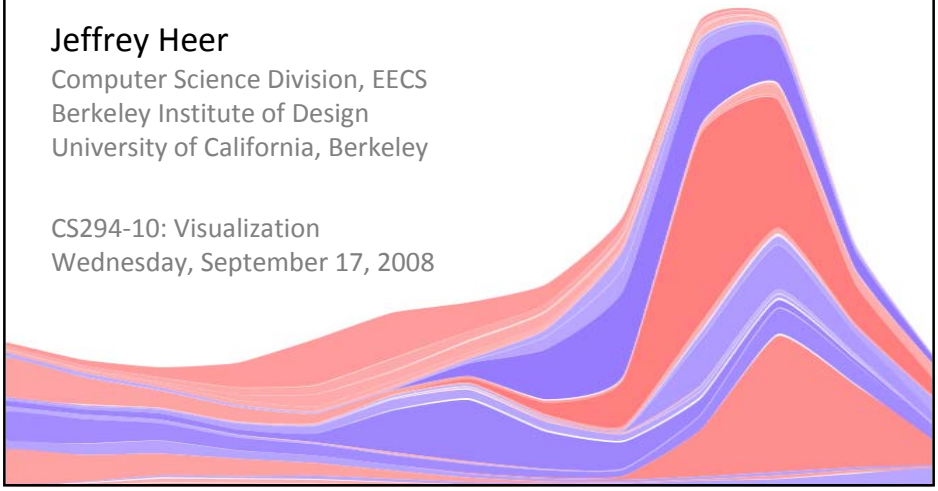
Visualization Programming with Flare

Jeffrey Heer
Computer Science Division, EECS
Berkeley Institute of Design
University of California, Berkeley

CS294-10: Visualization
Wednesday, September 17, 2008

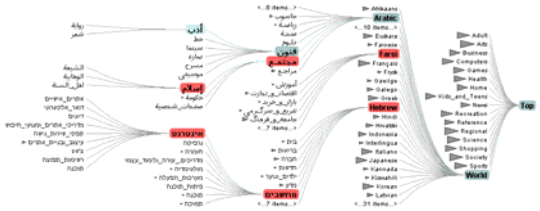


Visualization Programming

Setbacks to Visualization Adoption

Visualizations are hard to create
Layout algorithms, dynamic graphics

Pre-built "widgets" are not enough
Good designs tailored to an application domain

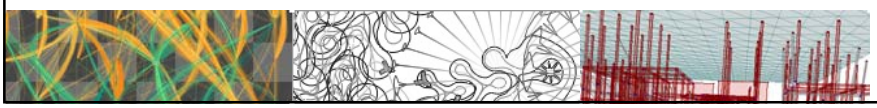


Data Management
Layout & Encoding
Rendering
Animation
Interaction

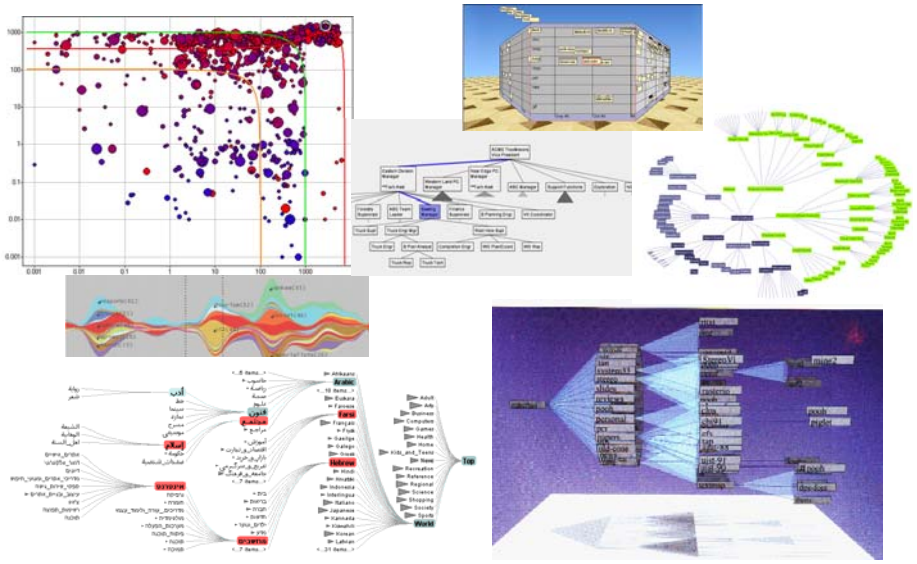
Graphics / Visualization Tools

OpenGL: Industry standard 2D/3D graphics API
Java2D, GDI+ (Win), Quartz (Mac): 2D graphics APIs
Processing: simplifed API, large user community, Java-based
Flash: vector graphics engine, Flex UI toolkit, 3rd party libraries

Prefuse / Flare: 2D InfoVis toolkits (Java / Flash)
InfoVis Toolkit: 2D Java toolkit, data mgmt and vis "widgets"
VTK: 2D/3D scientific visualization toolkit



How to support diverse visualizations?



What should tools provide?

(Class Exercise)

Needs of an InfoVis Framework?

Most **user interface tools** provide unified structures for *graphics* and *interaction*

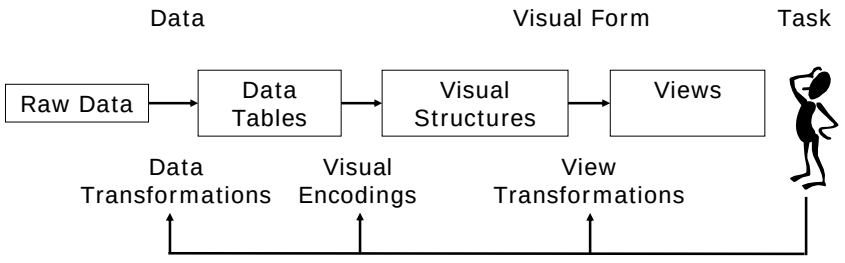
Visualization frameworks must also consider*:*

Data modeling, Visual encoding, Layout

View transformations, Navigation, Animation

Interaction techniques (dynamic queries…)

The InfoVis Reference Model



Integrate data handling, graphics, and interaction

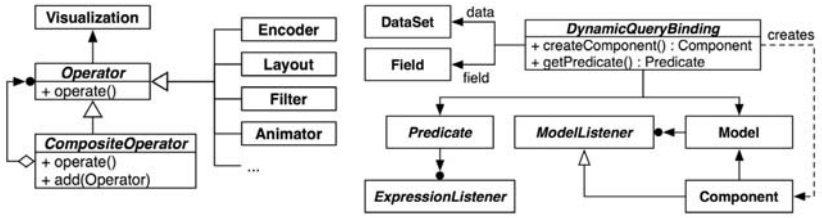
Extensible operator language for visual encodings

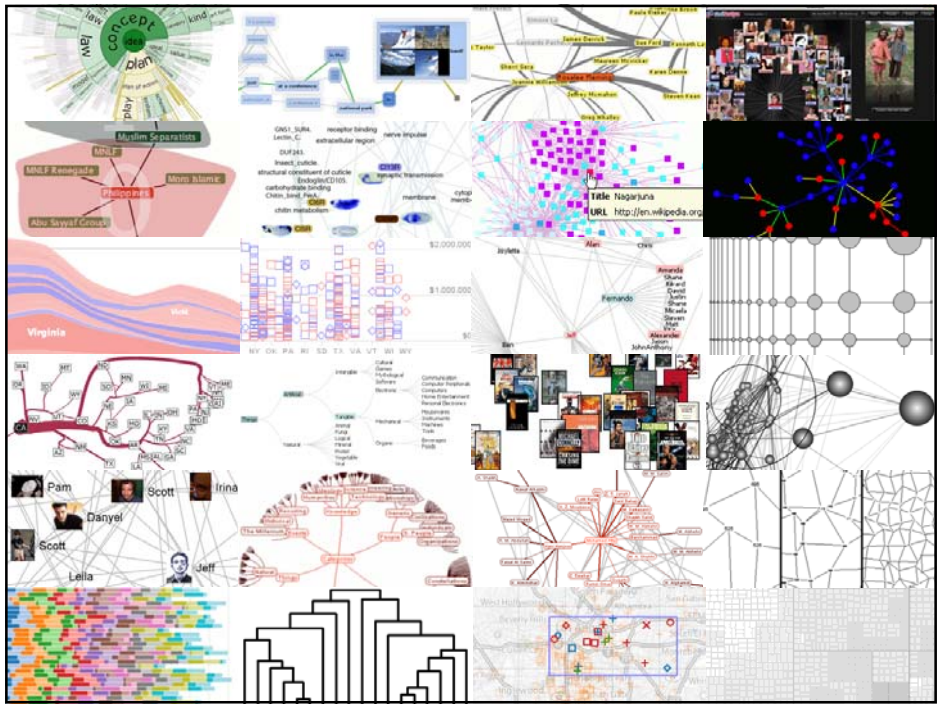
Exploration via dynamic queries, search, zooming

The Prefuse and Flare Toolkits

Open Source projects 70,000+ downloads
Used by students, researchers, corporations

Design Patterns describing architecture [InfoVis 06]





Flare Visualization Toolkit

Web Page: http://flare.prefuse.org
Demos: http://flare.prefuse.org/demo
Tutorial: http://flare.prefuse.org/tutorial

Tutorial Code Examples:
http://www.cs.berkeley.edu/~jheer/tutorial/

Example 1: Flash, Sprites, and Animation

Animation

Interpolate ("tween") visual variables over time:

start value, end value, duration, easing

Transitioner simplifies animation authoring by collecting values and creating "tweens"

```
// Animate color change for all items
var t:Transitioner = new Transitioner(2);
for each (var n:NodeSprite in vis.data.nodes)
  t.$(n).fillColor = 0xff0000ff;
t.play();

// Animate result of running encoding operators
vis.update(new Transitioner(2)).play();
```

Example 2: Visualizing Tabular Data

Data Management

Load external data to an internal representation

Flare supports tabular and tree/graph data

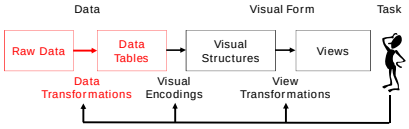
Formats include Tab delimited, JSON, GraphML

Data Table

```
var rows:Array =
[ {id:1, a:0.5, b:2.4, c:3.1},
  {id:2, a:4.7, b:3.1, c:4.3},
  {id:3, a:3.2, b:1.4, c:2.3}, …];
```

Edge Table

```
var edges:Array =
[ {source:1, target:2},
  {source:1, target:3},
  {source:2, target:3}, …];
```



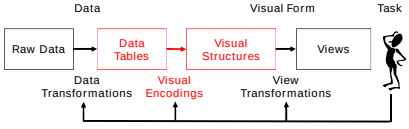
Visual Objects: DataSprites

Associate data with visual objects

Flare **Data** class: container of **DataSprite** items

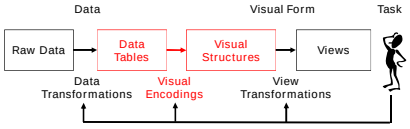
- **NodeSprite** -> table rows, graph nodes
- **EdgeSprite** -> links between nodes

```
// creates visualized data
var data:Data = new Data();
for each (var obj:Object in rows) {
  // adds a NodeSprite ns with
  // ns.data == obj
  data.addNode(obj);
}
```



Manipulating Data Lists

Access Data List:	`data.nodes; data.edges; data.group("name")`
Access By Index:	`data.nodes[0]  data.nodes.length`
Iteration:	`for each (n in data.nodes) …`
Visitation:	`data.nodes.visit(function(n:NodeSprite) {…`
Batch Set Values:	`data.edges["lineWidth"] = 2;`
Multiple Values:	`data.edges.setProperties({ lineWidth: 2, lineColor: 0xffff0000 } [, new Transitioner(2)]);`

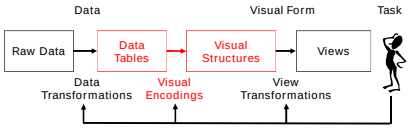


Visualization

A **Visualization** manages a **Data** collection

Visualizations contain multiple layers:

- axes: optional chart axes
- marks: sorted container of DataSprites
- labels: optional layer for text labels
- other layers can be added (e.g., maps)

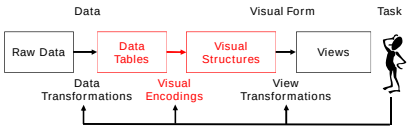


Visual Encoding

A **Visualization** manages the visual DataSprites

Encoding **operators** map data variables to visual variables such as *position*, *size*, *shape*, *color*, …

```
// creates visualization and sets encoding operators
var vis:Visualization = new Visualization(data);
vis.operators.add(new AxisLayout(“data.a”, “data.b”));
vis.operators.add(new ShapeEncoder(“data.c”));
vis.update();
```

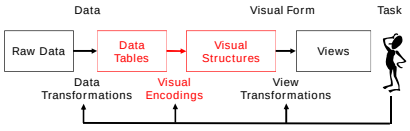


Visual Encoding

A **Visualization** manages the visual DataSprites

Encoding **operators** map data variables to visual variables such as *position*, *size*, *shape*, *color*, …

```
// creates visualization and sets encoding operators
var vis:Visualization = new Visualization(data);
vis.operators.add(new ForceDirectedLayout());
vis.operators.add(new ShapeEncoder(“data.c”));
vis.continuousUpdates = true;
```



Interaction

User input to change visualization state

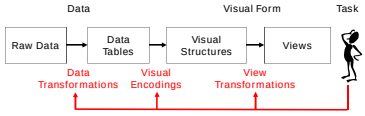
View Transformation: Panning and Zooming

Visual Encoding: Change visual mappings

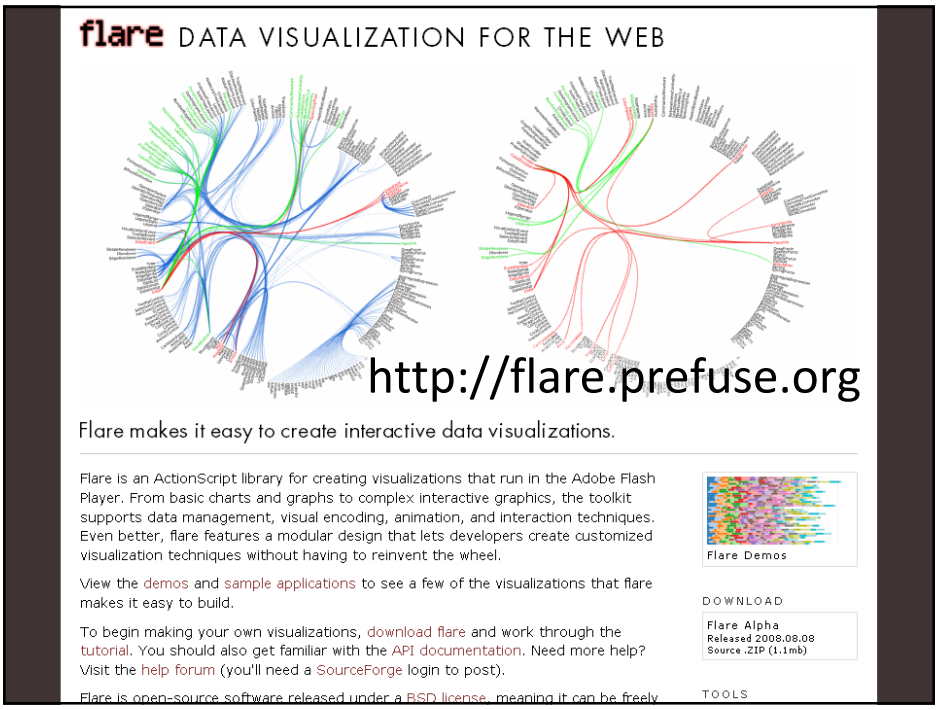
Data Transformation: Filter, dynamic query

```
// creates visualization and sets encoding operators
vis.controls.add(new DragControl(NodeSprite));
vis.controls.add(new PanZoomControl());
vis.controls.add(new HoverControl());

vis.controls[2].onRollOver =
 function(d:DataSprite):void {
  // define response to mouse-over
 };
```



Example 3: Visualizing Network Data



Visualization Programming with Flare

Jeffrey Heer UC Berkeley
jheer@cs.berkeley.edu
jheer.net | vis.berkeley.edu