4.1 Interaction Mechanisms
Visual information seeking

- Mantra for visual information seeking: „Overview first, zoom and filter, details on demand!“ (Ben Shneiderman)
Interactions

- Still images only allow for displaying some information.
- An analysis process requires interactions.
- Which interaction mechanisms exist?
1. Select

- Mark something as interesting
2. Explore

• Show me something else
3. Reconfigure

• Show me a different arrangement
4. Encode

• Show me a different representation
5. Abstract / Elaborate

- Show me more or less detail
6. Filter

• Show me something conditionally
7. Connect

- Show me related items
Interaction mechanisms

1. Select
2. Explore
3. Reconfigure
4. Encode
5. Abstract/Elaborate
6. Filter
7. Connect
4.2 Examples

Generated with Tableau software:
http://www.tableausoftware.com/learn/gallery
Visualization examples

The 2005 Hurricane Season

Select Basin
- Atlantic
- Eastern Pac.
- Western Pac.

Click to highlight:
- ADELENE
- BRET
- CINDY
- DENNIS
- EMILY
- FRANKLIN
- GERT
- HARVEY
- IRENE
- JOSE
- KATRINA
- LEE
- MARIA
- NATE
- OLIVIA
- PHILIPPE
- RITA
- STAN
- TAMMY
- VINCE
- WILMA
- ALPHA
- BETA
- DELTA
- EPSILON
- GAMMA
- ZETA

Energy and Wind Speed by Time

Jacobs University
Visualization and Computer Graphics Lab
Data Analytics
Visualization examples

Where is Oil Produced?

Select Year: 1998

- U.S.
- Saudi Arabia
- Russian Federation
- Iraq
- Venezuela
- China
- Mexico
- Canada
- Kuwait
- United Arab Emirates
- Iran
- Nigeria
- Libya
- United Kingdom
- United Arab Emirates

**1998**

Visualization examples

Seattle Real Estate: Overview

Select Dates:
May, 2009  January, 2009

Number of Home Sales
- King
- Pierce
- Snohomish
- Skagit
- Thurston
- Island
- Whatcom

Months of Supply

Change in Median Price

% Change in Price from First Period
Visualization examples
Visualization examples

California Revenue Sources

Select Date: 1999

Revenue Over Time

% Difference From Previous Year

Data Analytics

Jacobs University
Visualization and Computer Graphics Lab
Visualization examples

District of Columbia Crimespotting
Month of December 2009

Filters
Click and drag to limit the crimes shown

Crime Type
[Select]

District
[Select]

Crime Type
- HOMICIDE
- ASSAULT
- ROBBERY
- SEX OFFENSE
- ARSON
- BURGLARY
- THEFT
- THEFT AUTO

Monthly Overview by Day of the Week

Crime Frequency (click to select)
- THEFT 20%
- ROBBERY 18%
- THEFT AUTO 14%
- BURGLARY 13%
- STOLEN AUTO 12%
- ASSAULT 5%
Visualization examples
Visualization examples

Venture Financing

Although software funding has dramatically declined after the dot-com period, it still receives more funding than its competing sections.

Select Date Range:
- 1995 - 2010

Select Quarter:
- Q1
- Q2
- Q3
- Q4

Select Sector:
- (All)
- (Software)
- (Hardware)
- (Industrial)
- (Biotech)

Select Funding Type:
- (All)
- First round
- Total

Data Analytics
4.3 Tools & programming environments
Tableau

http://www.tableausoftware.com

Seattle Real Estate: Overview

Number of Home Sales

Months of Supply

Select Date:
May, 2000
January, 2009

Change in Median Price

Select Date:
May, 2000
January, 2009
Many Eyes

- Free online visualization tool by IBM
- Discontinued

Bubble chart
A bubble chart visualization shows popular digital camera models based on several hundred million Flicker images. View this chart on Many Eyes.
Python & Pygal

• Pygal: Dynamic charting library for Python
• [http://pygal.org](http://pygal.org)
• Others exist: vispy, bokeh, seaborn, folium, networkx,...

```python
data = [('Firefox', [None, None, 0, 16.6, 25, 31, 36.4, 45.5, 46.3, 42.8, 37.1]),
        ('Chrome', [None, None, None, None, None, None, 0, 3.9, 10.8, 23.8, 35.3]),
        ('IE', [85.8, 84.6, 84.7, 74.5, 66, 58.6, 54.7, 44.8, 36.2, 26.6, 20.1]),
        ('Others', [14.2, 15.4, 15.3, 8.9, 9, 10.4, 8.9, 5.8, 6.7, 6.8, 7.5])]

line_chart = pygal.Line()
line_chart.title = 'Browser usage evolution (in %)'
line_chart.x_labels = map(str, range(2002, 2013))
for label, values in data:
    line_chart.add(label, values)
line_chart.render()
```
Pygal - Line Charts

Browser usage evolution (in %)

- Firefox
- Chrome
- IE
- Others

Pygal - Stacked Line Chart

```python
stackedline_chart = pygal.StackedLine(fill=True)
stackedline_chart.title = 'Browser usage evolution (in %)'
stackedline_chart.x_labels = map(str, range(2002, 2013))
stackedline_chart.add('Firefox', [None, None, 0, 16.6, 25, 31, 36.4, 45.5, 46.3, 42.8, 37.1])
stackedline_chart.add('Chrome', [None, None, None, None, None, None, 0, 3.9, 10.8, 23.8, 35.3])
stackedline_chart.add('IE', [85.8, 84.6, 84.7, 74.5, 66, 58.6, 54.7, 44.8, 36.2, 26.6, 20.1])
stackedline_chart.add('Others', [14.2, 15.4, 15.3, 8.9, 9, 10.4, 8.9, 5.8, 6.7, 6.8, 7.5])
stackedline_chart.render()
```
Pygal - Bar Charts

```python
bar_chart = pygal.Bar()
bar_chart.title = 'Browser usage evolution (in %)'
bar_chart.x_labels = map(str, range(2002, 2013))
bar_chart.add('Firefox', [None, None, 0, 16.6, 25, 31, 36.4, 45.5, 46.3, 42.8, 37.1])
bar_chart.add('Chrome', [None, None, None, None, None, None, None, 0, 3.9, 10.8, 23.8, 35.3])
bar_chart.add('IE', [85.8, 84.6, 84.7, 74.5, 66, 58.6, 54.7, 44.8, 36.2, 26.6, 20.1])
bar_chart.add('Others', [14.2, 15.4, 15.3, 8.9, 9, 10.4, 8.9, 5.8, 6.7, 6.8, 7.5])
bar_chart.render()
```
Pygal - Pyramid Charts

England population by age in 2010 (source: ons.gov.uk)
Pygal - Pyramid Charts

ages = [(364381, 358443, 360172, 345848, 334895, 326914, 323053, 312576, 302015, 301277, 309874, 318295, 32
(346205, 340570, 342668, 328475, 319010, 312898, 308153, 296752, 289639, 290466, 296190, 303871, 309886,
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 23, 91, 412, 1319, 2984, 5816, 10053, 16045, 24240, 350
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 74, 392, 1351, 3906, 7847, 12857, 19913, 29108, 42475,
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 5, 17, 15, 31, 34, 38, 35, 45, 299, 295, 218, 247, 2
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 8, 0, 8, 21, 34, 49, 84, 97, 368, 401, 414, 557, 654
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 5, 0, 11, 35, 137, 331, 803, 1580, 2361, 3632, 48
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 6, 11, 20, 68, 179, 480, 1077, 2094, 3581, 5151, 704

types = ['Males single', 'Females single',
'Males married', 'Females married',
'Males widowed', 'Females widowed',
'Males divorced', 'Females divorced']

pyramid_chart = pygal.Pyramid(human_readable=True, legend_at_bottom=True)
pyramid_chart.title = 'England population by age in 2010 (source: ons.gov.uk)'
pyramid_chart.x_labels = map(lambda x: str(x) if not x % 5 else '', range(90))
for type, age in zip(types, ages):
    pyramid_chart.add(type, age)
pyramid_chart.render()
Pygal - Dot Charts

```python
dot_chart = pygal.Dot(x_label_rotation=30)
dot_chart.title = 'V8 benchmark results'
dot_chart.x_labels = ['Richards', 'DeltaBlue', 'Crypto', 'RayTrace', 'EarleyBoyer',
'Chrome', [6395, 8212, 7520, 7218, 12464, 1660, 2133, 8607]],
dot_chart.add('Chrome', [6395, 8212, 7520, 7218, 12464, 1660, 2133, 8607])
dot_chart.add('Firefox', [7473, 8099, 11700, 2651, 6361, 1044, 3797, 9450])
dot_chart.add('Opera', [3472, 2933, 4203, 5229, 5810, 1828, 9013, 4669])
dot_chart.add('IE', [43, 41, 59, 79, 144, 136, 34, 102])
dot_chart.render()
```
Pygal - Boxplots

```python
box_plot = pygal.Box()
box_plot.title = 'V8 benchmark results'
box_plot.add('Chrome', [6395, 8212, 7520, 7218, 12464, 1660, 2123, 8607])
box_plot.add('Firefox', [7473, 8099, 11700, 2651, 6361, 1044, 3797, 9450])
box_plot.add('Opera', [3472, 2933, 4203, 5229, 5810, 1828, 9013, 4669])
box_plot.add('IE', [43, 41, 59, 79, 144, 136, 34, 102])
box_plot.render()
```
Pygal - Pie Charts

```python
pie_chart = pygal.Pie()
pie_chart.title = 'Browser usage in February 2012 (in %)'
pie_chart.add('IE', 19.5)
pie_chart.add('Firefox', 36.6)
pie_chart.add('Chrome', 36.3)
pie_chart.add('Safari', 4.5)
pie_chart.add('Opera', 2.3)
pie_chart.render()
```
```python
xy_chart = pygal.XY(stroke=False)
xy_chart.title = 'Correlation'
xy_chart.add('A', [[0, 0], [0.1, 0.2], [0.3, 0.1], [0.5, 1], [0.8, 0.6], [1, 1.08], [1.3, 1.1], [2, 3.23], [2.43, 2]])
xy_chart.add('B', [[0.1, 0.15], [0.12, 0.23], [0.4, 0.3], [0.6, 0.4], [0.41, 0.21], [0.5, 0.3], [0.6, 0.8], [0.7, 0.8]])
xy_chart.add('C', [[0.05, 0.01], [0.13, 0.02], [1.5, 1.7], [1.52, 1.6], [1.8, 1.63], [1.5, 1.82], [1.7, 1.23], [2.1, 0]])
xy_chart.render()
```
Pygal - Radar Charts

```python
radar_chart = pygal.Radar()
radar_chart.title = 'V8 benchmark results'
radar_chart.x_labels = ['Richards', 'DeltaBlue', 'Crypto', 'RayTrace', 'EarleyBoyer',
                        'Chrome', [6395, 8212, 7520, 7218, 12464, 1660, 2123, 8607]],
radar_chart.add('Firefox', [7473, 8099, 11700, 2651, 6361, 1044, 3797, 9450]),
radar_chart.add('Opera', [3472, 2933, 4203, 5229, 5810, 1828, 9013, 4669]),
radar_chart.add('IE', [43, 41, 59, 79, 144, 136, 34, 102])
radar_chart.render()
```
Pygal - World Maps

```python
worldmap_chart = pygal.Worldmap()
worldmap_chart.title = 'Some countries'
worldmap_chart.add('F countries', ['fr', 'fi'])
worldmap_chart.add('M countries', ['ma', 'mc', 'md', 'me', 'mg', 'mk', 'ml', 'mn', 'mo', 'mr', 'mt', 'mu', 'mv', 'mw', 'mx', 'my', 'mz'])
worldmap_chart.add('U countries', ['ua', 'ug', 'us', 'uy', 'uz'])
worldmap_chart.render()
```
R & R Graphics

- R: statistical software
- R Graphics: statistical graphics

```r
x1 <- rnorm(50)
png("plottype.png")
par(mfrow = c(2,2))
plot(x1, type = "p", main = "points", ylab = "", xlab = "")
plot(x1, type = "l", main = "lines", ylab = "", xlab = "")
plot(x1, type = "b", main = "both", ylab = "", xlab = "")
plot(x1, type = "o", main = "both overplot", ylab = "", xlab = ")
dev.off()
```
Javascript & D3

- Javascript: dynamic operations in web browser
- D3 – Javascript library
- http://d3js.org
Graphics Programming

- Graphical user interface (GUI)
- Windowing system
  - X
  - Integrated in OS: Microsoft Windows, Mac OS
- Microsoft Direct3D
- OpenGL (Open Graphics Library)
- OpenGL 2.0 (and higher)
  - GLSL (GL Shading Language)
- Cg (nVidia)
- HLSL (high level shader language, DirectX)
- GPGPU programming: CUDA (nVidia), CTM (ATI), OpenCL
C & OpenGL

- OpenGL is a software interface for 3D computer graphics

- Originally developed by SGI (IRIS GL) since 1992 under control of ARB (Architecture Review Board)

- OpenGL specification is
  - hardware-independent,
  - window system independent and
  - operating system independent.

- Different implementations (as hardware and/or software)

- C library
OpenGL architecture

- CPU
  - Display List
  - Polynomial Evaluator
  - Per Vertex Operations & Primitive Assembly
  - Rasterization
  - Texture Memory
  - Pixel Operations
  - Per Fragment Operations
  - Frame Buffer
OpenGL examples
Summary

- Excel etc.: No interaction.
- Tableau, ManyEyes: Powerful, but not open source.
- Python: Flexible, but only some interactions possible.
- R: Easy to create plots, but interaction limited.
- D3: Powerful and flexible, Javascript programming.
- OpenGL: Most flexible, basic graphics library, C programming.
4.4 Assignment
Assignment 3

Consider, again, the Automobile Data Set as in Assignment 2.

1. Analysis Task 1 is to find the relation between the six numerical attributes curb-weight, engine-size, horsepower, city-mpg, highway-mpg, and price. Come up with a solution that uses the interaction mechanisms “Select” and “Reconfigure” on suitable visual encodings to find the relation between more than two of the attributes. Implement the interactive mechanisms (and visual encodings). What conclusions can be drawn?

2. Enhance the Analysis Task 1 by allowing for the interaction mechanism “Filter” with respect to the other attributes, i.e., make, fuel-type, body-style, and num-of-cylinders. What additional insight can be obtained?

3. Develop an own Analysis Task 2, where you are required to implement any of the remaining four interaction mechanisms to do the analysis best. Discuss the findings.