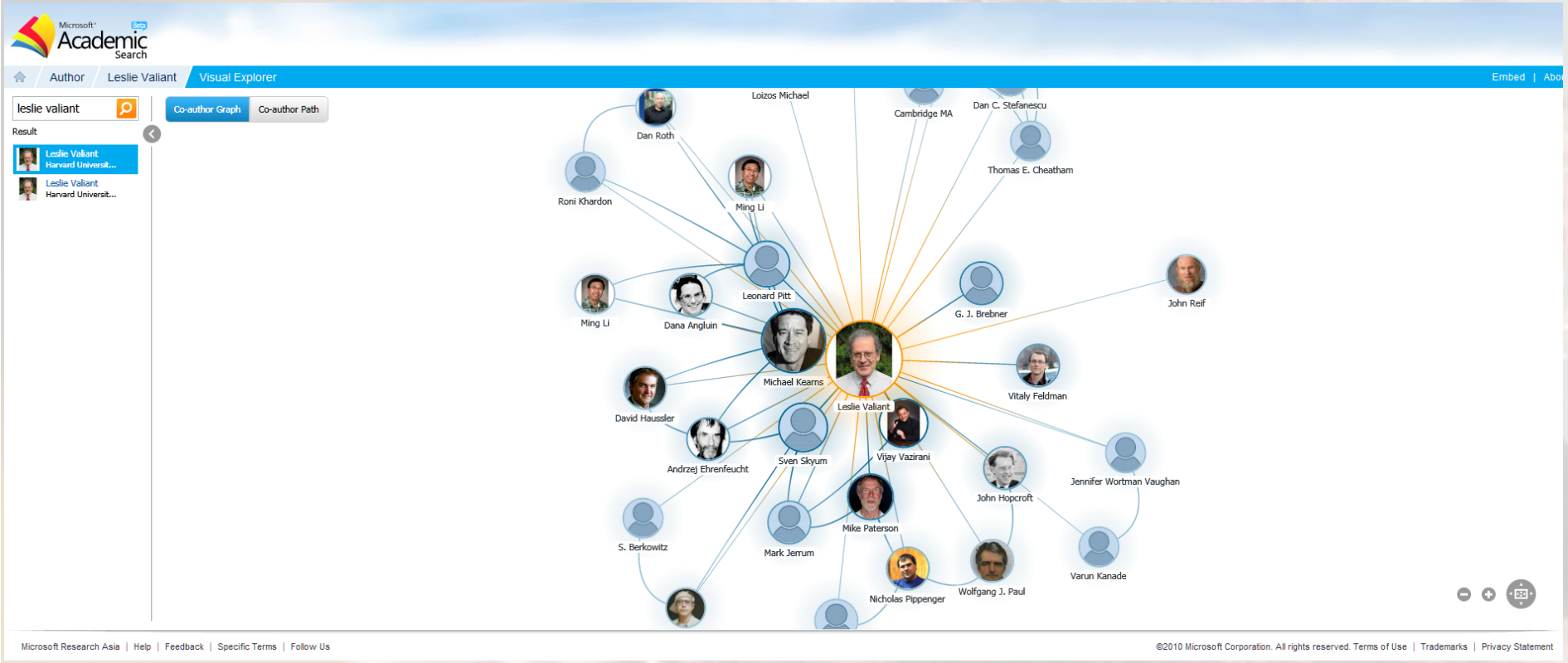


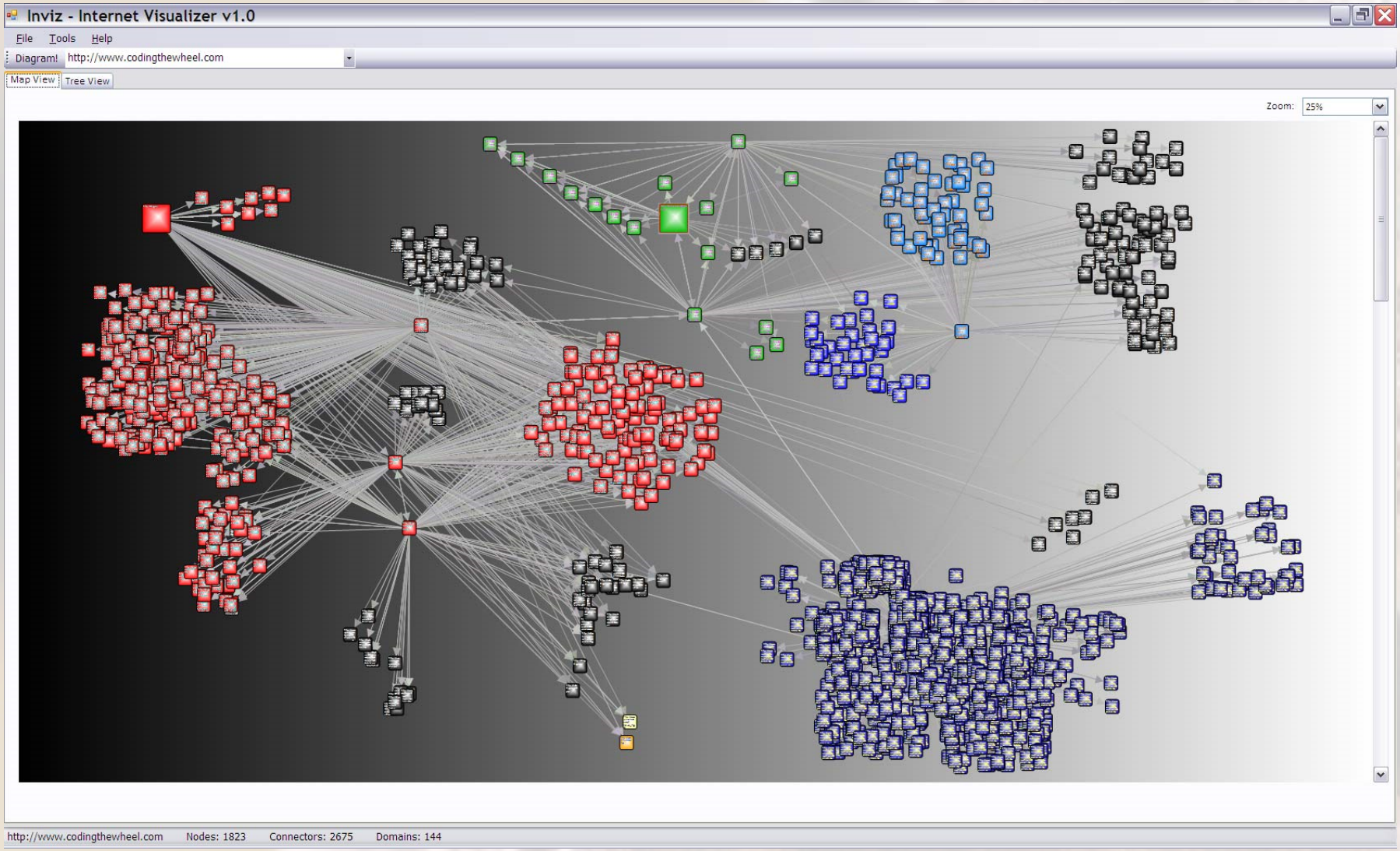
Fig. 1. The multiscale MizBee browser allows biologists to explore many kinds of conserved synteny relationships with linked views at the genome, chromosome, and block levels. Here we compare the genomes of two fish, the stickleback and the pufferfish.

MSR Academic Vis Project

- http://academic.research.microsoft.com/VisualExplorer.aspx#450662

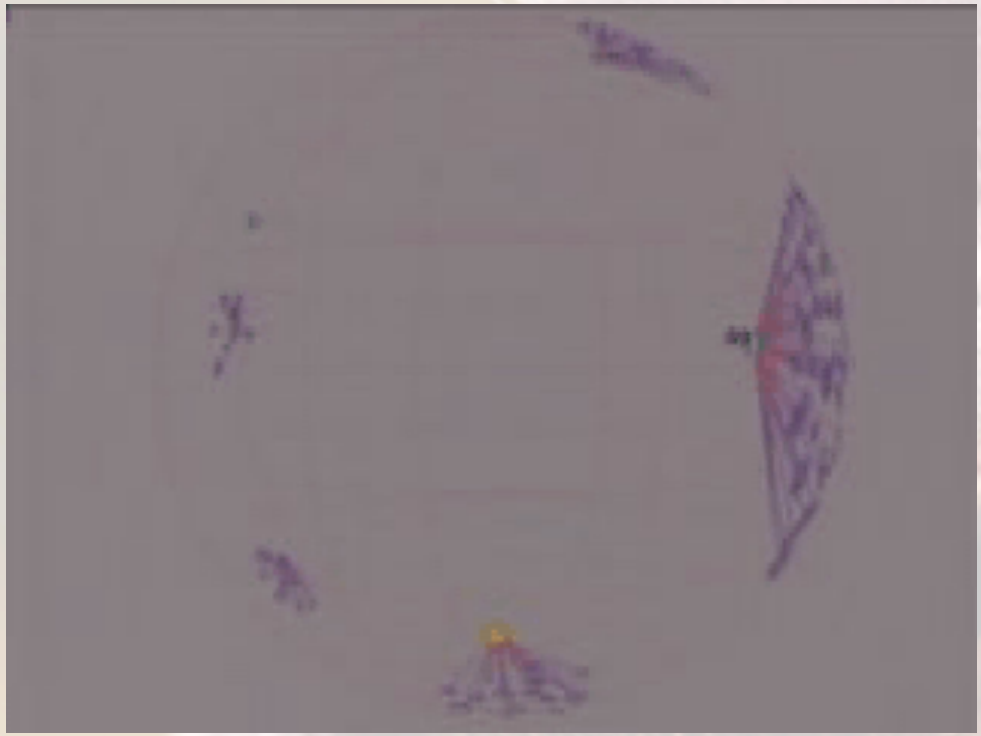


Visualizing the Internet



http://www.codingthewheel.com/archives/visualizing-the-internet

Stanford Munzer Graphics Lab



http://graphics.stanford.edu/videos/h3/

Getting Started

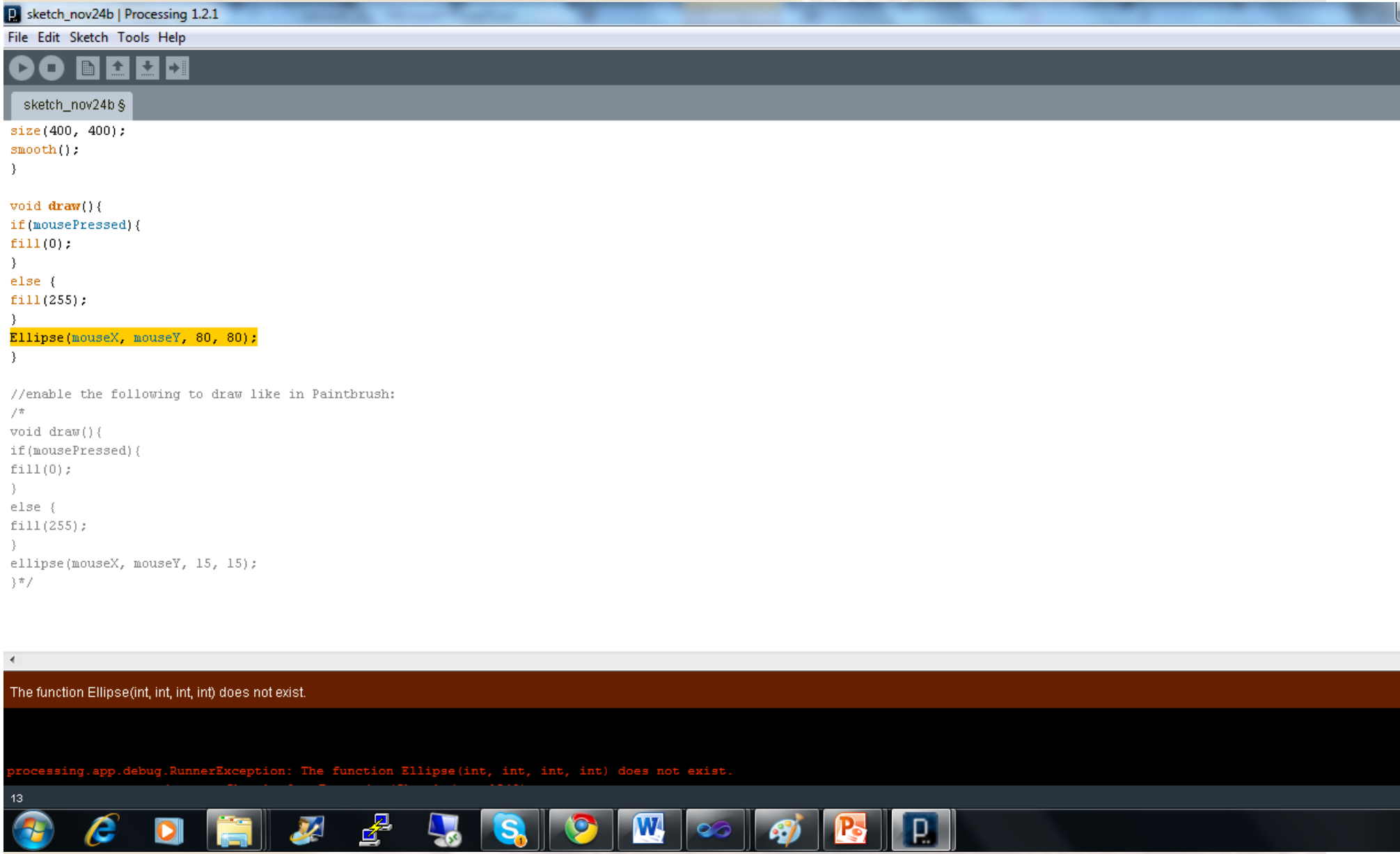
Every program has two functions:

- Setup – here you create a window for your draw, call functions to load your data sets
- Draw – pretty much everything related to what will be displayed inside the empty window you've created at setup

Hello, World! In … *Processing*

```
void setup(){
size(400, 400);
smooth();
}

void draw(){
if(mousePressed){
fill(0);
}
else {
fill(255);
}
Ellipse(mouseX, mouseY, 80, 80);
}
```


sketch_nov24b | Processing 1.2.1
File Edit Sketch Tools Help
sketch_nov24b §
size(400, 400);
smooth();
}
void draw(){
if(mousePressed){
fill(0);
}
else {
fill(255);
}
Ellipse(mouseX, mouseY, 80, 80);
}
//enable the following to draw like in Paintbrush:
/*
void draw(){
if(mousePressed){
fill(0);
}
else {
fill(255);
}
ellipse(mouseX, mouseY, 15, 15);
}*/
The function Ellipse(int, int, int, int) does not exist.
processing.app.debug.RunnerException: The function Ellipse(int, int, int, int) does not exist.
13

Draw - primitives

- Upper left corner of screen has coordinates (0,0), y axis orientation: down
- The following are functions for creating geometric primitives. Note that for “filling” with color (the fill function), the color selected remains the same for all future calls inside draw unless you call fill again to change the color. Fill takes either (r,g,b) in 0 to 255 range for each channel, or the color in hex format.

Basic Shapes

- point(200, 120);
- line(x1,y1,x2,y2);
- triangle(x1,y1,x2,y2,x3,y3);
- rect(x,y, width, height); //x,y top left corner
- ellipse(x,y, width, height);
- arc(x,y, width, height, start, stop); //start, stop in radians; arc can be used for pacman! ☺

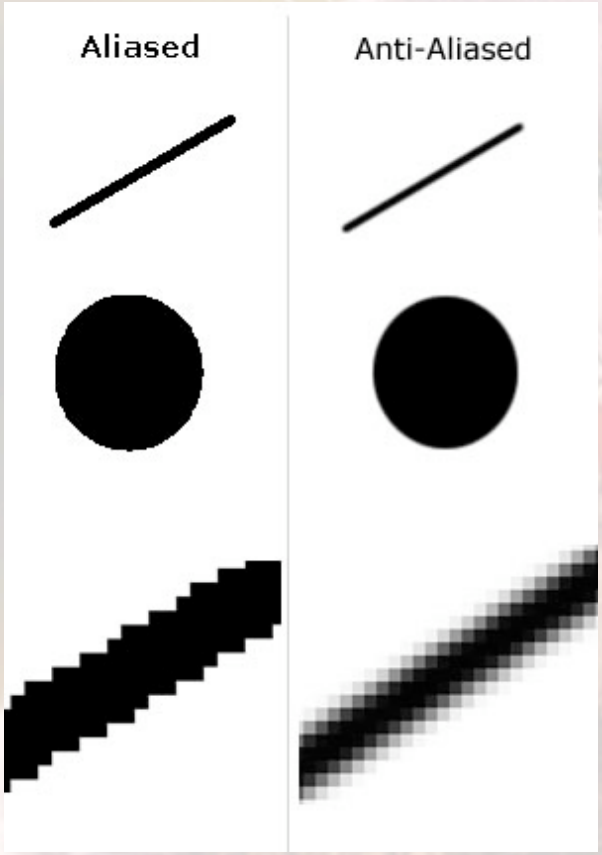
Drawing order

- The *last* line representing a function call to draw a geometric shape represents the shape drawn on top of all previous shapes.
- Simply: read your code from bottom to top – the last element is the one on top of the collage of geometric shapes you've drawn.

Radians() and Stroke()

- Useful function: call radians(x) and it will convert the value x in degrees to radians. Very useful in drawing arcs.
- Stroke is the outline of a shape. You can disable it with noStroke(). You can change the width of the contour line by using strokeWeight(x pt value);
- Smooth() is used to reduce aliasing. Can be disabled with noSmooth();

Aliasing and Anti-Aliasing (Smooth)



http://en.wikipedia.org/wiki/Aliasing

Fill, Stroke, Background

- If drawing in grayscale, these functions take only 1 parameter, 0 to 255 which represents degree of white (0 is black, 255 white).
- Fill can be disabled with noFill(). Then the color inside the drawn shape is the color of the background or of the shape in a more distal plane.
- Alpha value = transparency (0 as transparent, 255 as opaque). Optional fourth parameter to fill and stroke.

Custom Shapes

- Bound definition of a custom shape with

beginShape();

…

endShape();

Inside you can have as many vertices as you need:

vertex(x,y);

Variables, logic, loops, comments

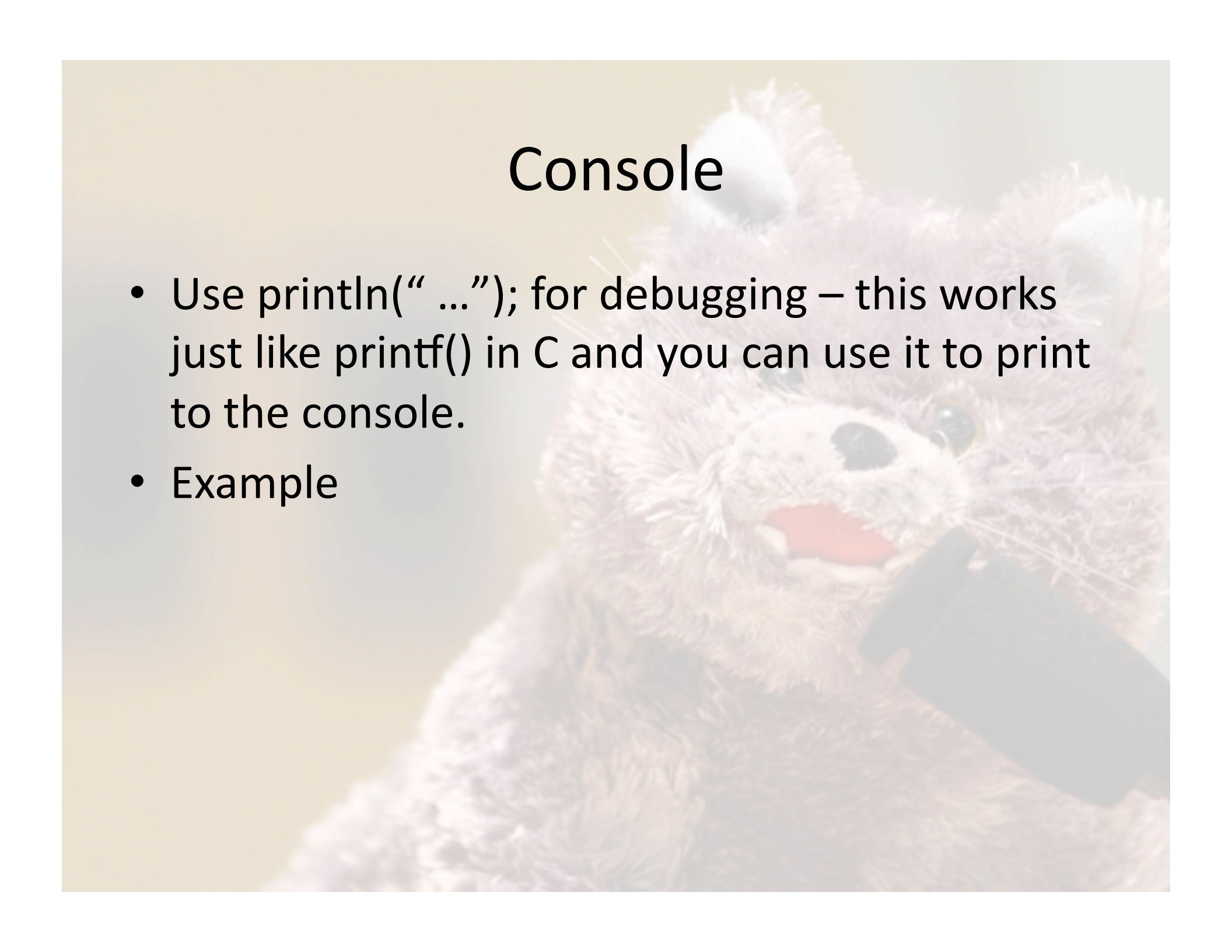
- Simple: variables are defined just as in C; same for loops, branching, and comments. The syntax is preserved.

- Also, arrays are declared as:

int**[]** a = **new** int[10];

Console

- Use println(“ …”); for debugging – this works just like printf() in C and you can use it to print to the console.
- Example



Setup()

- Setup only gets called once
- You can use it to load your datasets, or open textures, or simply set the fill color, window size, aliasing or anti-aliasing, etc. properties for the draw.

Mouse Variables

- Useful for drawing continuously: you have current mouseX, mouseY, but also the mouse positions at the previous frame, specifically pmouseX, pmouseY. These are directly accessible to you through Processing.

- Example

```
void draw(){
line(mouseX, mouseY, pmouseX, pmouseY);
}
```

Mouse Variables

- mousePressed is of type bool
- Returns true whenever a mousebutton is clicked

```
if(mousePressed)
        if(mouseButton == LEFT)
                …
```

Keyboard interaction

```
if(keyPressed)
        if((key == ‘a’) || (key == ‘z’))
                …//code for zoom for example
//for arrow keys
if(keyPressed && key == CODED)
        if(keyCode == LEFT)
                ….//leftarrow pressed
```

Media Export

- Can export a frame using:

saveFrame(“output.png”);

Can export to PDF:

//top of code:

import processing.pdf.*

In setup:

size(x, y, PDF, “output.PDF”);

In draw - at the end of draw, before}

exit();

Media Import

At top of code (declare global):

Pimage img;

- In setup:

img = loadImage(“foo.jpg”); //or png, gif

In draw:

image(img, 0,0);

Bounds of circles/rectangles

- Useful for creating buttons and for interaction with the visualization
- dist(x,y,x2,y2) represents the Euclidean distance between the two points
- Example

Text

```
//Global:
PFont font;
//In Setup:
font = loadFont(“foo.vlw”);
//In Draw():
textFont(font);
textAlign(CENTER);
textSize(14);
fill(0);
text(“LOLCAT”, x, y);
```

2D Animation

- You can check if a shape is off-screen by comparing its coordinates with the width and height you give to the window in setup() call to size(width, height). Thus you can implement “bouncing” off walls.
- Any animation implementation would be inside draw().
- Example

Random

- You can draw randomly, for instance in the pacman example we could randomly display an edible item:

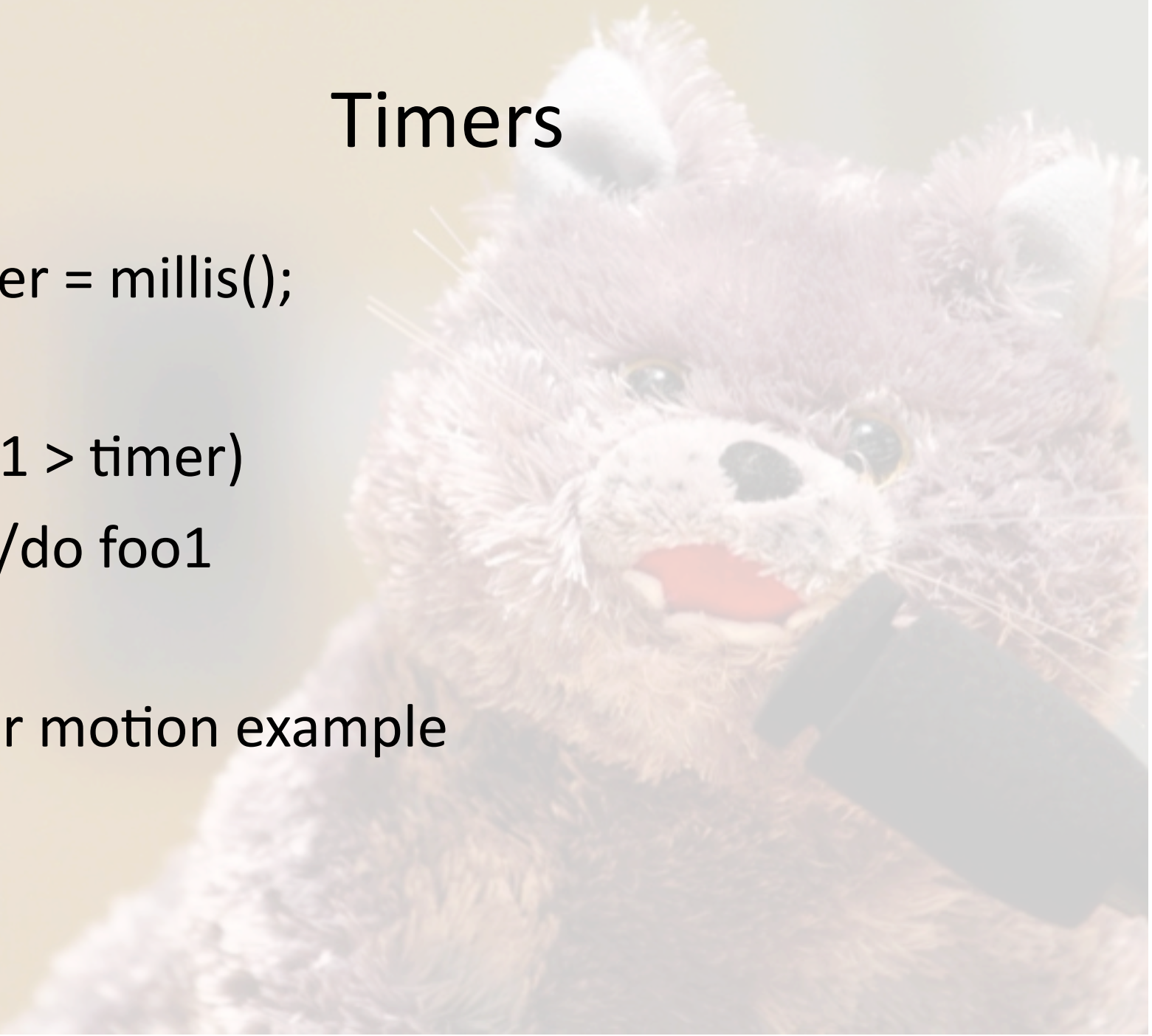
```
float randomX = random(0, mouseX);
float randomY = random(0, mouseY);
fill(100, 100, 100);
ellipse(randomX, randomY, 5, 5);
```

Timers

```
int timer = millis();

if(time1 > timer)
        //do foo1
```

Circular motion example

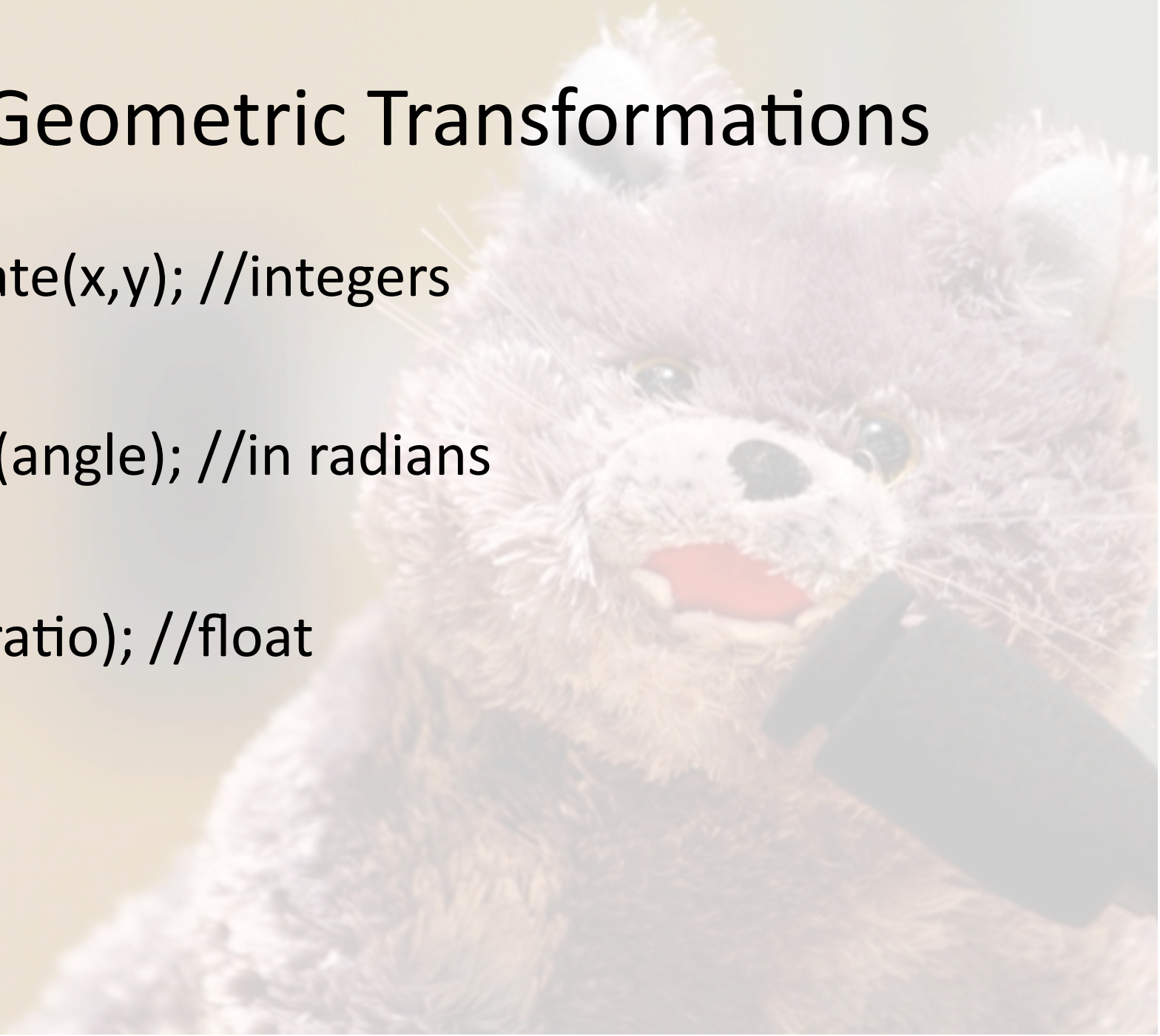


Geometric Transformations

translate(x,y); //integers

rotate(angle); //in radians

scale(ratio); //float



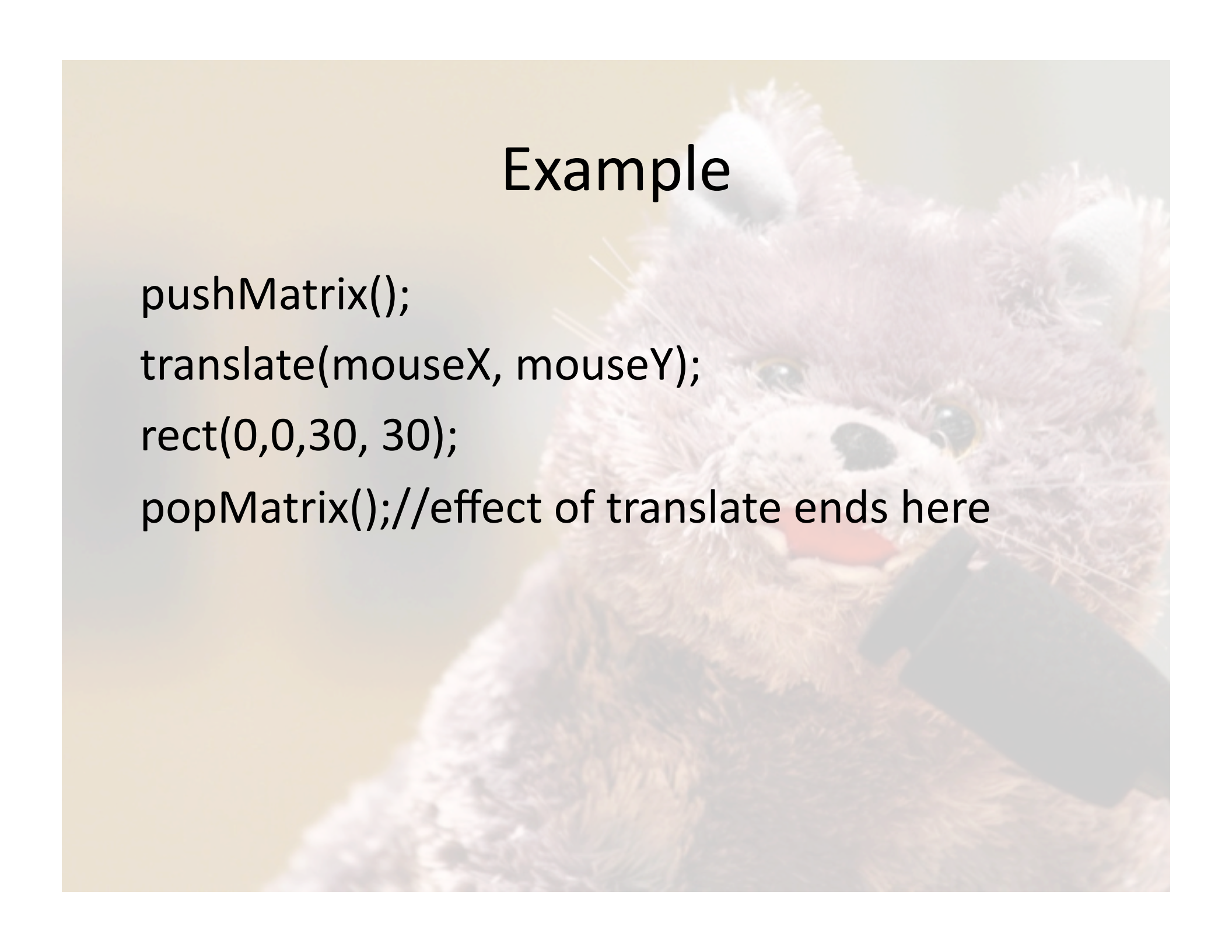
Geometric transformations

If you are using more than 1 geometric transform, you need to precede the transformations by pushMatrix() and end the stream of transformations with popMatrix()

These are always used in pairs and allow you to limit the effect of the transformations within the bounds of the push and pop (within those two you must also call the functions to draw the objects you want the transforms to apply to).

Example

```
pushMatrix();
translate(mouseX, mouseY);
rect(0,0,30, 30);
popMatrix();//effect of translate ends here
```



3D and OpenGL (intro)

```
import processing.opengl.*;

void setup()
{
 size(880, 440, OPENGL);
 noStroke();
 fill(255);
}
```

3D and OpenGL (intro)

Coordinate system: positive z axis points into the screen (away from you);

Lighting – ambient, directional, point (advanced)

Camera – points toward center of screen. The camera() function offers control over camera location, orientation (i,j,k), and camera target.

3D Intro

- Examples!

Part II: More advanced applications

- Example 1: getting data from a table

```
Table yourTable; //global
//in setup():
yourTable = new Table("C:\...");
int rowCount = yourTable.getRowCount();
//in draw:
for(int row = 0; i < rowCount; i++)
{
        float x=yourTable.getFloat(row, 1);
        float y = yourTable.getFloat(row,2);
}
```

Example 1: US Map

table.getFloat (row, column) works well for a TSV data input file (TSV = Tab Separated Value); this means every two data points on a row are separated by a tab.

You can replace the getFloat part with whatever data type you are using.

To add a file (ie, table.tsv) to your code folder, click on Sketch -> Add file

DEMO

Example 1

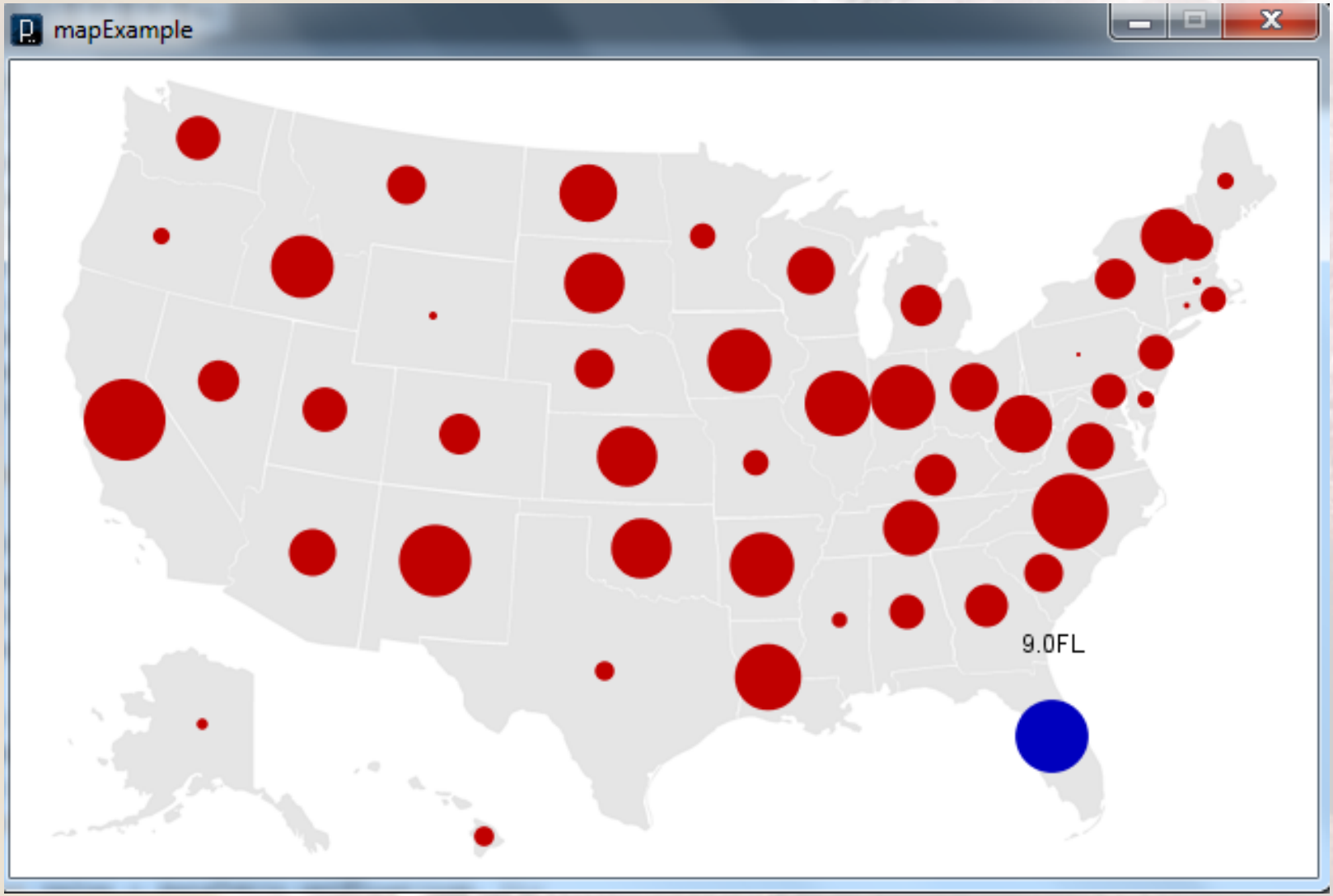
- You will need to download the library for dealing with Tables:

http://benfry.com/writing/map/Table.pde

Then Sketch->Add File->Table.pde

You will see the .pde (Processing source file) in a different tab

Outcome



Example 2: WordMap

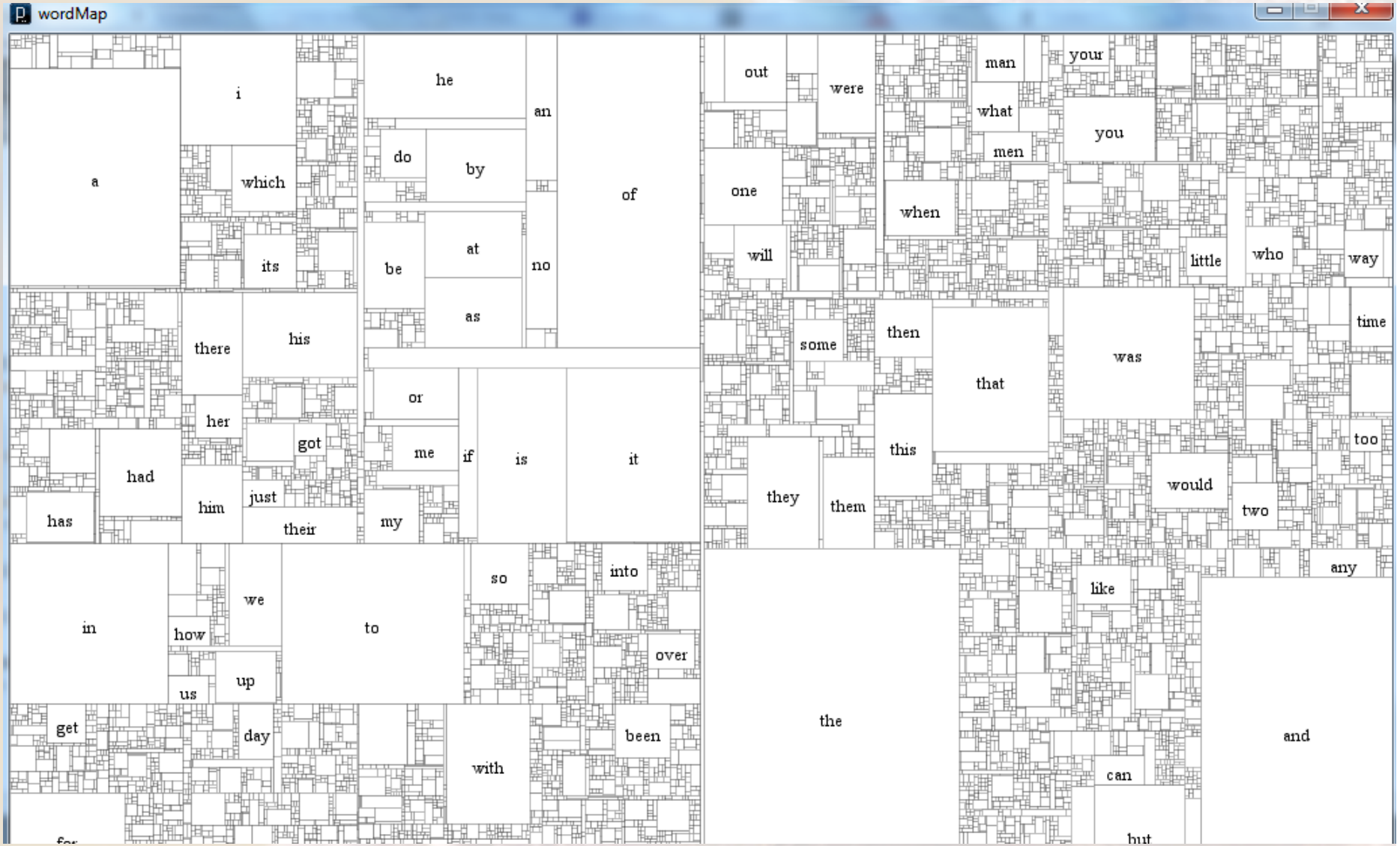
- The actual implementation of the wordmap is done in the *treemap* library:

http://benfry.com/writing/treemap/library.zip

Follow instructions in Chaper 7 of *Visualizing Data*, add user interaction using the building blocks taught in part 1.

For a test dataset, use Twain's *Following the Equator* from:
http://benfry.com/writing/treemap/equator.txt

Outcome


wordMap
a
i
which
its
he
do
by
an
of
be
at
as
no
there
his
her
got
had
him
just
has
their
or
me
my
if
is
it
out
were
man
your
what
you
men
one
when
will
little
who
way
time
some
then
that
was
too
this
would
they
them
two
any
like
in
how
we
to
so
into
over
us
up
the
and
get
day
been
with
can
but

DEMO

- See treemap.pde
- For you – enhance the visualization by adding user interaction – highlight of words, relationships between words and chapters, etc.

Getting your own dataset

- For final project, either create a dataset of your own, in TSV format (like a tech demo project)
- OR
- Use a scripting language such as Python combined with regular expressions
- For web scraping, use Python + library "Beautiful Soup" (allows HTML tag parsing)

Getting your own dataset

- For creating a dataset from a text file to create your TSV input file to Processing, please use:

http://docs.python.org/library/re.html

(regular expression library)

Recommended references

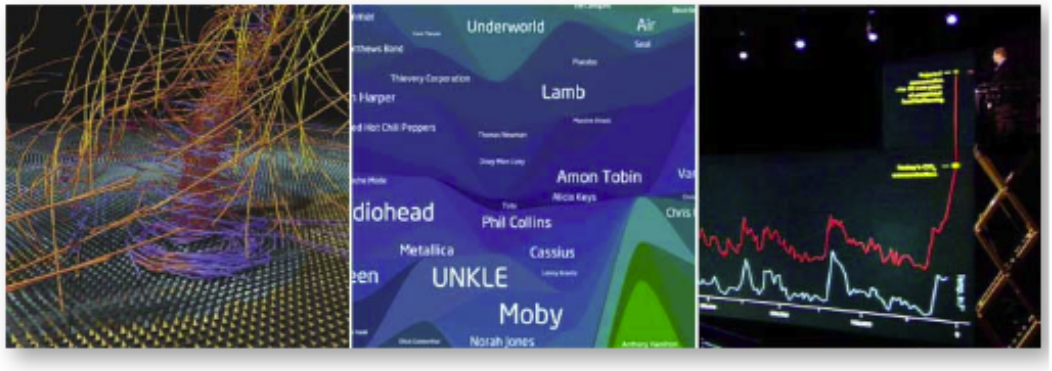
- Install processing (download) from processing.org. In the processing folder you will find the folder “examples”
- **Learning Processing**, Daniel Shiffman, Morgan Kaufman (2008)
- **Visualizing Data**, Ben Fry, O'Reilly (2007)
- The latter is highly recommended – walks you through full code of various large examples. *Learning Processing* is a good reference book.
- Processing.org/reference

After CS50…

CS 171 Visualization

Harvard
School of Engineering
and Applied Sciences

Home | News | Syllabus | Schedule | Homework | Projects | Forum | Resources



The amount and complexity of information produced in science, engineering, business, and everyday human activity is increasing at staggering rates. The goal of this course is to expose you to visual representation methods and techniques that increase the understanding of complex data. Good visualizations not only present a visual interpretation of data, but do so by improving comprehension, communication, and decision making.

Instructor: Hanspeter Pfister
Staff: Alberto Pepe (Head TF), Alex Chang, Kane Hsieh, Calvin McEachron, Lakshmi Parthasarathy, Mike Teodorescu

Lectures: M W 1-2:30 pm
Maxwell Dworkin G115

Sections: F 1-2:30 pm
Maxwell Dworkin G125

Adobe Connect Live Classroom
Adobe Connect Office Hours
Live Video (alternate)
Video Archive

Thank you!