Identifying Design Principles

Maneesh Agrawala

CS 448B: Visualization
Spring 2016

Announcements

Final project

Design new visualization method (e.g. software)

- Pose problem, Implement creative solution
- Design studies/evaluations less common but also possible (talk to us)

Deliverables

- Implementation of solution
- 6-8 page paper in format of conference paper submission
- Project progress presentations

Schedule

- Project proposal: 5/11
- Project progress presentation: 5/23 in class (3-4 min) slide presentation
- Final poster presentation: 6/3 12:15-3:15pm Location: TBD
- Final paper: 6/5 11:59pm

Grading

- Groups of up to 3 people, graded individually
- Clearly report responsibilities of each member

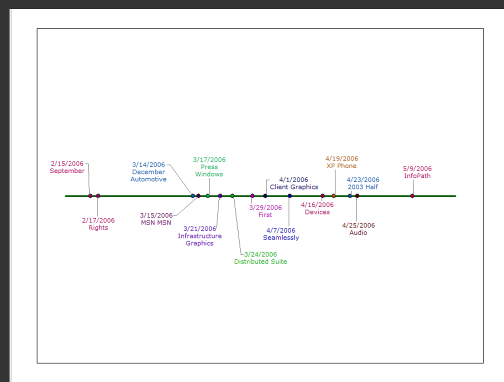
Last Time: Spatial Layout

Problem

Input: Set of graphic elements (scene description)

Goal: Select visual attributes for elements

- **Position**
- Orientation
- Size
- Color
- …



Approaches

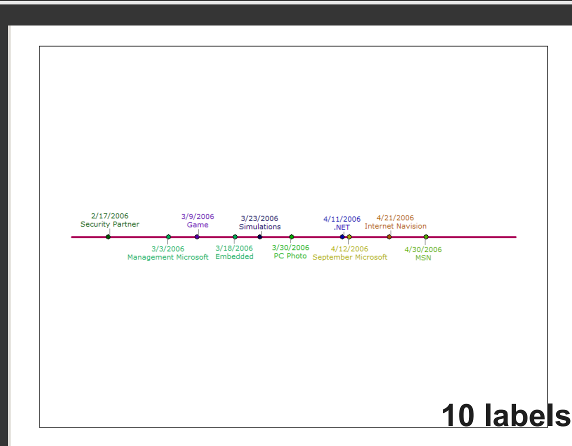
Direct rule-based methods

Constraint satisfaction

Optimization

Direct Rule-Based Methods

Rule-based timeline labeling



- Alternate above/below line
- Center labels with respect to point on line

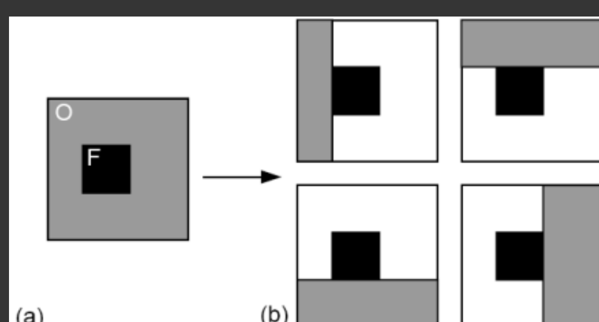
Dynamic space management [Bell 00]

Manage *free space* on desktop to prevent window overlap

Dynamic space management [Bell 00]

Goal: Place new elements to avoid overlap

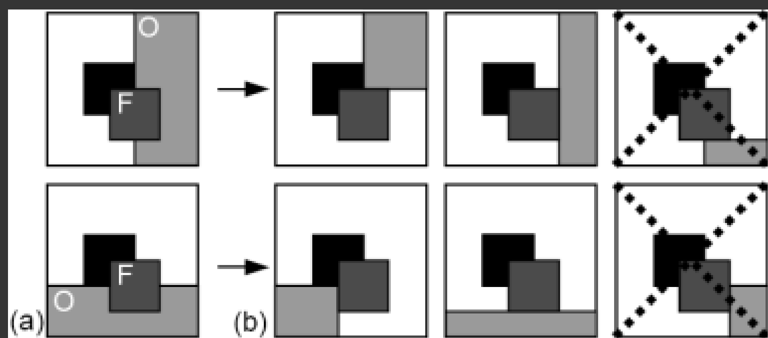
- Elements are axis-aligned rectangles
- Keep track of largest empty space rectangles



Dynamic space management [Bell 00]

Goal: Place new elements to avoid overlap

- Elements are axis-aligned rectangles
- Keep track of largest empty space rectangles



Pros and cons

Pros

- **Designed to run extremely quickly**
- **Simple layout algorithms are easy to code**

Cons

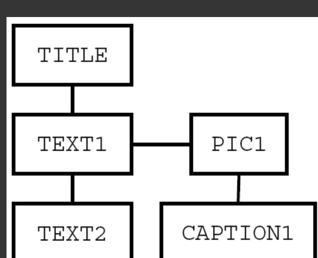
- **Complex layouts require large rule bases with lots of special cases**

Linear Constraint Satisfaction

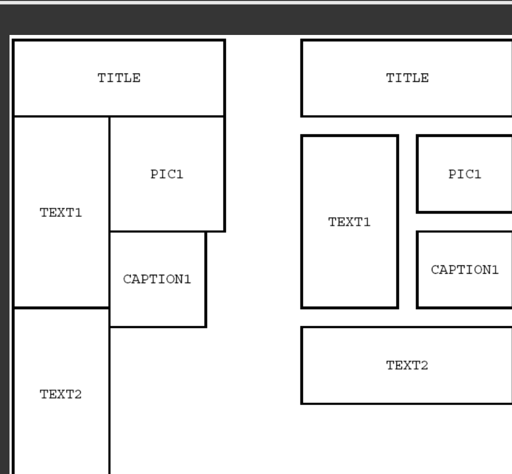
Network of layout constraints

TITLE ABOVE TEXT1
TITLE FULL PAGE WIDTH
TEXT1 LEFT OF PIC1
CAPTION1 BELOW PIC1
TEXT2 BELOW TEXT1

Constraints



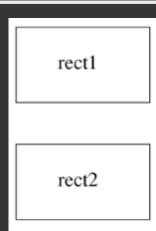
Network



Two possible layouts

[from Lok and Feiner 01]

Constraints as linear equations



C1: rect2.top = rect1.top + rect1.height + 10

C2: rect2.height = rect1.height

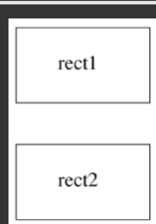
C3: rect2.bottom = rect2.top + rect2.height

Local propagation
- Set any variable
- Update other variables to maintain constraints

One-way
- Each constraint has 1 output variable
- Update output when any input changes

Multi-way
- Each constraint can be written so that any variable is output
- More complicated to maintain

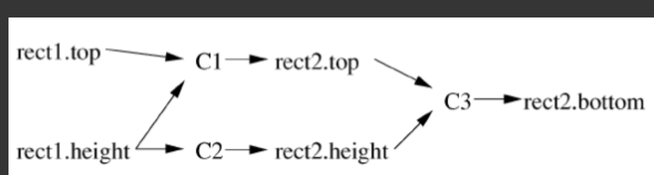
One-way constraints



C1: rect2.top = rect1.top + rect1.height + 10

C2: rect2.height = rect1.height

C3: rect2.bottom = rect2.top + rect2.height

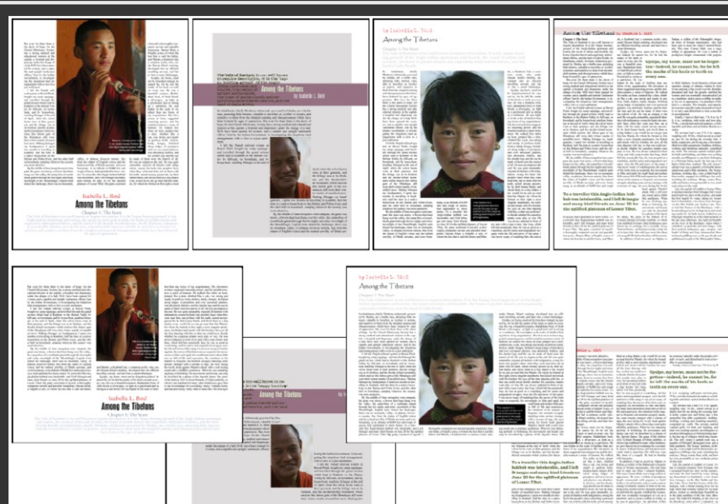


One-way constraints form a directed acyclic graph (DAG). Given the value for any variable we propagate it's value locally through the graph updating the other variable.

Page layout example [Weitzman and Wittenburg 94]

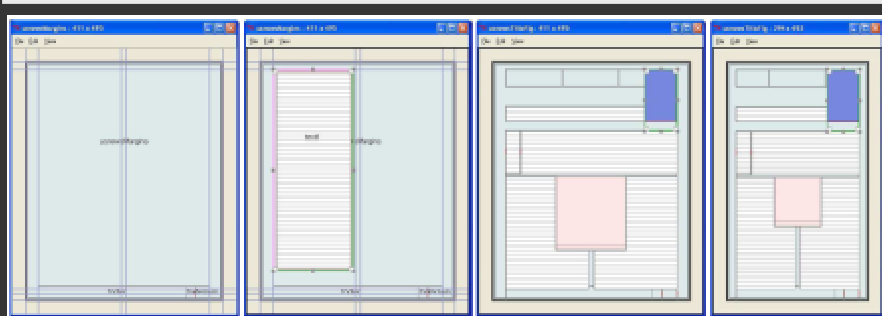
```
(Defrule (Make-Article The-Grammar)
  Article -> Text Text Text Number Image
    0         1    2    3     4      5
  (Author-Of 2 1)
  (Description-Of 4 1)
  (Page-Of 4 1)
  (Image-Of 5 1)
  (article-name 0) = r
  (article-image 0) = 5
  :OUT
  (right-of 1 5)
  (top-aligned 1 5)
  (top-aligned 5 4)
  (spaced-below 2 1)
  (spaced-below 3 2)
)
```

Adaptive document layout [Jacobs 03]



Users authors templates which use one-way constraints to adapt to changes in page size

ADL template authoring [Jacobs 03]



Adaptive Grid~Based Document Layout

Chuck Jacobs[1] Wilmot Li[2] Evan Schrier[2]
David Bargeron[1] David Salesin[1,2]

[1]Microsoft Research [2]University of Washington

Pros and cons

Pros

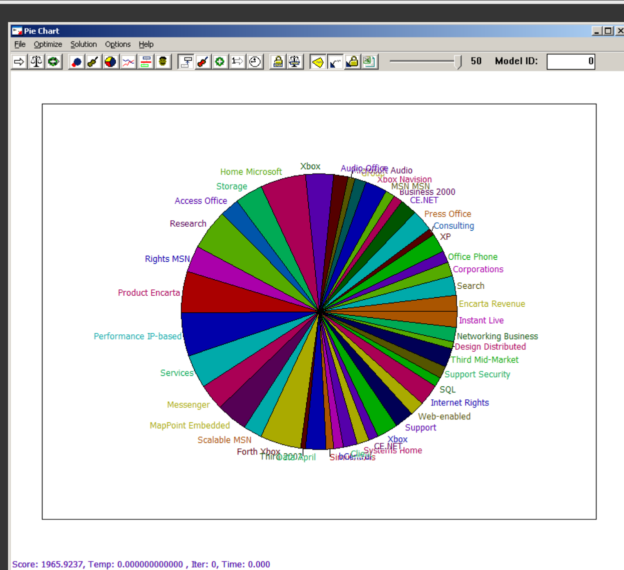
- Often run fast (at least one-way constraints)
- Constraint solving systems are available online
- Can be easier to specify relative layout constraints than to code direct layout algorithm

Cons

- Easy to over-constrain the problem
- Constraint solving systems can only solve some types of layout problems
- Difficult to encode desired layout in terms of mathematical constraints

Optimization

Demo



Layout as optimization

Scene description

- **Geometry:** polygons, bounding boxes, lines, points, etc.
- **Layout parameters:** position, orientation, scale, color, etc.

Large design space of possible layouts

To use optimization we will specify …

- **Initialize/Perturb functions:** Form a layout
- **Penalty function:** Evaluate quality of layout
- **.. and find layout that minimizes penalty**

Optimization algorithms

There are lots of them:

line search, Newton's method, A*, tabu, gradient descent, conjugate gradient, linear programming, quadratic programming, simulated annealing, …

Differences

- Speed
- Memory
- Properties of the solution
- Requirements

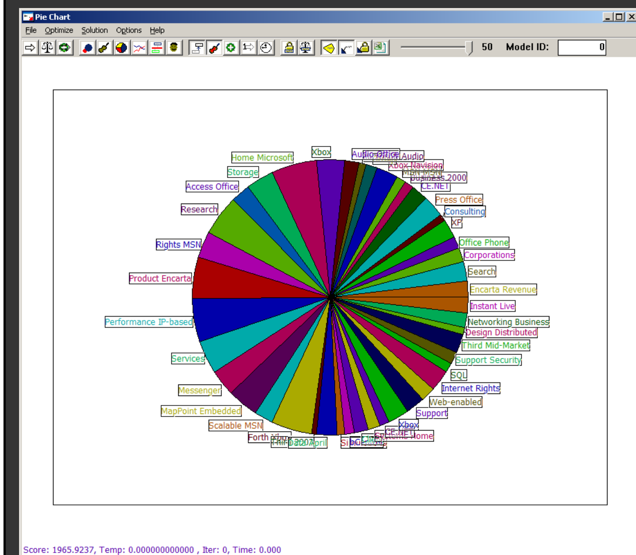
Simulated annealing

currL ← Initialize()	Form initial layout
while(**! termination condition)**	
newL ← Perturb(***currL***)	Perturb to form new layout
currE ← Penalty(***currL***)	Evaluate quality of layouts
newE ← Penalty(***newL***)	
if**((*newE* < *currE*) or**	Always accept lower penalty
(rand[0,1) < $e^{-\Delta E/T}$**))**	Small probability of accepting higher penalty
then ***currL*** ← ***newL***	
Decrease(*T*)	

Perturb: Efficiently cover layout design space

Penalty: Describes desirable/undesirable layout features

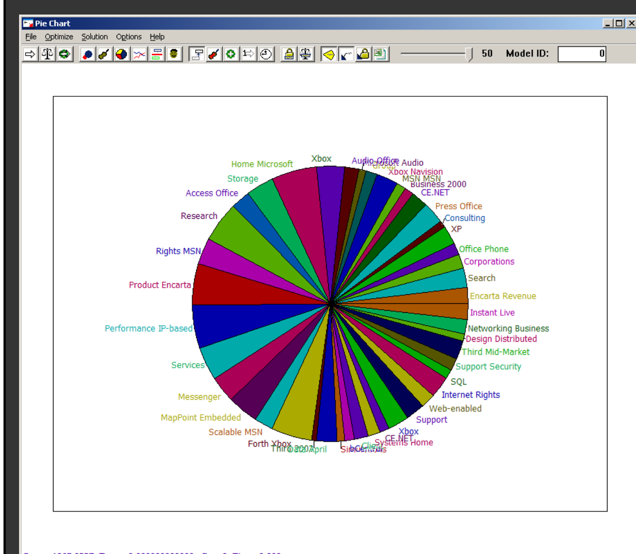
Scene description



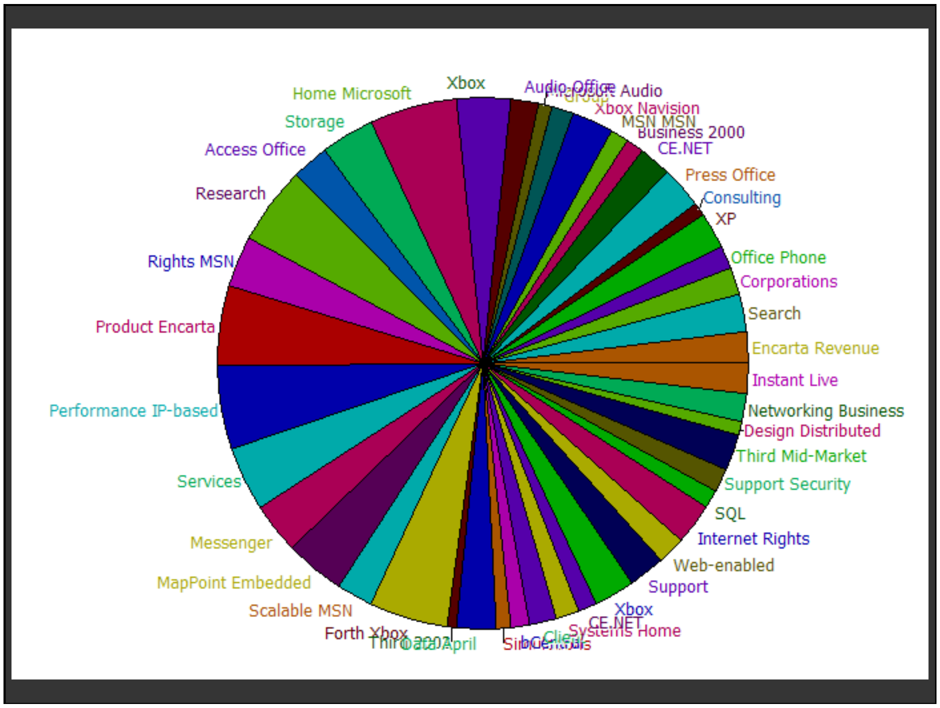
Geometry

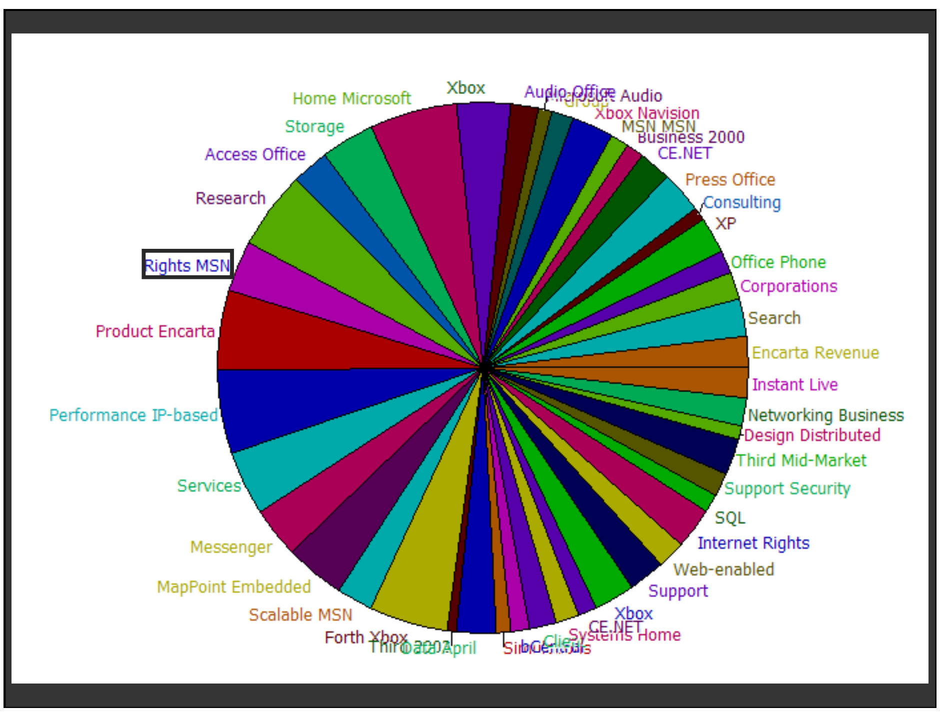
- **Pie slices**
  - anchors for labels
- **Labels**
  - bounding boxes

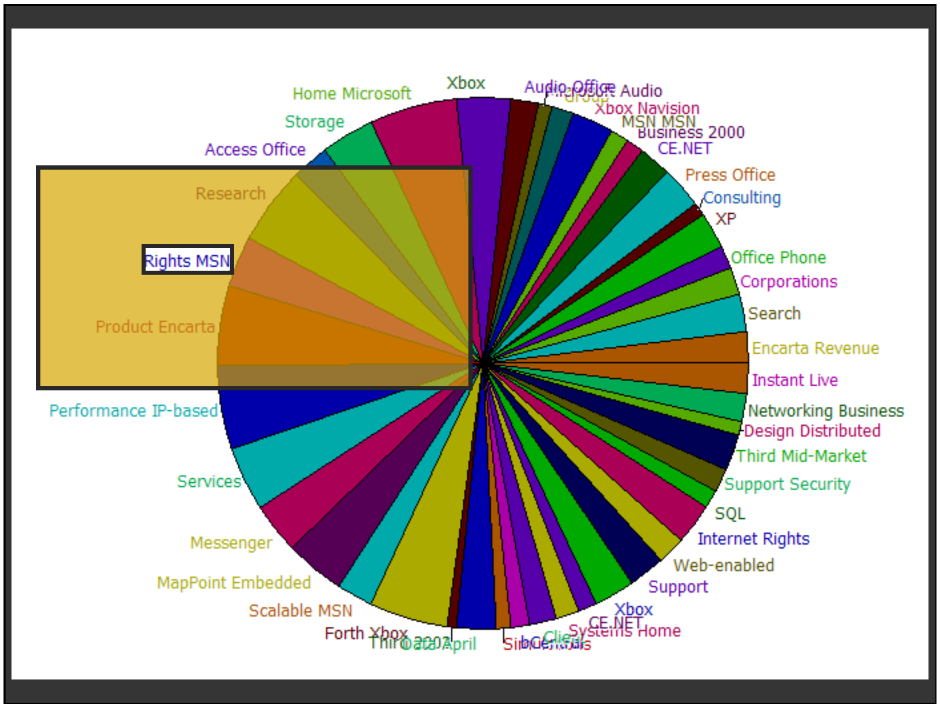
Layout parameters

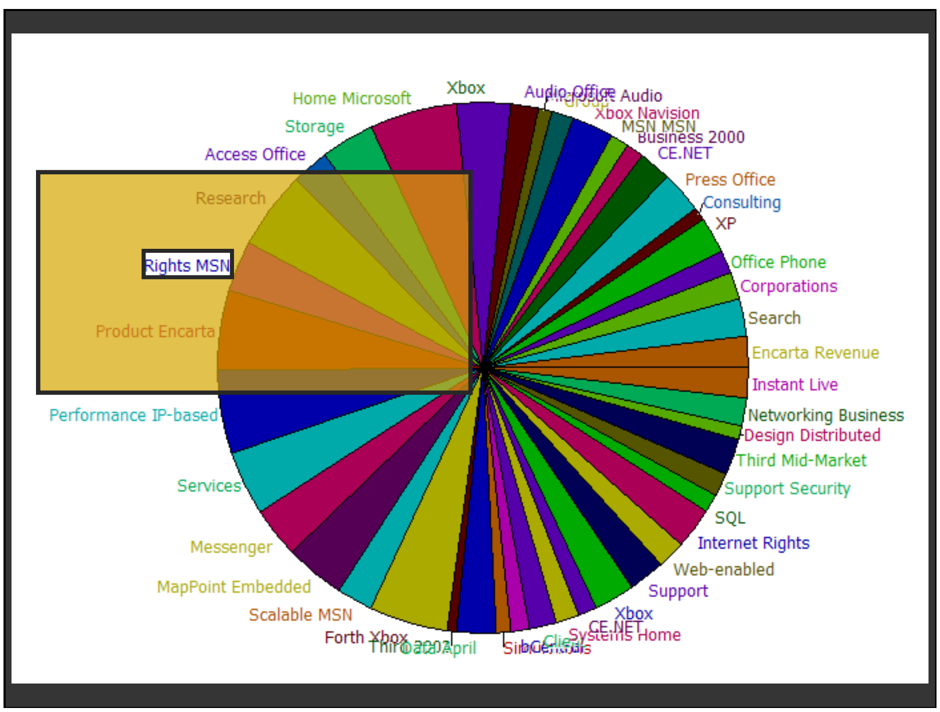


- **Position *(x, y)***
- **Leader line**
- **Word wrap**
- Color
- Alignment
- Orientation
- Scale

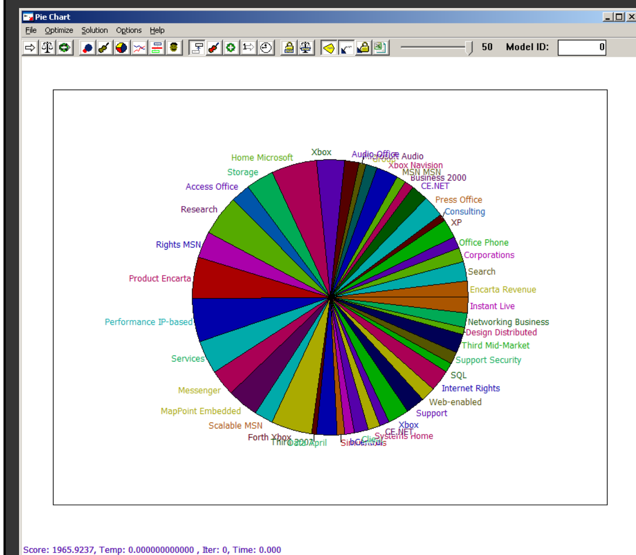

Home Microsoft
Xbox
Audio
Xbox Navision
MSN MSN
Business 2000
CE.NET
Storage
Access Office
Research
Press Office
Consulting
XP
Rights MSN
Office Phone
Corporations
Search
Product Encarta
Encarta Revenue
Instant Live
Performance IP-based
Networking Business
Design Distributed
Third Mid-Market
Services
Support Security
SQL
Messenger
Internet Rights
Web-enabled
MapPoint Embedded
Support
Scalable MSN
Xbox
CE.NET
Forth Xbox
Systems Home
April


Home Microsoft
Xbox
Audio
Xbox Navision
MSN MSN
Business 2000
CE.NET
Storage
Access Office
Research
Press Office
Consulting
XP
Rights MSN
Office Phone
Corporations
Search
Product Encarta
Encarta Revenue
Instant Live
Performance IP-based
Networking Business
Design Distributed
Third Mid-Market
Services
Support Security
SQL
Messenger
Internet Rights
Web-enabled
MapPoint Embedded
Support
Scalable MSN
Xbox
CE.NET
Forth Xbox
Systems Home
April


Home Microsoft
Xbox
Storage
Access Office
Research
Rights MSN
Product Encarta
Performance IP-based
Services
Messenger
MapPoint Embedded
Scalable MSN
Forth Xbox
April
Audio
Xbox Navision
MSN MSN
Business 2000
CE.NET
Press Office
Consulting
XP
Office Phone
Corporations
Search
Encarta Revenue
Instant Live
Networking Business
Design Distributed
Third Mid-Market
Support Security
SQL
Internet Rights
Web-enabled
Support
Xbox
CE.NET
Systems Home


Home Microsoft
Xbox
Storage
Access Office
Research
Rights MSN
Product Encarta
Performance IP-based
Services
Messenger
MapPoint Embedded
Scalable MSN
Forth Xbox
April
Audio
Xbox Navision
MSN MSN
Business 2000
CE.NET
Press Office
Consulting
XP
Office Phone
Corporations
Search
Encarta Revenue
Instant Live
Networking Business
Design Distributed
Third Mid-Market
Support Security
SQL
Internet Rights
Web-enabled
Support
Xbox
CE.NET
Systems Home

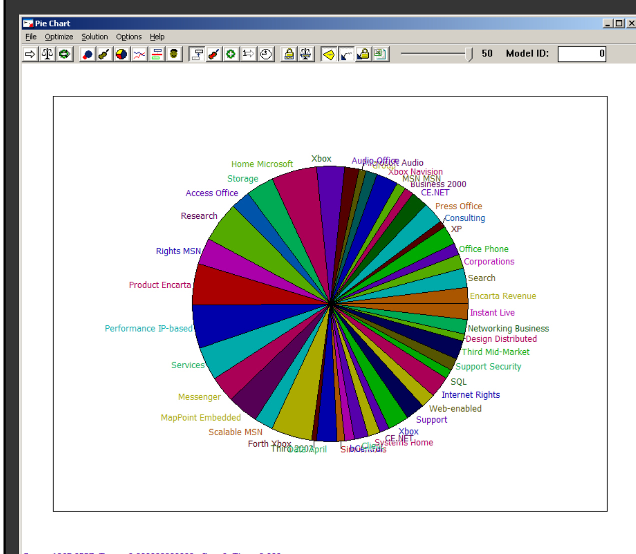
Many dimensions → large space



- **Position *(x, y)***
- Leader line
- Word wrap
- Color
- Alignment
- Orientation
- Scale

2D x 50 labels → 100D space

Penalties



Overlap & Distance

- Label – anchor slice
- Label – other slices
- Label – label

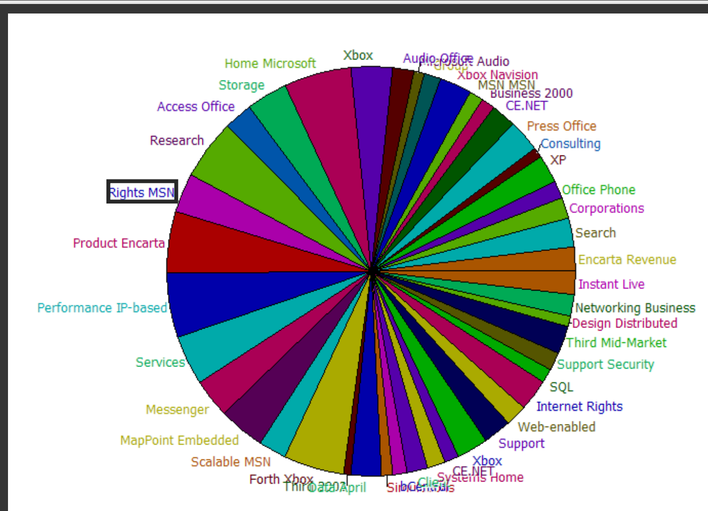
Leader lines

- Length
- Intersections

Word Wrap

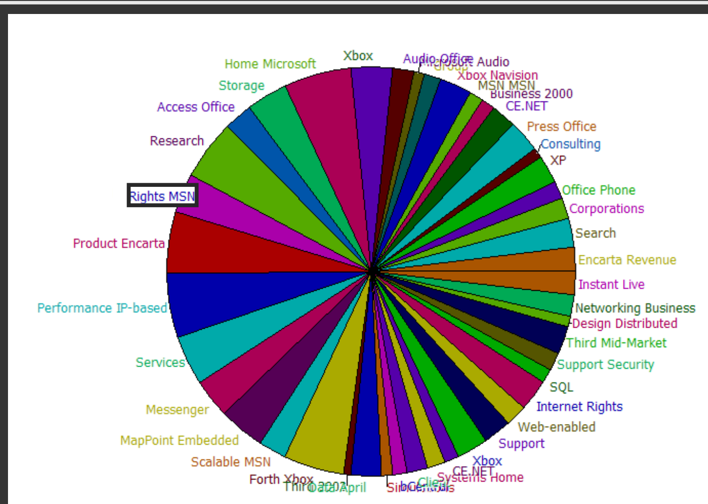
Annealing minimizes sum of all penalties

Overlap: Label – Anchor Slice



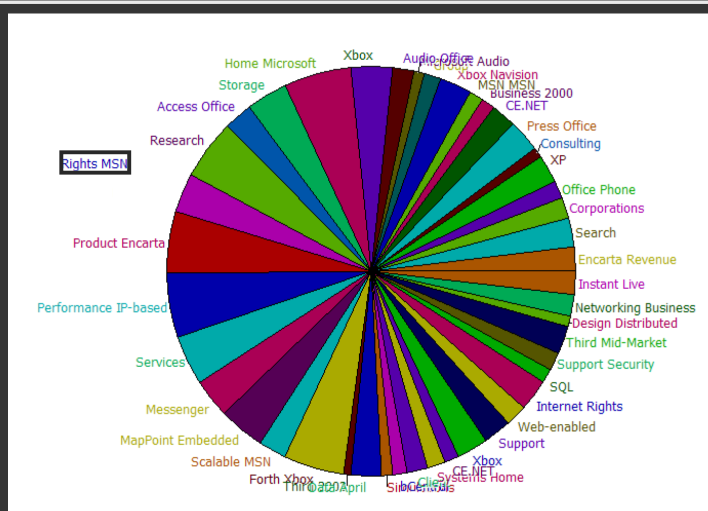
Avoid partial overlap: No penalty if fully inside /outside

Overlap: Label – Anchor Slice



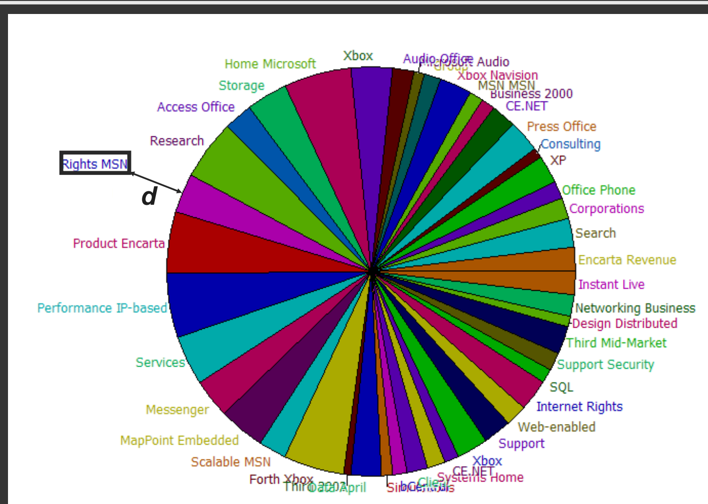
Penalize partial overlap by overlap amount

Distance: Label – Anchor Slice



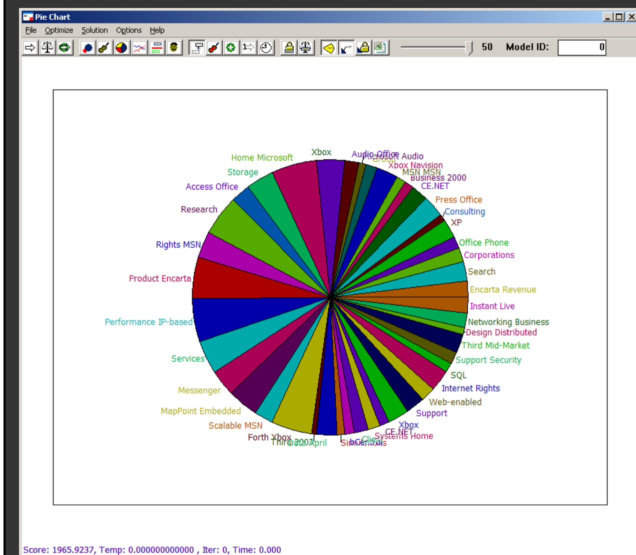
Ensure label near center of edge of anchor slice

Distance: Label – Anchor Slice



Minimize distance *d*

Penalties



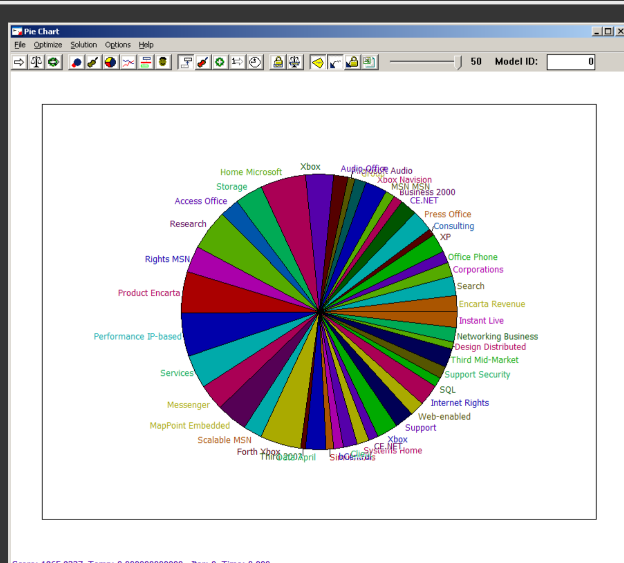
Overlap & Distance
- Label – anchor slice
- Label – other slices
- Label – label

Leader lines
- Length
- Intersections

Word Wrap

Annealing minimizes sum of all penalties

Demo



Pros and cons

Pros

- Much more flexible than linear constraint solving systems

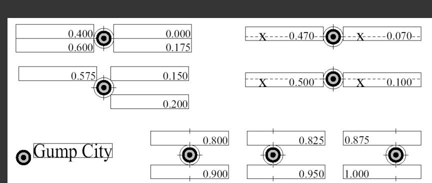
Cons

- Can be relatively slow to converge
- Need to set penalty function parameters (weights)
- Difficult to encode desired layout in terms of mathematical penalty functions

Design principles

Sometimes specified in design books

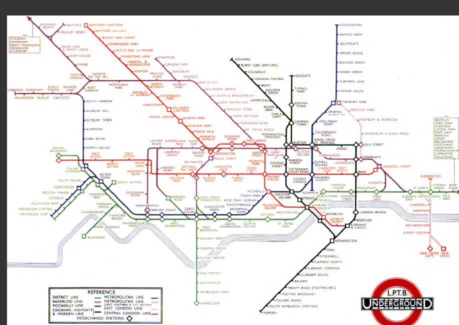
- Tufte, Few, photography manuals, cartography books …
- Often specified at a high level
- Challenge is to transform principles into constraints or penalties



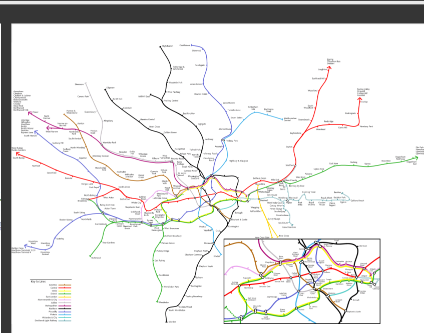
Cartographer Eduard Imhof's labeling heurists transformed into penalty functions for an optimization based point labeling system [Edmondson 97]

Identifying Design Principles

Good Design Improves Effectiveness

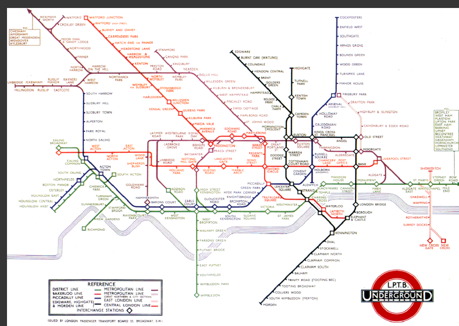


London Underground [Beck 33]

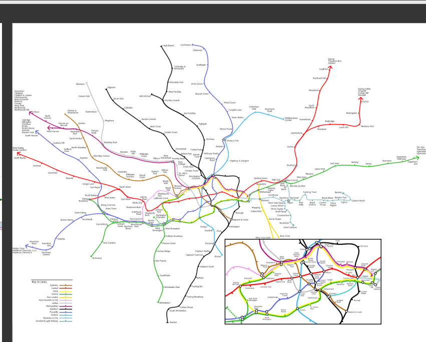


Geographic version of map

Good Design Improves Effectiveness



London Underground [Beck 33]



Geographic version of map

Design principle:

- Straighten lines to emphasize sequence of stops

Technique used to emphasize/de-emphasize information

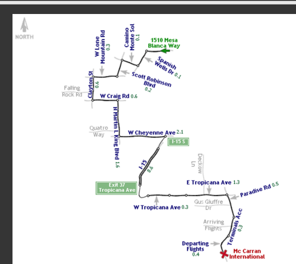
Approach

Identify design principles

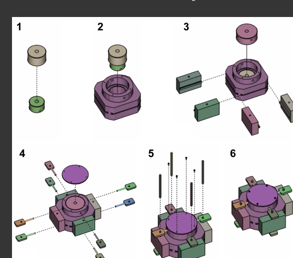
- Cognition and perception

Instantiate design principles

- Principles become constraints that guide an optimization process



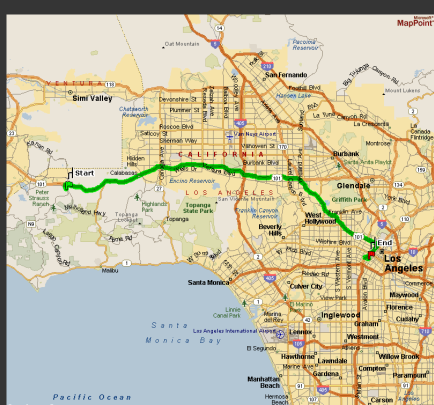
Route maps



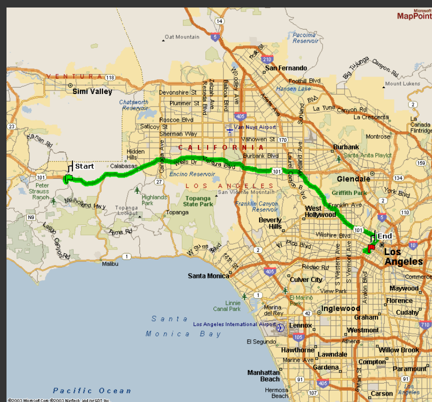
Assembly instructions

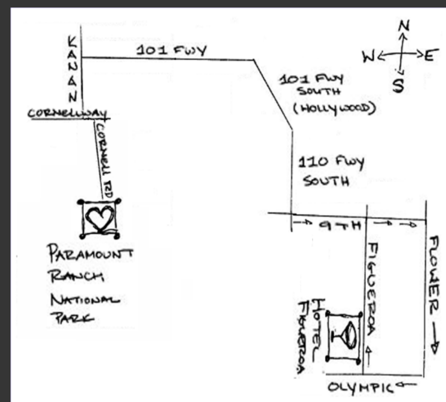
Route Maps

Visualizing Routes

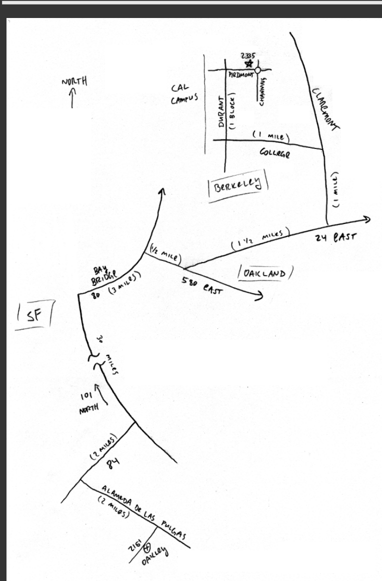


A Better Visualization





Cognition of Route Maps



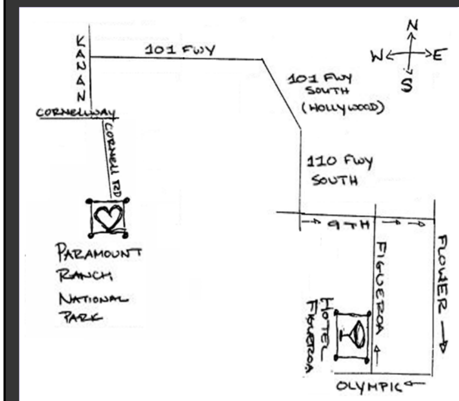
Essential information
- Turning points
- Route topology

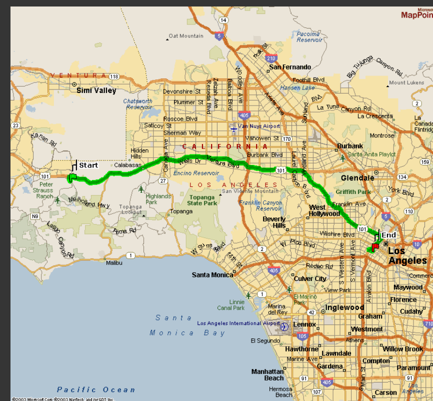
Secondary context information
- Local landmarks, cross streets, etc.
- Overview area landmarks, global shape

Exact geometry less important
- *Not* apprehended accurately
- *Not* drawn accurately

[Tversky 81] [Tufte 90] [Tversky 92]
[MacEachren 95] [Denis 97] [Tversky 99]

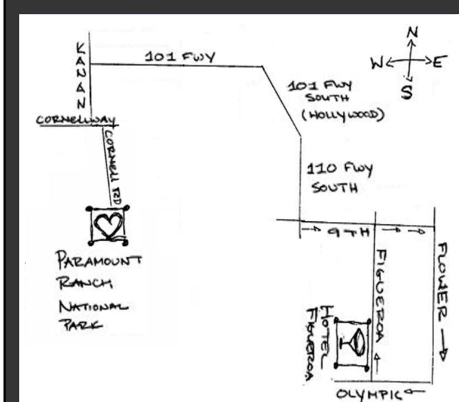
Design Principles



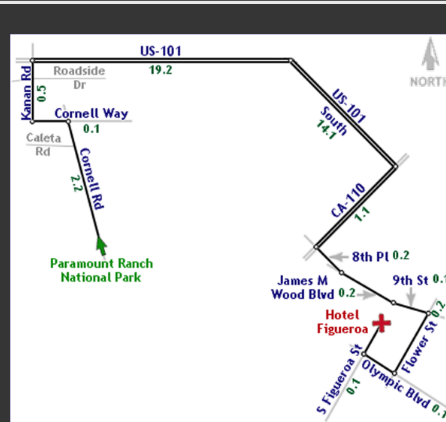


- Exaggerate road length
- Regularize turning angles
- Simplify road shape

LineDrive



Hand-drawn route map



LineDrive route map

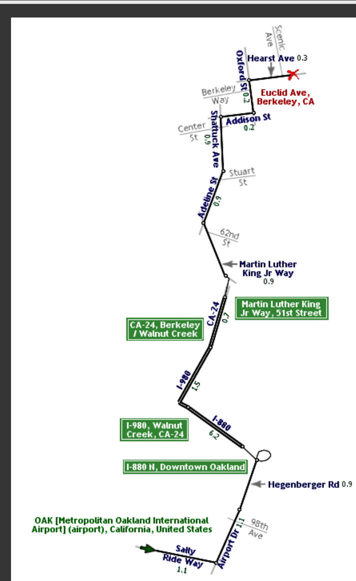
Map Design via Optimization

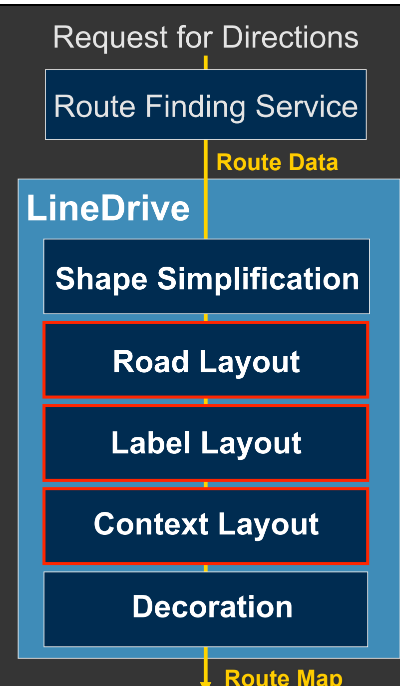
Set of graphic elements
- Roads, labels, cross-streets, …

Choose visual attributes
- Position, orientation, size, …
- Distortions increase flexibility

Develop constraints based on design principles

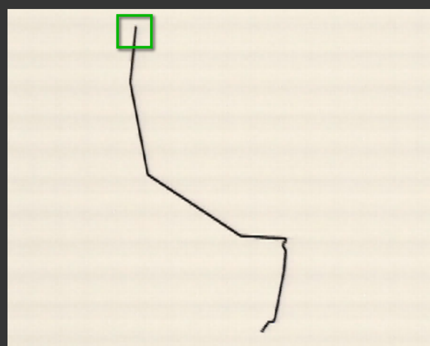
Simulated annealing
- Perturb: Form a layout
- Score: Evaluate quality
- Minimize score



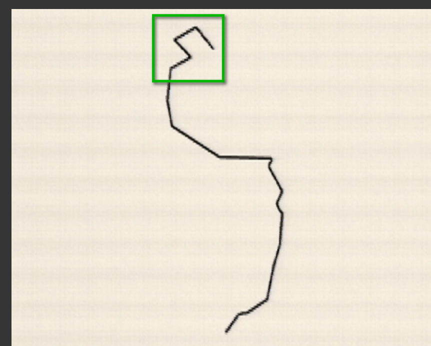


Road Layout

Choose road lengths and orientations



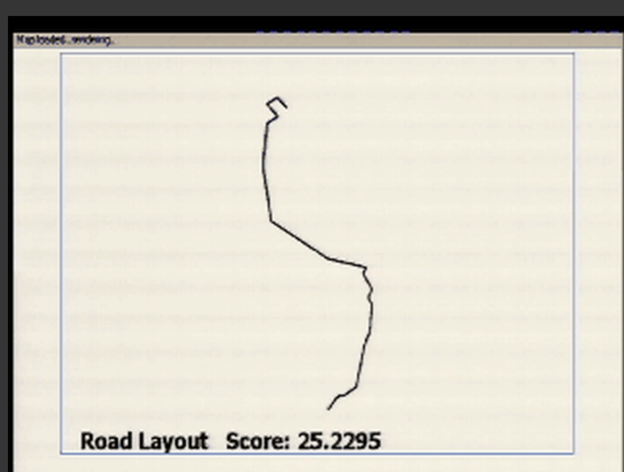
Before road layout



After road layout

Road Layout

Choose road lengths and orientations



Road Layout Constraints

Length

- Ensure all roads visible — $((L_{min} - l(r_i))/ L_{min})^2 * W_{small}$
- Maintain ordering by length — $W_{shuffle}$

Orientation

- Maintain original orientation — $|\alpha_{curr}(r_i) - \alpha_{orig}(r_i)| * W_{orient}$

Topological errors

- Prevent false — $\min(d_{origin}, d_{dest}) * W_{false}$
- Prevent missing — $d * W_{missing}$
- Ensure separation — $\min(d_{ext}, E) * W_{ext}$

Overall route shape

- Maintain endpoint direction — $|\alpha_{curr}(v) - \alpha_{orig}(v)| * W_{enddir}$
- Maintain endpoint distance — $|d_{curr}(v) - d_{orig}(v)| * W_{enddist}$

Balancing the Constraints

Prioritize scores by importance

1. Prevent topological errors
2. Ensure all roads visible
3. Maintain original orientation
4. Maintain ordering by length
5. Maintain overall route shape

Priorities set based on usability tests

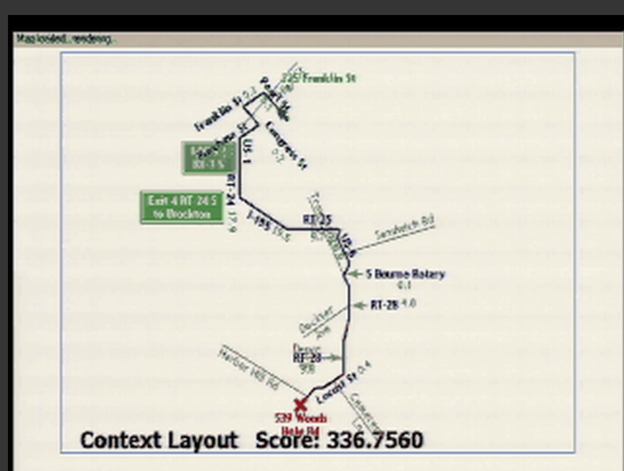
- Users given maps containing errors
- Rated which errors most confusing

Label Layout

Find overlap-free position for each label

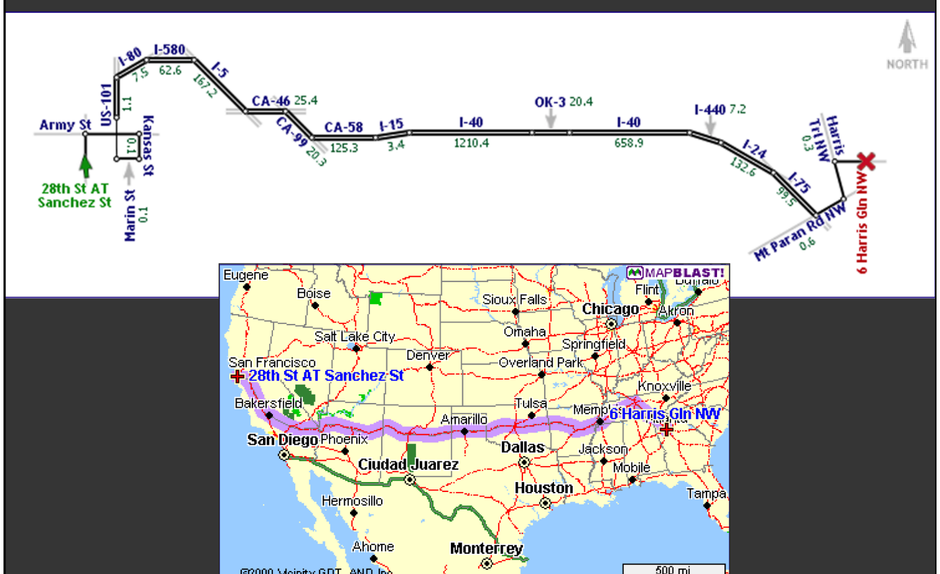
Context Layout

Place cross-streets and exit signs if possible



Bellevue to Seattle

Cross-Country Route



System Performance

7727 routes (sampled over 1 day at MapBlast!)

Median distance	52.5 miles
Median number turning points	13
Median computation time	0.7 sec
Short roads	5.4 %
False intersections	0.3 %
Missing intersections	0.2 %
Label-label overlap	0.5 %
Label-road overlap	11.7 %

Results

Beta version 6 months

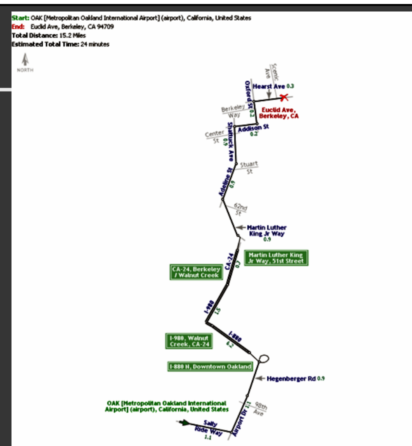
- 150,000 maps served

2242 responses

- Replace standard 55.6 %
- Use with standard 43.5 %
- Prefer standard 0.9 %

At peak

- Deployed at: mappoint.com
- Served 750,000 maps/day
- Taken offline in fall 2011

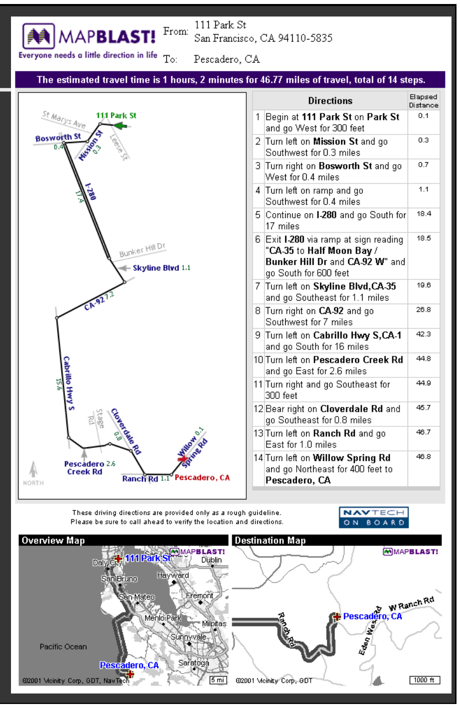


Directions	Miles
Start: Depart OAK [Metropolitan Oakland International Airport] (airport), California, United States on Sally Ride Way (East)	1.0
1: Turn LEFT (North) onto Airport Dr	1.0
2: Turn RIGHT (North) onto Hegenberger Rd	0.8
3: Bear RIGHT (North-West) onto ramp	0.2
4: Continue (North-West) on I-880,Nimitz Fwy	6.1
5: Continue (North) on I-980	1.5
6: Continue (North) on CA-24,Grove Shafter Fwy	0.7
7: Continue (North) on ramp	0.7
8: Bear RIGHT (North) onto Martin Luther King Jr Way	0.9
9: Turn RIGHT (North) onto Adeline St	0.9
10: Bear RIGHT (North) onto Shattuck Ave	0.8
11: Turn RIGHT (East) onto Addison St	0.1
12: Turn LEFT (North) onto Oxford St	0.2
13: Turn RIGHT (East) onto Hearst Ave	0.3
End: Arrive Euclid Ave, Berkeley, CA 94709	

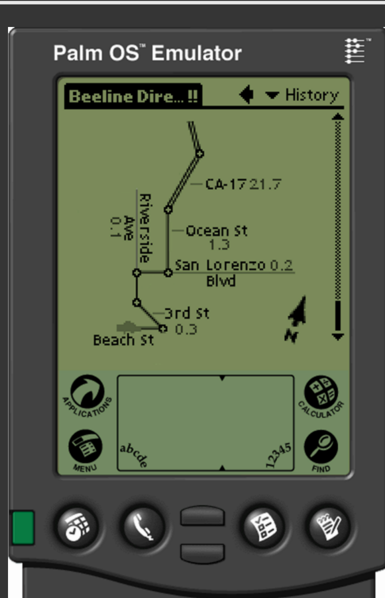
Original Design

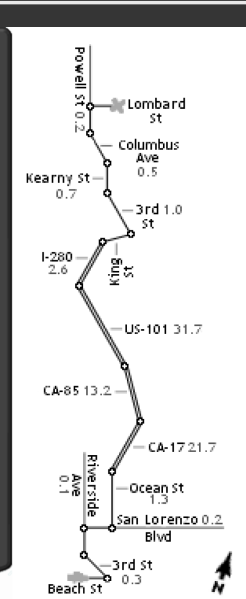
Layout

- Map and text close together
- Overview and destination maps for more content

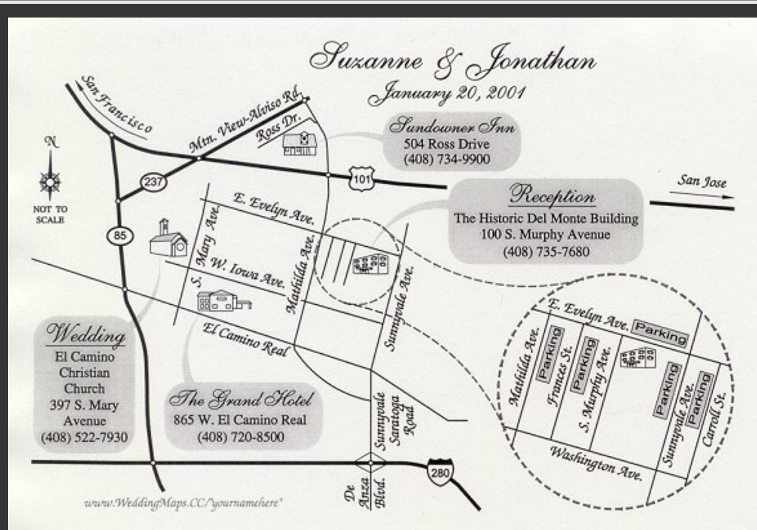


Limited Resolution PDA

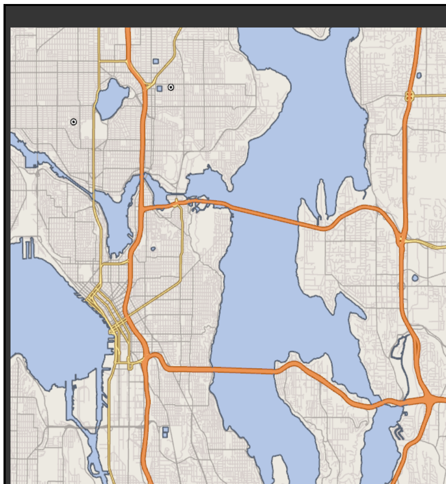




Next Steps: Wedding Maps



Hand-designed Wedding Map www.WeddingMaps.CC

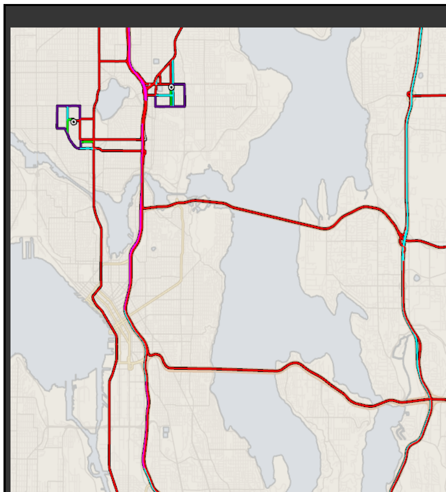


Input map drawn to scale

Our result

1st Ave. and 19th Ave. NW, Seattle WA

http://www.bing.com/maps/explore/#/c7pvw1whdkp6ggvw (Requires Windows, IE, Silverlight)

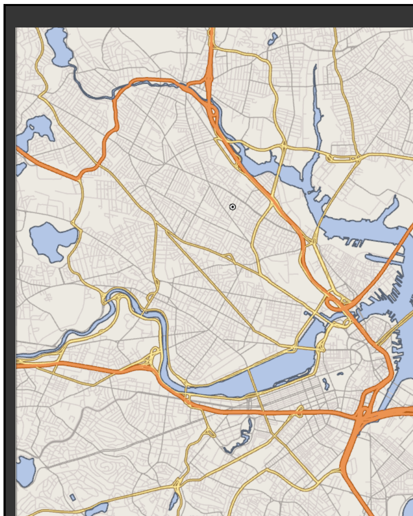


Roads selected from input

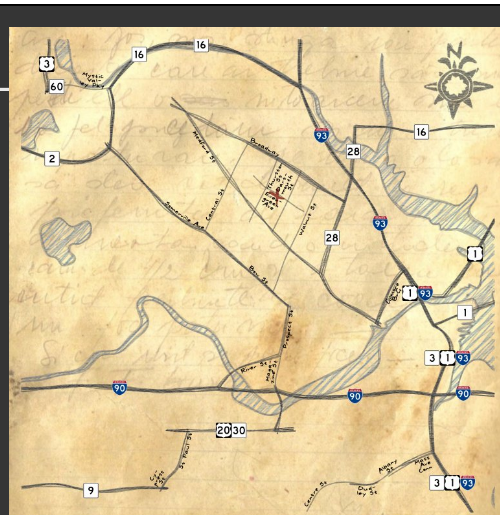
Our result

1st Ave. and 19th Ave. NW, Seattle WA

http://www.bing.com/maps/explore/#/c7pvw1whdkp6ggvw (Requires Windows, IE, Silverlight)



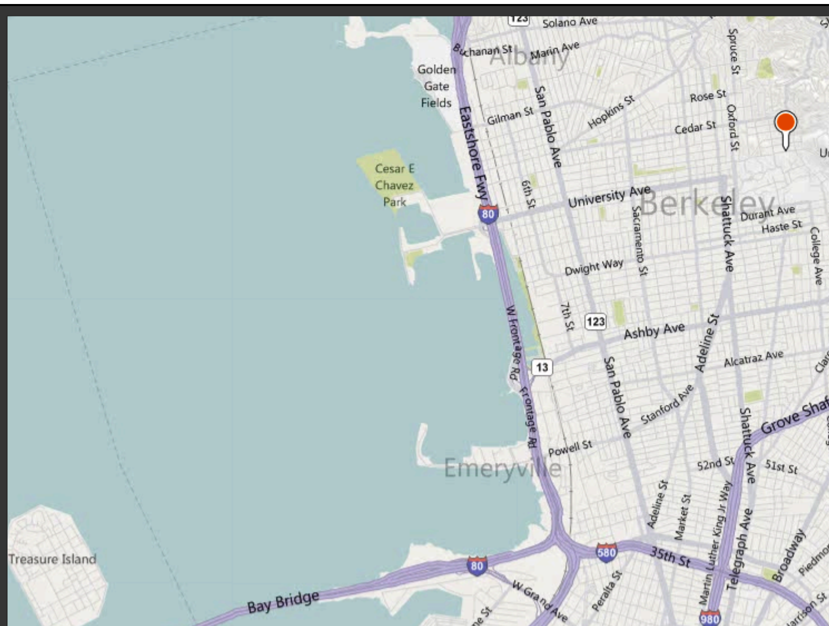
Input map drawn to scale



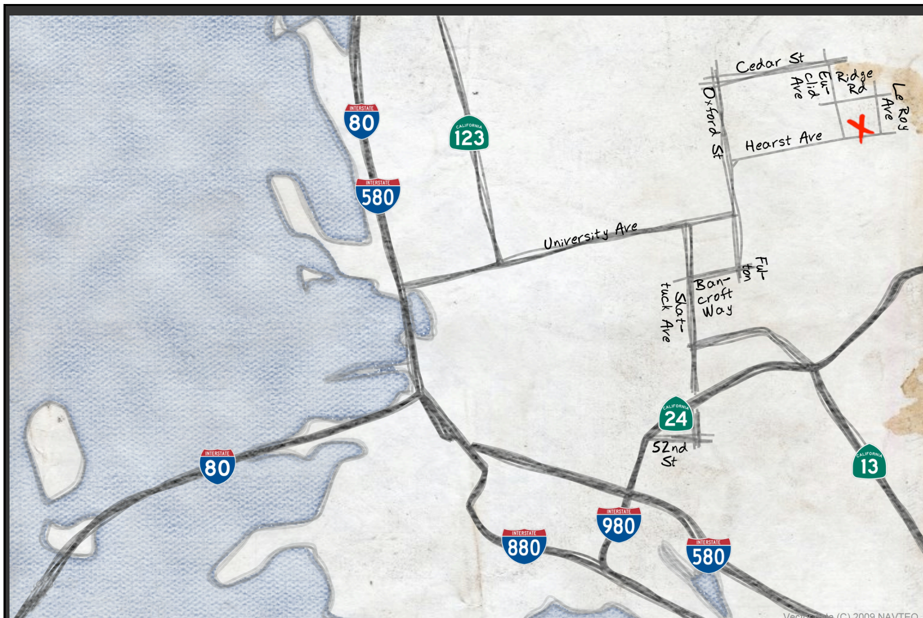
Our result

Evergreen Ave., Boston MA

http://www.bing.com/maps/explore/#/c7pvw1whdkp6ggvw (Requires Windows, IE, Silverlight)

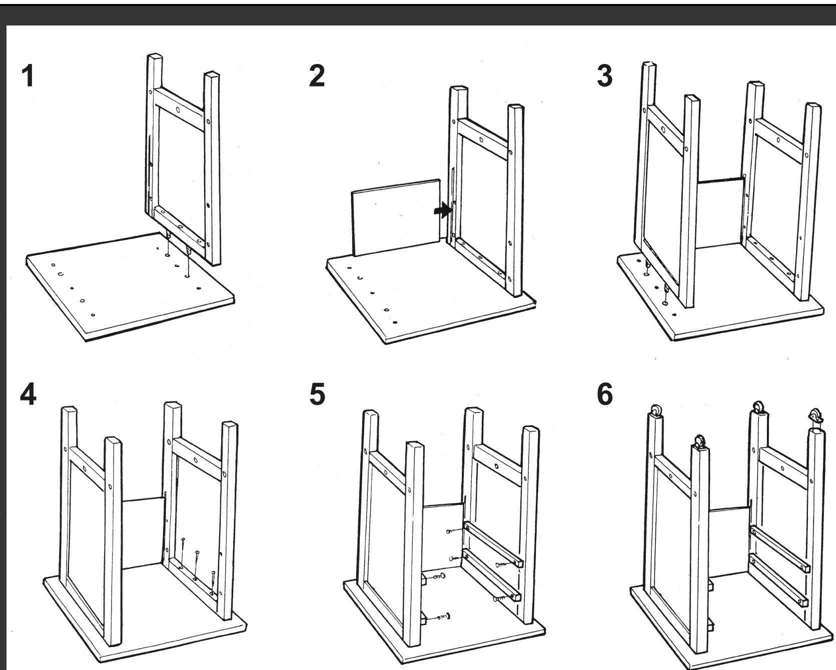


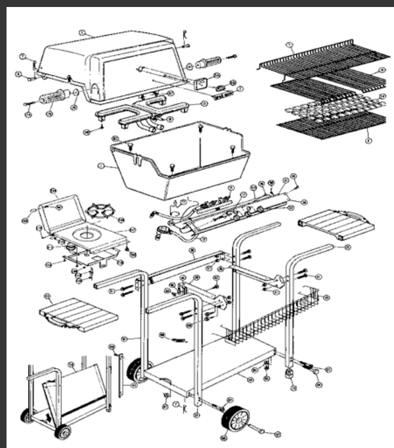
635 Soda Hall, Berkeley CA

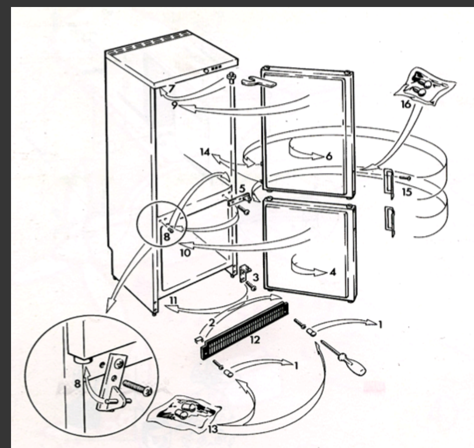


635 Soda Hall, Berkeley CA

Assembly Instructions





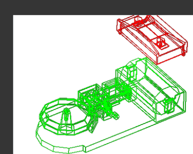


Previous Work

Planning

- Choose sequence of assembly operations
- Robotics / AI / Mechanical Engineering

[Wolter 89], [de Mello 91], [Wilson 92], [Romney 95]



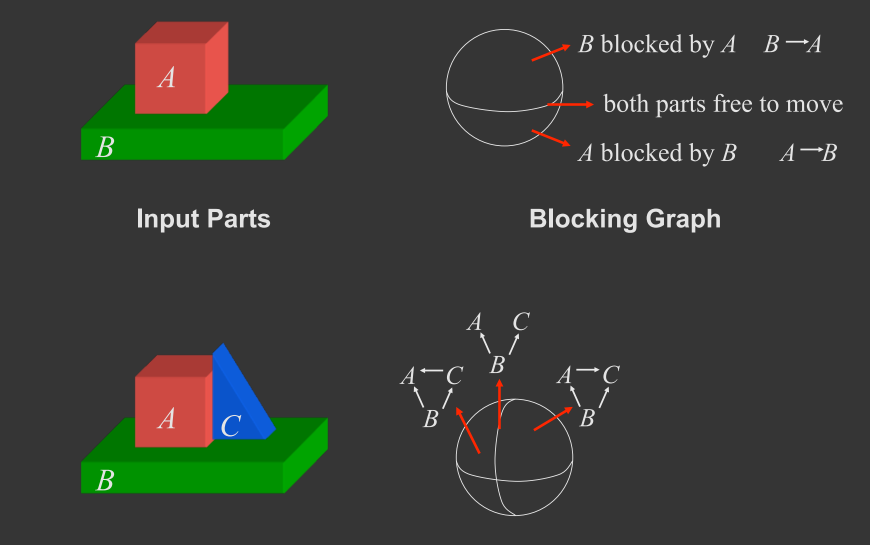
Presentation

- Visually convey assembly operations
- Visualization / Computer Graphics

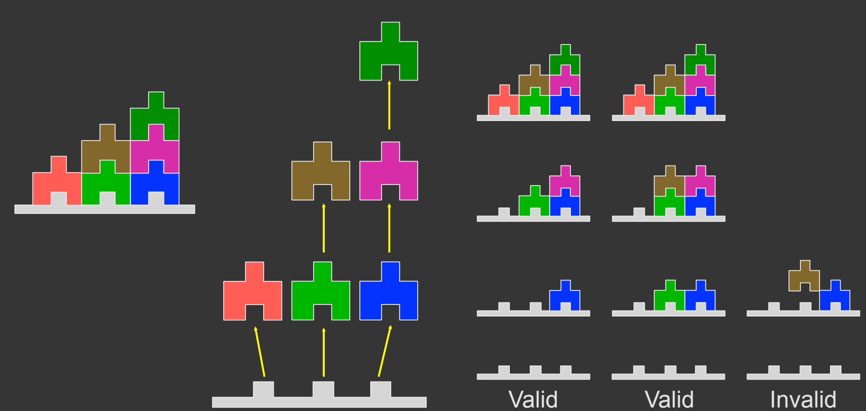
[Seligmann 91], [Rist 94], [Butz 97], [Strothotte 98]

Jointly optimize plan and presentation

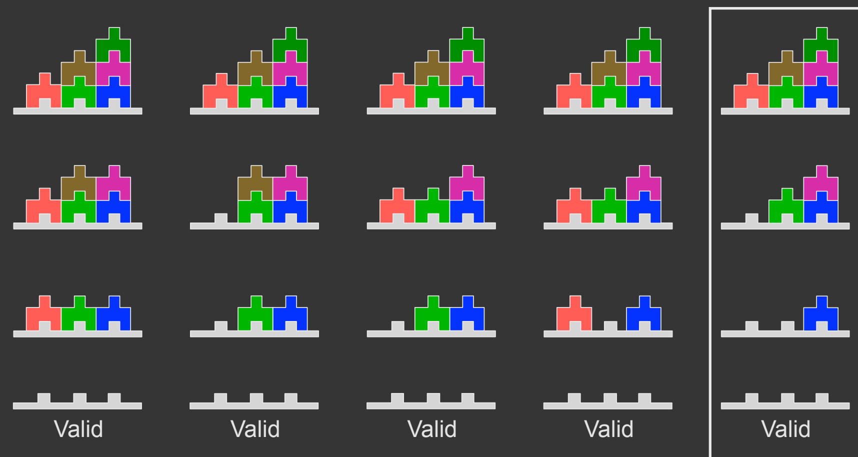
Geometric Analysis [Romney 95]



Geometric Assembly Planning



Many Geometrically Valid Sequences

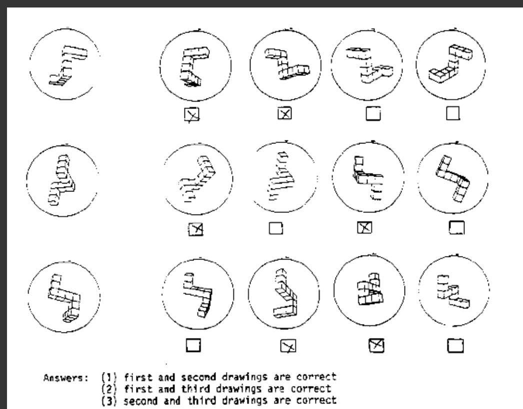


How do we choose the best sequence?

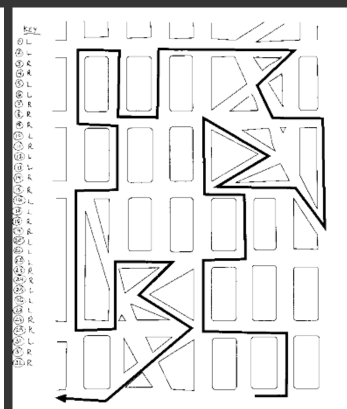
Identifying Design Principles

Stage 1: Production
Stage 2: Preference
Stage 3: Comprehension

Spatial Ability Tests



Mental Rotation [Vandenburg 78]



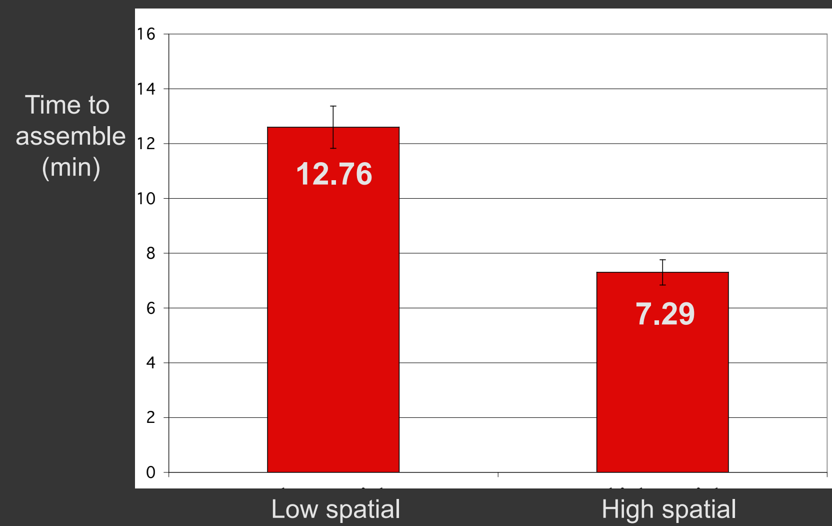
Navigation [Money 78]

Separate high and low spatial ability

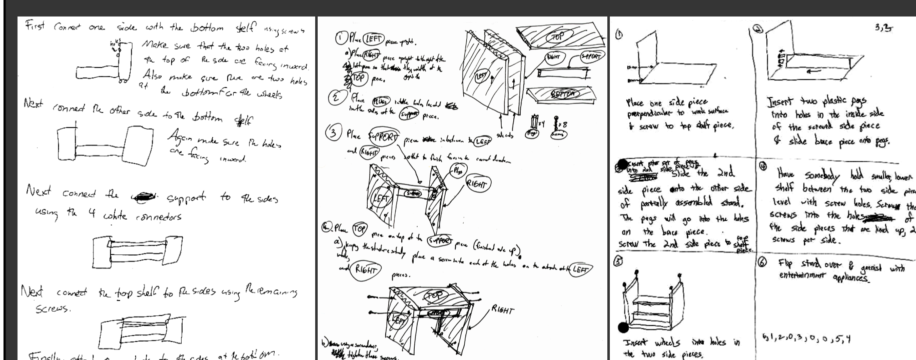
Stage 1: Production

- 43 Participants
- Assemble TV Stand without instructions
- Write instructions for novice assembler

Stage 1: Mean completion time

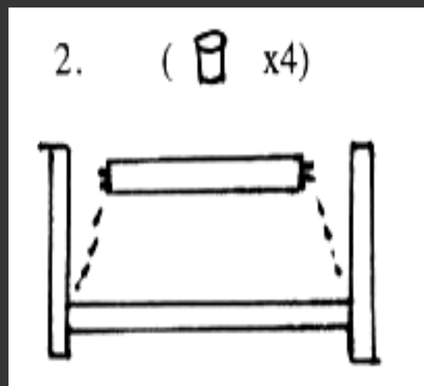


Stage 1: Instructions produced



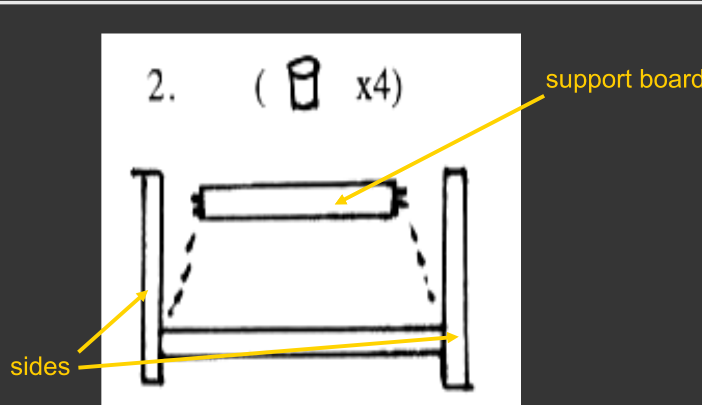
- Almost all contained diagrams 98%
- Text redundant with diagrams 62%

Stage 1: Errors in instructions



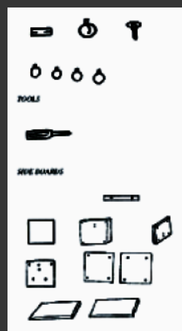
- Errors in low spatial instructions 86%
- Errors in high spatial instructions 12%

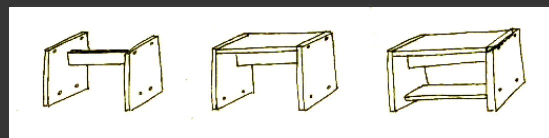
Stage 1: Errors in instructions

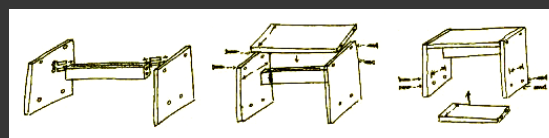


- Errors in low spatial instructions 86%
- Errors in high spatial instructions 12%

Stage 1: Classes of Diagrams


Parts menu

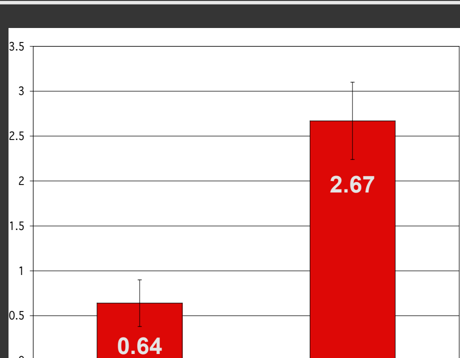

Structural diagrams


Action diagrams

- Parts menu to differentiate parts
- Structural diagrams depict completed step
- Action diagrams show assembly action/operation

Stage 1: Action diagrams

Mean number per set



Low spatial High spatial

- High spatial
  - More action diagrams
  - More 3D diagrams
  - Less text