introduction to processing
brushing & linking

Uta Hinrichs, CPSC 583, 2010
brushing & linking

- brushing
  - select a subset of the data to highlight or de-emphasize it
- linking
  - show connections of data in different visualizations
example: house data set

<table>
<thead>
<tr>
<th>factor</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>bungalow</td>
<td>2 storey</td>
<td>bungalow</td>
<td>2 storey</td>
<td>condomin</td>
<td>bungalow</td>
<td>2 storey</td>
<td>bungalow</td>
<td>2 storey</td>
<td>condomin</td>
<td>bungalow</td>
<td>2 storey</td>
</tr>
<tr>
<td>price</td>
<td>490000</td>
<td>550000</td>
<td>520000</td>
<td>1490000</td>
<td>830000</td>
<td>450000</td>
<td>630000</td>
<td>780000</td>
<td>890000</td>
<td>490000</td>
<td>450000</td>
<td>550000</td>
</tr>
<tr>
<td>sq footage</td>
<td>1400</td>
<td>1700</td>
<td>1200</td>
<td>2100</td>
<td>1900</td>
<td>950</td>
<td>990</td>
<td>1900</td>
<td>2500</td>
<td>1000</td>
<td>1100</td>
<td>1320</td>
</tr>
<tr>
<td>Distance parks</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>10</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Distance downtown</td>
<td>20</td>
<td>20</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>15</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>5</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>Distance LTR</td>
<td>5</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>5</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>Distance shops</td>
<td>5</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>5</td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>
housing data set: linking & brushing
housing data set: linking & brushing

- vis 1 (left): price, squ footage, year built, house type
  - x-axis: year built
  - y-axis: price
  - size: square footage
  - color: type of house
- vis 2 (right): distance vis (from left to right)
  - park
  - downtown
  - LRT
  - shops
housing data set: linking & brushing

- step 1: reading in the data
- step 2: create two “visualization windows”
- step 3: draw vis1; price, squ footage, year, house type
- step 4: draw vis2; distances
- step 5: make shapes interactively react to mouse hover
preparation

http://innovis.cpsc.ucalgary.ca/Courses/InfoVisTutorial2010

- download & unzip dataset.csv (houses.zip)
- download & unzip opencsv.jar (opencsv-1.7.zip)
- create a new java project in eclipse
- add setup() and draw() functions to your project
- add core.jar to “build path”
- add opencsv.jar to “build path”
- add houses.csv to “bin” folder of java project
- copy and paste “Houses” class into your java project
- copy and paste “Shape” class into your java project
housing data set: linking & brushing

- step 1: reading in the data
- step 2: create two “visualization windows”
- step 3: draw vis1; price, squ footage, year, house type
- step 4: draw vis2; distances
- step 5: make shapes interactively react to mouse hover
step 1: reading in the data

Vector houseList = new Vector<Houses>(); // holds all houses and their attributes

- copy and paste “readData” function into your main class
- call “readData” from setup() to read in houses.csv
- every house and attributes get stored in Houses object
- vector houseList contains all houses and their attributes
housing data set: linking & brushing

- step 1: reading in the data
- **step 2: create two “visualization windows”**
- step 3: draw vis1; price, sq footage, year, house type
- step 4: draw vis2; distances
- step 5: make shapes interactively react to mouse hover
step 2: create two visualization windows

- create function that draws two simple squares
  - each square will define the space for one of the visualizations
- make use of variables to define position & size of squares

```
EXAMPLE
int w1_xPos = 100;
int w1_yPos = 50;

int w2_xPos = 650;
int w2_yPos = 50;

int windowSize = 500;
```
step 2: create two visualization windows

- create function that creates two simple squares
- make use of variables to define position & size of squares

```java
int w1_xPos = 100;
int w1_yPos = 50;
int w2_xPos = 650;
int w2_yPos = 50;
int windowSize = 500;

public void drawWindows()
{
    this.rectMode(CORNER); // I want to draw the vis windows from the
    fill(255);
    stroke(0);
    strokeWeight(1);

    rect(w1_xPos, w1_yPos, windowSize, windowSize);
    rect(w2_xPos, w2_yPos, windowSize, windowSize);

    this.rectMode(CENTER); // I want to draw the data points from the center
}
housing data set: linking & brushing

- step 1: reading in the data
- step 2: create two “visualization windows”
- step 3: draw vis1; price, sq footage, year, house type
- step 4: draw vis2; distances
- step 5: make shapes interactively react to mouse hover
step 3: create visualization 1

- x-axis: year built
- y-axis: price
- size: square footage
- color: type of house
- important: create visuals based on Shape class

```
IN HOUSE CLASS
Vector shapeList = new Vector<Shape>();
public void initializeHouseShape(int windowPosX, int windowPosY, int windowSize)
{
    // create a Shape
    // define & define position, size, and color of this Shape (square) here
    // add Shape to shapeList
}

public void drawCostSizeYearVis()
{
    // have Shape drawn here -> shape.drawFunction...
}
```
excursion: mapping data to space

- normalize data to fit into our vis window
- define a maxValue
- define a minValue
- \((\text{value} - \text{minValue})/(\text{maxValue} - \text{minValue}) \times \text{windowSize}\)

between 0 and 1
excursion: mapping data to space

EXAMPLE: PRIZE & YEAR
float maxPrize = 1490000;
float minPrize = 450000;
float maxYear = 2007;
float minYear = 1911;

float xpos = windowPosX + ((year - minYear)/(maxYear - minYear) * windowSize);
float ypos = windowPosY + windowSize - ((price - minPrize)/(maxPrize - minPrice) * windowSize);
hints for visualization 1

- each Houses object should know about its Shapes
- each Shape knows what Houses object it belongs to
  - constructor of Shape class has been changed accordingly
- coloring
  - each Houses object should have a fixed color (based on its type)
  - each Houses object should have a current color that changes depending on mouse interaction
public void initializeHouseShape(int windowPosX, int windowPosY, int windowSize) {
    float maxPrice = 1490000;
    float minPrice = 450000;
    float maxYear = 2007;
    float minYear = 1911;
    float maxSize = 2100;
    float minSize = 950;
    float ypos = windowPosY + windowSize - ((price - minPrice)/(maxPrice - minPrice) * windowSize);
    float xpos = ((year - minYear)/(maxYear - minYear) * windowSize) + windowPosX;
    parent.strokeWeight(1);

    float s = (size - minSize)/(maxSize - minSize)* 30 + 10;  //sets the size (min size is 10, max size is 40)

    Shape houseShape = new Shape(parent, this, xpos, ypos, s, s);
    shapeList.add(houseShape);

    if(type.equals("bungalow"))
    {
        color = parent.color(255,0,0,alpha);
    }
    else if(type.equals("2 storey"))
    {
        color = parent.color(0,255,0,alpha);
    }
    else
    {
        color = parent.color(0,0,255,alpha);
    }
    currColor = color;
HOUSES CLASS

public void drawCostSizeYearVis() // gets called from the main class to draw shape
{
    parent.fill(currColor);
    parent.noStroke();
    ((Shape)(shapeList.elementAt(0))).drawShapeSizeYearVis(); // first element in shapeList is the
    // square for vis 1
}

SHAPE CLASS

void drawShapeSizeYearVis()
{
    parent.rect(xPos, yPos, sWidth, sHeight);
}
visualization 1 – call from main class

MAIN CLASS
public void draw() {
    background(230);
    drawWindows();
    for(int i = 0; i<houseList.size(); i++) {
        Houses h = ((Houses)houseList.elementAt(i));
        h.currColor = h.color;
        h.currStrokeWeight = 1;

        ((Houses)houseList.elementAt(i)).drawCostSizeYearVis();
        ((Houses)houseList.elementAt(i)).drawDistanceVis();
    }
}
housing data set: linking & brushing

- step 1: reading in the data
- step 2: create two “visualization windows”
- step 3: draw vis1; price, squ footage, year, house type
- step 4: draw vis2; distances
- step 5: make shapes interactively react to mouse hover
step 4: draw vis2; distances

- draw four vertical axis (one for each distance)
- for each house mark the different distances as a circle
- add each circle to the shapeList of each house
- for each house connect the according circles through lines
- color circles and lines according to the house type

Hints:
- distribute data evenly on axis
  → maxDistance, minDistance
public void initializeDistShapes(int windowPosX, int windowPosY, int windowSize, float l2_xPos, float l3_xPos) {
    float minDist = 0;
    float maxDist = 20;

    Shape s1 = new Shape(parent, this, (float)windowPosX, (parkDist - minDist)/(maxDist - minDist)*windowSize + windowPosY, 5, 5);
    Shape s2 = new Shape(parent, this, l2_xPos, (DTDist - minDist)/(maxDist - minDist)*windowSize + windowPosY, 5, 5);
    Shape s3 = new Shape(parent, this, l3_xPos, (LRTDist - minDist)/(maxDist - minDist)*windowSize + windowPosY, 5, 5);
    Shape s4 = new Shape(parent, this, windowPosX+ windowSize,(shopDist - minDist)/(maxDist - minDist)*windowSize + windowPosY, 5, 5);

    shapeList.add(s1);
    shapeList.add(s2);
    shapeList.add(s3);
    shapeList.add(s4);
}
public void drawDistanceVis()
{
    parent.fill(currColor);
    parent.stroke(currColor);
    parent.strokeWeight(currStrokeWeight);
    for(int i = 1; i<shapeList.size(); i++)
    {
        if(i<shapeList.size()-1)
        {
            ((Shape)(shapeList.elementAt(i))).drawDistances((Shape)(shapeList.elementAt(i+1)));
        }
        else
        {
            ((Shape)(shapeList.elementAt(i))).drawDistances(null);
        }
    }
}
SHAPE CLASS

void drawDistances(Shape s)
{
    parent.ellipse(xPos, yPos, sWidth, sHeight);
    if(s != null)
    {
        parent.line(xPos, yPos, s.xPos, s.yPos);
    }
}
public void setup()
{
    size (1200, 600);
    drawWindows();
    readData(dataPath);
    for(int i = 0; i<houseList.size(); i++)
    {
        ((Houses)houseList.elementAt(i)).initializeHouseShape(w1_xPos, w1_yPos, windowSize);
        ((Houses)houseList.elementAt(i)).initializeDistShapes(w2_xPos, w2_yPos, windowSize, l1_xPos, l2_xPos);
    }
}

public void draw()
{
    background(230);
    drawWindows();
    for(int i = 0; i<houseList.size(); i++)
    {
        Houses h = ((Houses)houseList.elementAt(i));
        h.currColor = h.color;
        h.currStrokeWeight = 1;
        ((Houses)houseList.elementAt(i)).drawCostSizeYearVis();
        ((Houses)houseList.elementAt(i)).drawDistanceVis(); //draw distance vis
    }
}
housing data set: linking & brushing

- step 1: reading in the data
- step 2: create two “visualization windows”
- step 3: draw vis1; price, sq footage, year, house type
- step 4: draw vis2; distances
- step 5: make shapes interactively react to mouse hover
step 5: make shapes interactively react to mouse hover

- go through all shapes and check if the mouse pointer “touches” them
  - Shape.contains(x,y);
- if mouse pointer touches:
  - highlight the shape in a special color
  - also highlight the according data in the other visualization
  - draw strokes in highlighting color
  - change thickness of strokes
public void draw()
{
    background(230);
    drawWindows();
    for(int i = 0; i<houseList.size(); i++)
    {
        Houses h = ((Houses)houseList.elementAt(i));
        h.currColor = h.color;
        h.currStrokeWeight = 1;
        for(int j = 0; j<h.shapeList.size(); j++)
        {
            Shape s = ((Shape)(h.shapeList.elementAt(j)));
            if(s.contains(mouseX, mouseY))
            {
                h.currColor = color(255,255,0,255);
                h.currStrokeWeight = 3;
            }
        }
        ((Houses)houseList.elementAt(i)).drawCostSizeYearVis();
        ((Houses)houseList.elementAt(i)).drawDistanceVis();
    }
}