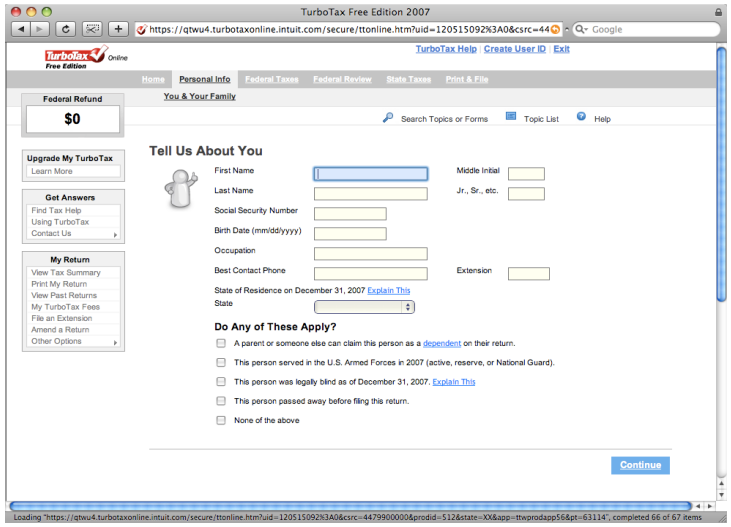
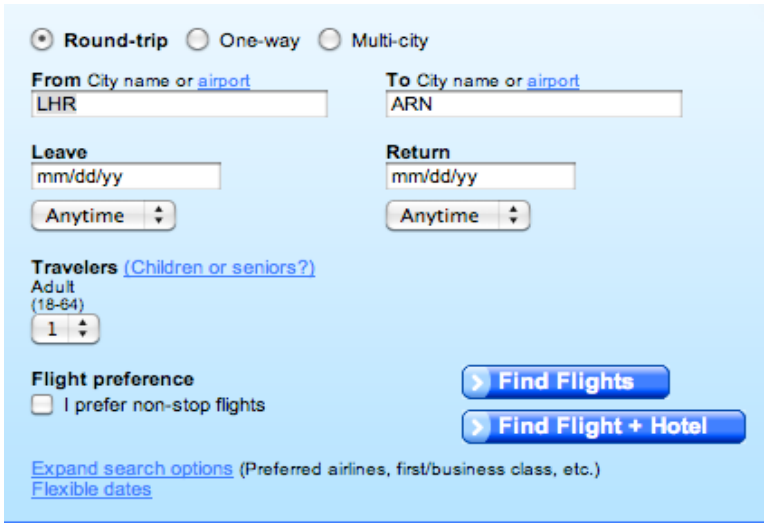
Logic Programming
Worksheets

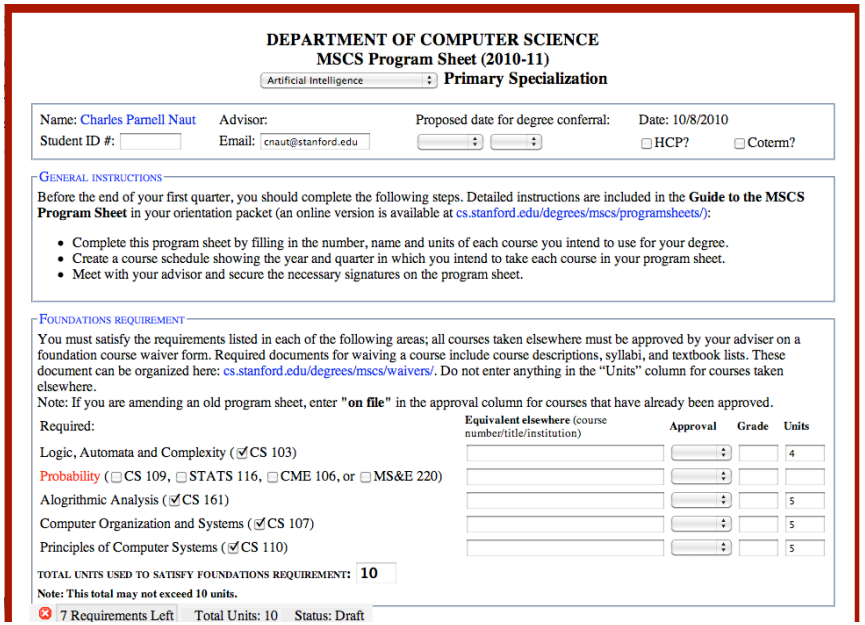
Michael Genesereth
Computer Science Department
Stanford University

Applications

Worksheets







DEPARTMENT OF COMPUTER SCIENCE
MSCS Program Sheet (2010-11)
Artificial Intelligence **Primary Specialization**

Name: Charles Parnell Naut Advisor: Proposed date for degree conferral: Date: 10/8/2010
Student ID #: Email: cnaut@stanford.edu HCP? Coterm?

GENERAL INSTRUCTIONS

Before the end of your first quarter, you should complete the following steps. Detailed instructions are included in the **Guide to the MSCS Program Sheet** in your orientation packet (an online version is available at cs.stanford.edu/degrees/mscs/programsheets/):

- Complete this program sheet by filling in the number, name and units of each course you intend to use for your degree.
- Create a course schedule showing the year and quarter in which you intend to take each course in your program sheet.
- Meet with your advisor and secure the necessary signatures on the program sheet.

FOUNDATIONS REQUIREMENT

You must satisfy the requirements listed in each of the following areas; all courses taken elsewhere must be approved by your adviser on a foundation course waiver form. Required documents for waiving a course include course descriptions, syllabi, and textbook lists. These document can be organized here: cs.stanford.edu/degrees/mscs/waivers/. Do not enter anything in the "Units" column for courses taken elsewhere.
Note: If you are amending an old program sheet, enter **"on file"** in the approval column for courses that have already been approved.

Required:	Equivalent elsewhere (course number/title/institution)	Approval	Grade	Units
Logic, Automata and Complexity (☑CS 103)				4
Probability (☐CS 109, ☐STATS 116, ☐CME 106, or ☐MS&E 220)				
Alogrithmic Analysis (☑CS 161)				5
Computer Organization and Systems (☑CS 107)				5
Principles of Computer Systems (☑CS 110)				5

TOTAL UNITS USED TO SATISFY FOUNDATIONS REQUIREMENT: 10
Note: This total may not exceed 10 units.

7 Requirements Left Total Units: 10 Status: Draft

Characteristics

Meaningful Data Display

- All data readily accessible
- Tables, Charts, Graphs

Modifiability

- What-you-see-is-what-you-get
- Random access - data can be changed in any order

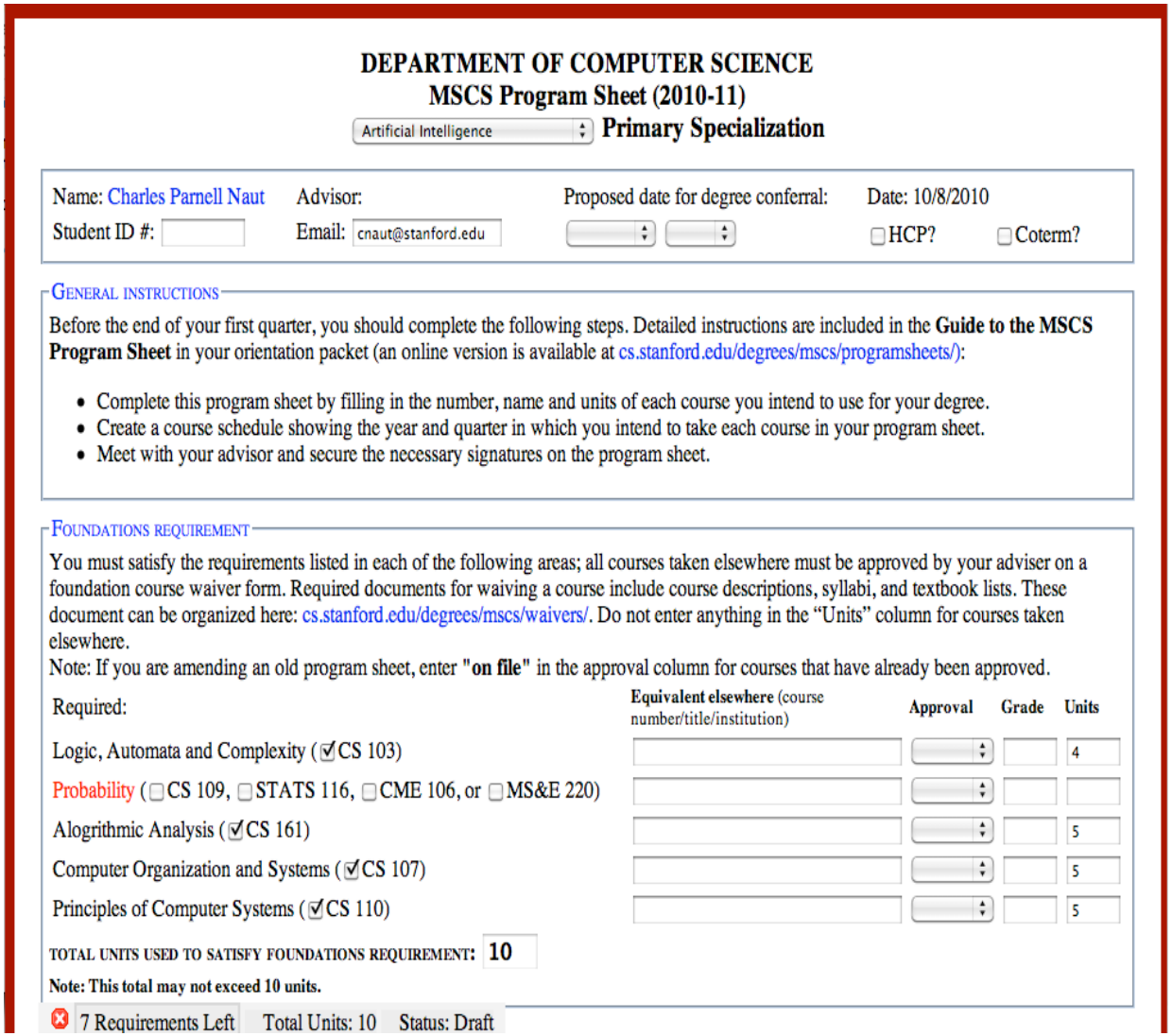
Constraint Checking

- Completeness and Consistency
- Problem alerting and Guidance in solving

Automatic Computation of Results

- Consequences computed
- Presentation automatically updated

Assignment - Academic Program Sheet



DEPARTMENT OF COMPUTER SCIENCE
MSCS Program Sheet (2010-11)
Artificial Intelligence **Primary Specialization**

Name: Charles Parnell Naut Advisor: Proposed date for degree conferral: Date: 10/8/2010
Student ID #: Email: cnaut@stanford.edu ☐HCP? ☐Coterm?

GENERAL INSTRUCTIONS

Before the end of your first quarter, you should complete the following steps. Detailed instructions are included in the **Guide to the MSCS Program Sheet** in your orientation packet (an online version is available at cs.stanford.edu/degrees/mscs/programsheets/):

- Complete this program sheet by filling in the number, name and units of each course you intend to use for your degree.
- Create a course schedule showing the year and quarter in which you intend to take each course in your program sheet.
- Meet with your advisor and secure the necessary signatures on the program sheet.

FOUNDATIONS REQUIREMENT

You must satisfy the requirements listed in each of the following areas; all courses taken elsewhere must be approved by your adviser on a foundation course waiver form. Required documents for waiving a course include course descriptions, syllabi, and textbook lists. These document can be organized here: cs.stanford.edu/degrees/mscs/waivers/. Do not enter anything in the "Units" column for courses taken elsewhere.
Note: If you are amending an old program sheet, enter **"on file"** in the approval column for courses that have already been approved.

Required:	**Equivalent elsewhere** (course number/title/institution)	**Approval**	**Grade**	**Units**
Logic, Automata and Complexity (☑CS 103)				4
Probability (☐CS 109, ☐STATS 116, ☐CME 106, or ☐MS&E 220)				
Alogrithmic Analysis (☑CS 161)				5
Computer Organization and Systems (☑CS 107)				5
Principles of Computer Systems (☑CS 110)				5

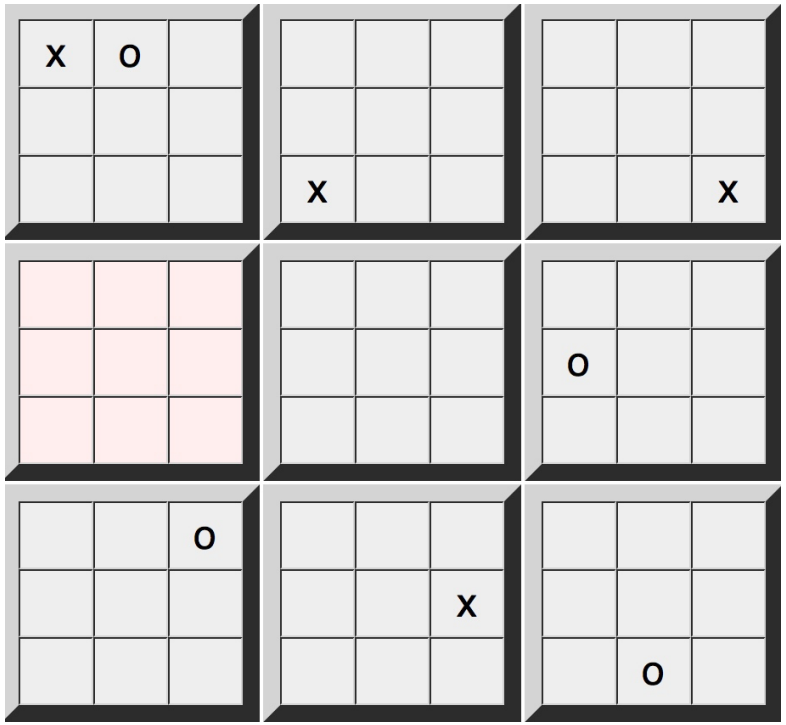
TOTAL UNITS USED TO SATISFY FOUNDATIONS REQUIREMENT: 10

Note: This total may not exceed 10 units.

7 Requirements Left Total Units: 10 Status: Draft

Demonstration

Assignment - Nineboard Tic Tac Toe

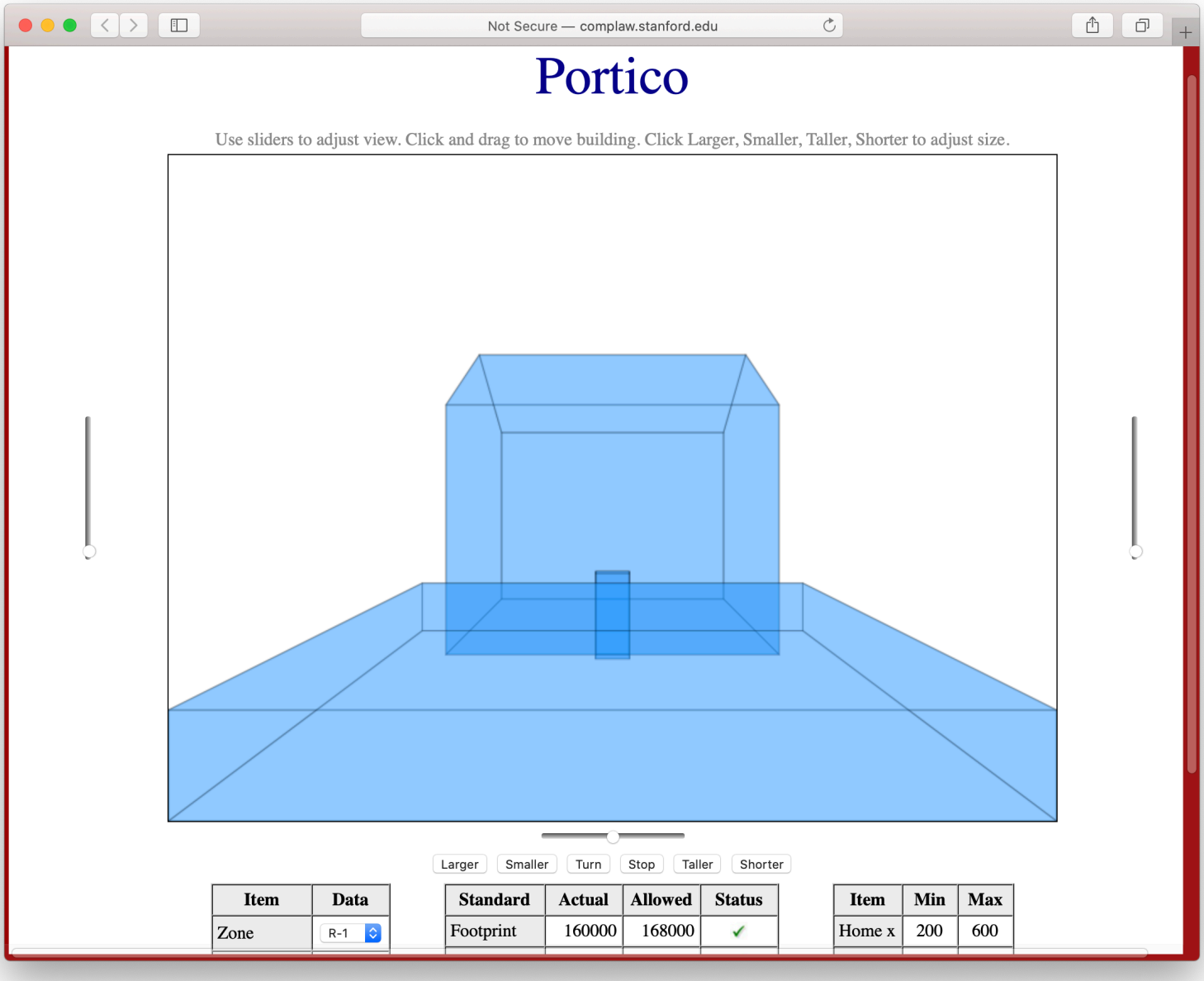


Demonstration

Example - Solar System

Demonstration

Assignment - Portico

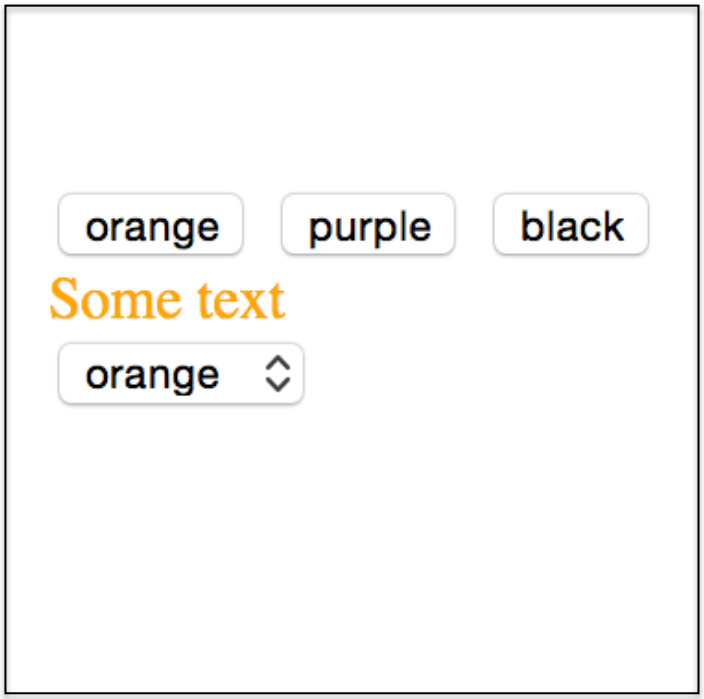


Demonstration

Javascript Worksheets

"Plain" HTML

Web browsers read HTML, create internal representation called the Document Object Model (DOM), and render page.




```
<html>
 <body>
    <input type='button' value='orange'/>
    <input type='button' value='purple'/>
    <input type='button' value='black'/>
    <p>Some text.</p>
    <select>
      <option>orange</option>
      <option>purple</option>
      <option>black</option>
    </select>
 </body>
</html>
```

"Mirror Semantics": Web browsers keeps the display in sync with the DOM. As DOM changes, page changes.

Reading the DOM with Javascrcipt

DOM:

```
<p id='text' style="color:orange">Some text.</p>
<select id='selector' onchange='handle(this)'>
  <option>black</option>
  <option selected>orange</option>
  <option>purple</option>
</select>
```

Javascript:

```
text = document.getElementById('text')
text.id has value "text"
text.style.color has value "orange"
text.innerHTML has value "Some text."

selector = document.getElementById('selector')
selector.id has value "page"
selector.value has value "orange"
```

Modifying the DOM with Javascript

DOM before:

```
<p id='text' style="color:orange">Some text.</p>
<select id='selector'>
  <option>black</option>
  <option selected>orange</option>
  <option>purple</option>
</select>
```

Dataset:

```
text.style.color = "purple"
text.innerHTML = "More text."
selector.value = "purple"
```

DOM after:

```
<p id='text' style="color:purple">More text.</p>
<select id='selector'>
  <option>black</option>
  <option>orange</option>
  <option selected>purple</option>
</select>
```

User Gestures

DOM:

```
<input type='button' value='orange' onclick='setcolor(this.value)'/>
```

user clicks

Action:

```
setcolor('orange')
```

Our Event Handler:

```
function setcolor (color)
 {document.getElementById('text').style.color = color;
  document.getElementById('selector').value = color}
```

Example

DOM:

```
<select id='selector' onchange='setcolor(this.value)'>
  <option>black</option>
  <option>orange</option> -> user selects
  <option>purple</option>
</select>
```

Action:

```
setcolor('orange')
```

Our Event Handler:

```
function setcolor (color)
 {document.getElementById('text').style.color = color;
  document.getElementById('selector').value = color}
```

Logical Worksheets

Our "Mirror" Semantics

Same as Javascript worksheets except that
- state is stored as a dataset
- values computed by applying rules to dataset

Dynamics
- User gestures translated to Javascript actions
- Actions change the dataset
- Changes to dataset reflected in DOM
- Changes to DOM are reflected in visible web page

Data and Rules

Lambda:

```
mood(gloomy)
```

Library:

```
setcolor(black) :: mood(M) ==> ~mood(M) & mood(gloomy)
setcolor(orange) :: mood(M) ==> ~mood(M) & mood(bright)
setcolor(purple) :: mood(M) ==> ~mood(M) & mood(despondent)

color(black) :- mood(gloomy)
color(orange) :- mood(bright)
color(purple) :- mood(despondent)
```

Javascript

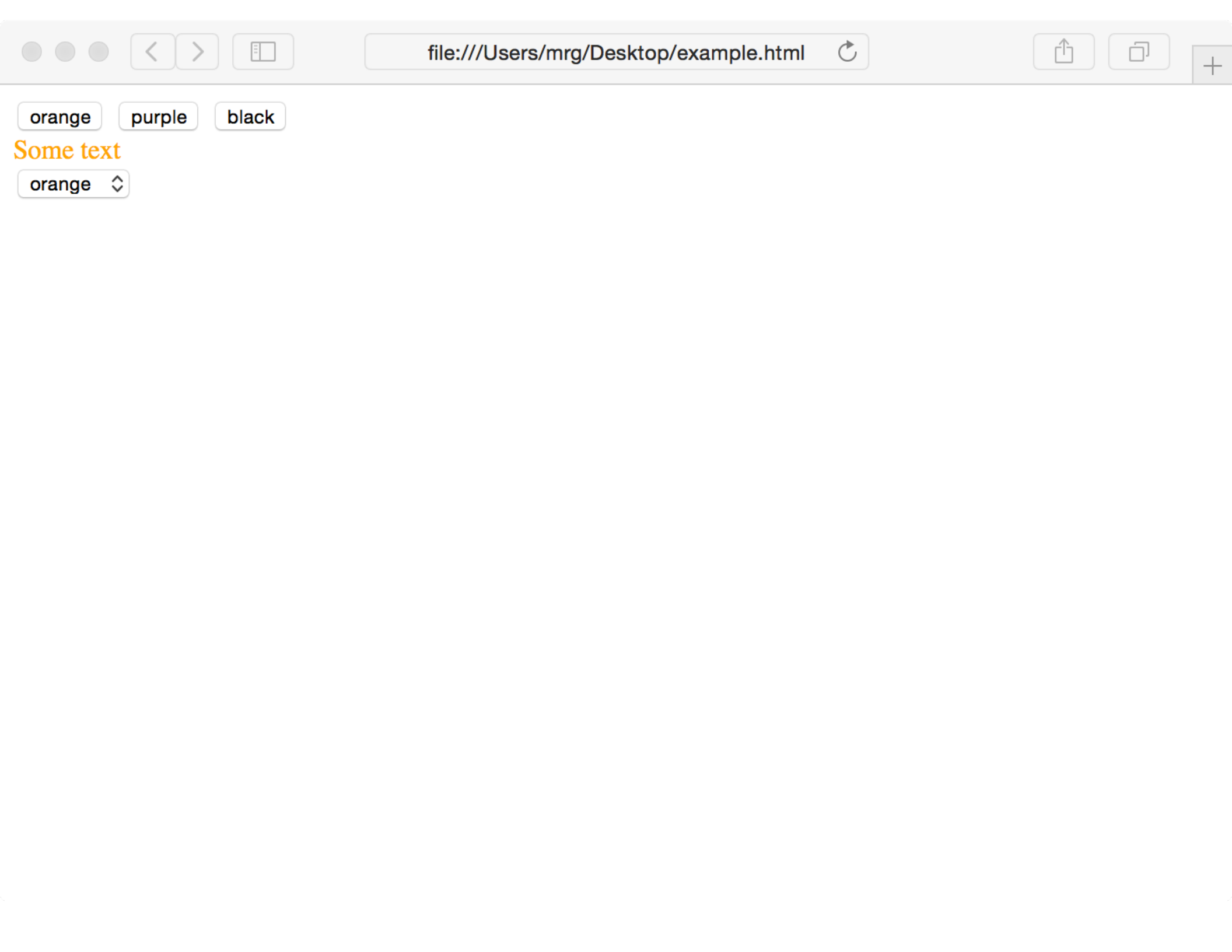
```
var lambda = [];
var library = [];

function initialize ()
 {var facts = readdata(document.getElementById('lambda').value);
  definefacts(lambda,facts);
  var rules = readdata(document.getElementById('library').value);
  definerules(library,rules)}

function setcolor (color)
 {compexecute(['setcolor',color],lambda,library);
  populate()}

function populate ()
 {var color = compfindx('C',read('color(C)'),lambda,library);
  document.getElementById('text').style.color = color;
  document.getElementById('selector').value = color}
```

Putting It All Together


file:///Users/mrg/Desktop/example.html
orange
purple
black
Some text
orange

Raw HTML

```
<html>
  <body>
    <input type='button' value='black'/>
    <input type='button' value='orange'/>
    <input type='button' value='purple'/>
    <p style="color:black">Some text.</p>
    <select>
      <option>black</option>
      <option>orange</option>
      <option>purple</option>
   </select>
  </body>
</html>
```

Add Identifiers

```
<html>
  <body>
    <input type='button' value='black'/>
    <input type='button' value='orange'/>
    <input type='button' value='purple'/>
    <p id='text' style="color:black">Some text.</p>
    <select>
      <option>black</option>
      <option>orange</option>
      <option>purple</option>
   </select>
  </body>
</html>
```

Add Event Handlers

```
<html>
  <body onload='initialize()'>
    <input type='button' value='black' onclick='setcolor(this.value)'/>
    <input type='button' value='orange' onclick='setcolor(this.value)'/>
    <input type='button' value='purple' onclick='setcolor(this.value)'/>
    <p id='text' style="color:black">Some text.</p>
    <select onchange='setcolor(this.value)'>
      <option>black</option>
      <option>orange</option>
      <option>purple</option>
   </select>
  </body>
</html>
```

Load Epilog

```
<html>
  <head>
    <script src='http://epilog.stanford.edu/javascript/epilog.js'></script>
  </head>
  <body onload='initialize()'>
    <input type='button' value='black' onclick='setcolor(this.value)'/>
    <input type='button' value='orange' onclick='setcolor(this.value)'/>
    <input type='button' value='purple' onclick='setcolor(this.value)'/>
    <p id='text' style="color:black">Some text.</p>
    <select onchange='setcolor(this.value)'>
      <option>black</option>
      <option>orange</option>
      <option>purple</option>
   </select>
  </body>
</html>
```

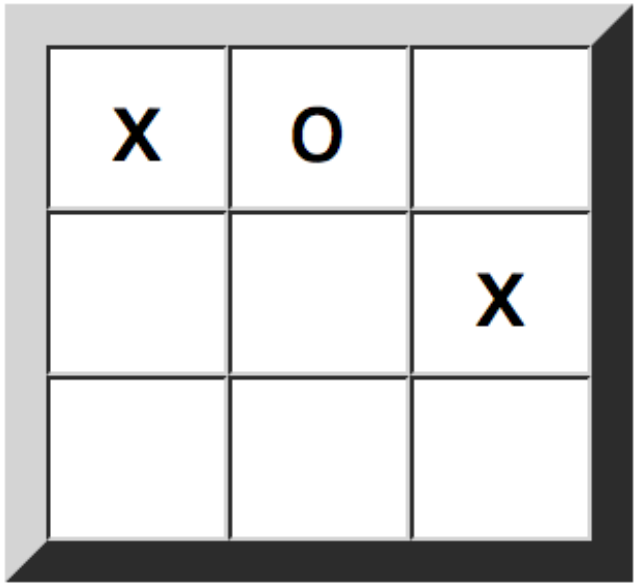
Define Handlers

```
<html>
  <head>
    <script src='http://epilog.stanford.edu/javascript/epilog.js'></script>
    <script type='text/javascript'>
      function initialize () {...}
      function setcolor (color) {...}
      function populate () {...}
    </script>
  </head>
  <body onload='initialize()'>
    <input type='button' value='black' onclick='setcolor(this.value)'/>
    <input type='button' value='orange' onclick='setcolor(this.value)'/>
    <input type='button' value='purple' onclick='setcolor(this.value)'/>
    <p id='text' style="color:black">Some text.</p>
    <select onchange='setcolor(this.value)'>
      <option>black</option>
      <option>orange</option>
      <option>purple</option>
   </select>
  </body>
</html>
```

Add Data and Rules

```
<html>
  <head>
    <script src='http://epilog.stanford.edu/javascript/epilog.js'></script>
    <script type='text/javascript'>
      function initialize () {...}
      function setcolor (color) {...}
      function populate () {...}
    </script>
  </head>
  <body onload='initialize()'>
    <input type='button' value='black' onclick='setcolor(this.value)'/>
    <input type='button' value='orange' onclick='setcolor(this.value)'/>
    <input type='button' value='purple' onclick='setcolor(this.value)'/>
    <p id='text' style="color:black">Some text.</p>
    <select onchange='setcolor(this.value)'>
      <option>black</option>
      <option>orange</option>
      <option>purple</option>
    </select>
  </body>
  <textarea id='lambda' style='display:none'>...</textarea>
  <textarea id='library' style='display:none'>...</textarea>
</html>
```

Tic Tac Toe



Tic Tac Toe Augmented HTML

```
<html>

  <head>
    <title>Tic Tac Toe</title>
    <script type='text/javascript' src='/epilog/javascript/epilog.js'></script>
    <script type='text/javascript'>
      function initialize() {definefacts(...); definerules(...)}
      function execute (id) {compexecute(read(id)); populate()}
      function populate() {... document.getElementById(cell).innerHTML = compfindx(...); ...}
    </script>
  </head>

  <body onload='initialize()'>
    <center>
      <table width='640' cellpadding='0' cellspacing='0' border='0'>
        <tr>
          <td width='640' height='90' align='center'style='font-size:48px;color:#000088;cursor:pointer'>
            Tic Tac Toe
          </td>
        </tr>
      </table>
      <table bgcolor='#ffffff' cellspacing='0' border='10'>
        <tr>
           <td id='mark(1,1)' width='60' height='60' align='center' valign='center' onclick='execute(this.id)'> </td>
            td id='mark(1,2)' width='60' height='60' align='center' valign='center' onclick='execute(this.id)'> </td>
            <td id='mark(1,3)' width='60' height='60' align='center' valign='center' onclick='execute(this.id)'> </td>
        </tr>
        <tr>
          <td id='mark(2,1)' width='60' height='60' align='center' valign='center' onclick='execute(this.id)'> </td>
          <td id='mark(2,2)' width='60' height='60' align='center' valign='center' onclick='execute(this.id)'> </td>
          <td id='mark(2,3)' width='60' height='60' align='center' valign='center' onclick='execute(this.id)'> </td>
        </tr>
        <tr>
          <td id='mark(3,1)' width='60' height='60' align='center' valign='center' onclick='execute(this.id)'> </td>
          <td id='mark(3,2)' width='60' height='60' align='center' valign='center' onclick='execute(this.id)'> </td>
          <td id='mark(3,3)' width='60' height='60' align='center' valign='center' onclick='execute(this.id)'> </td>
        </tr>
      </table>
    </center>
  </body>

  <textarea id='lambda' style='display:none'>...</textarea>
  <textarea id='library' style='display:none'>...</textarea>

</html>
```

Collaborative Worksheets


Google
Sheets

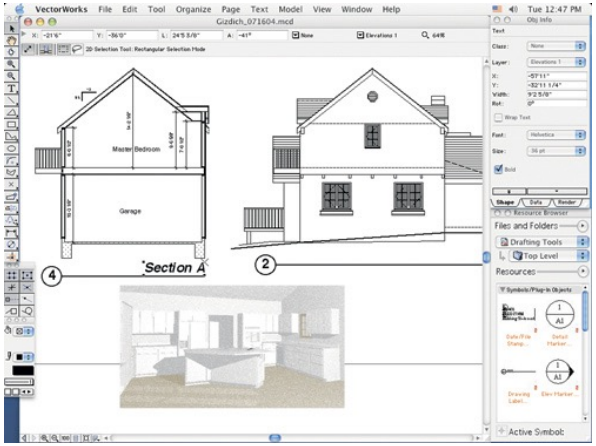
Skirmish

http://worksheets.stanford.edu/examples/skirmish/peertopeer.html?room=skirmish

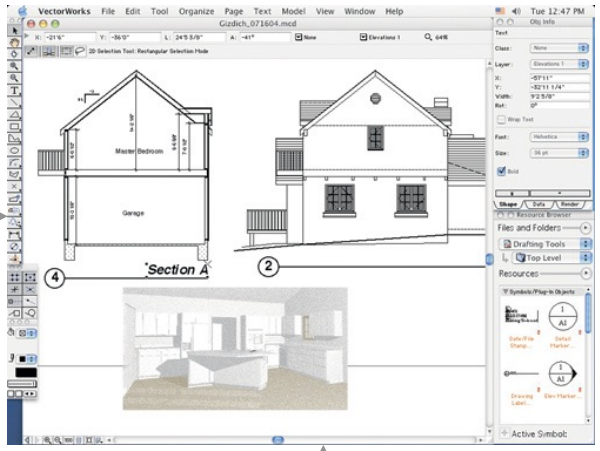
Collaborative Skirmish

http://worksheets.stanford.edu/examples/skirmish/peertopeer.html?room=skirmish

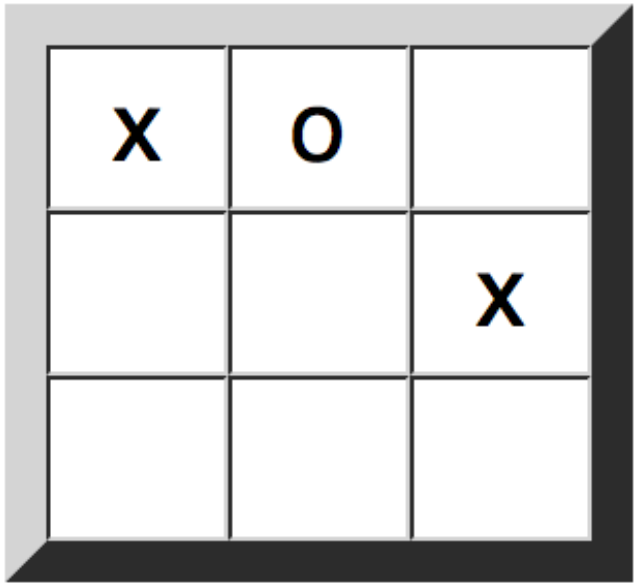
Heterogeneous Worksheets



Collaborative Heterogeneous Worksheets

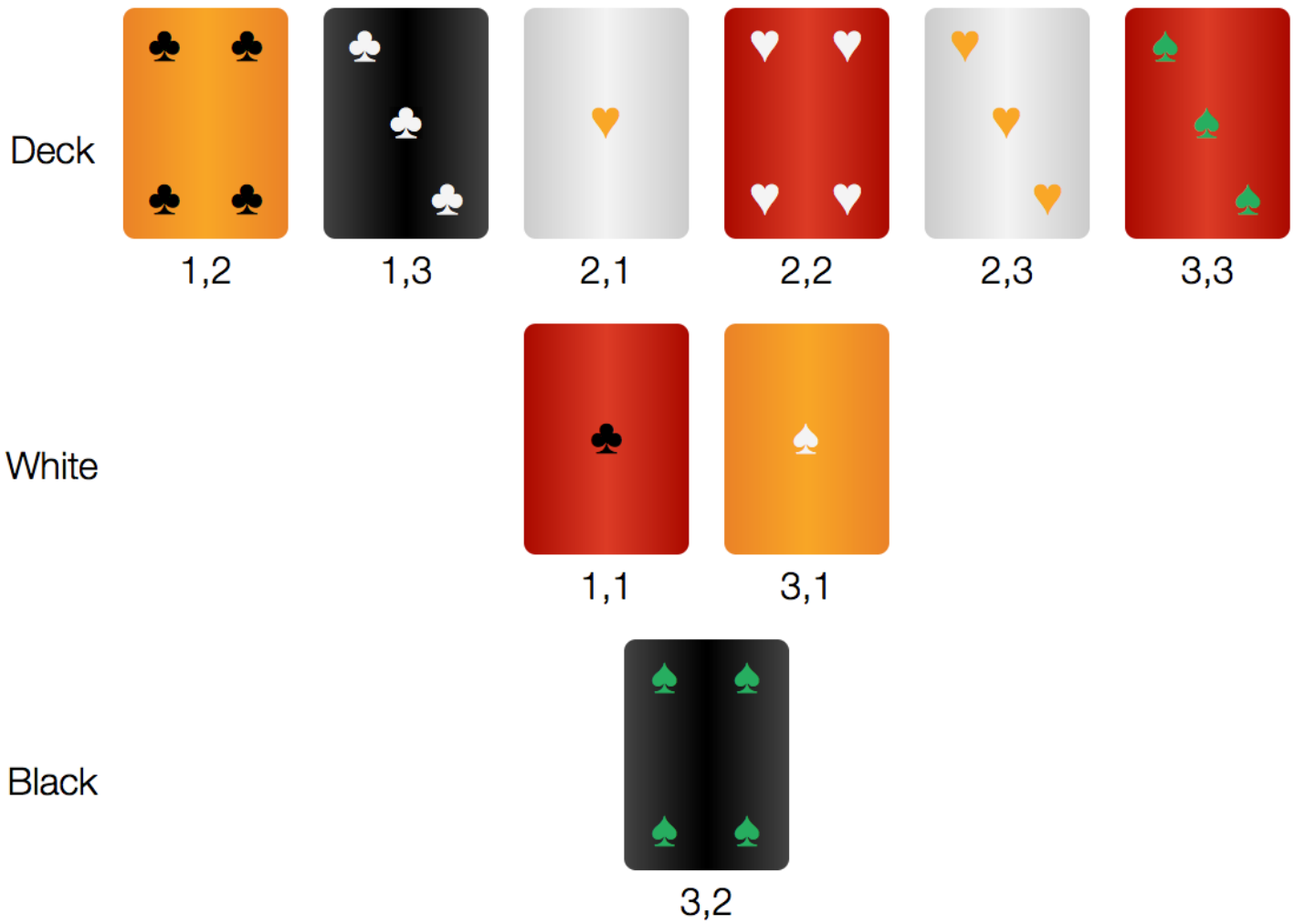


Tic Tac Toe



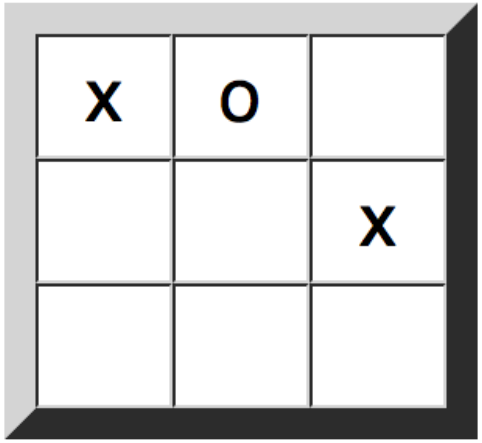
http://worksheets.stanford.edu/examples/tictactoe/peertopeer.html?room=cs151

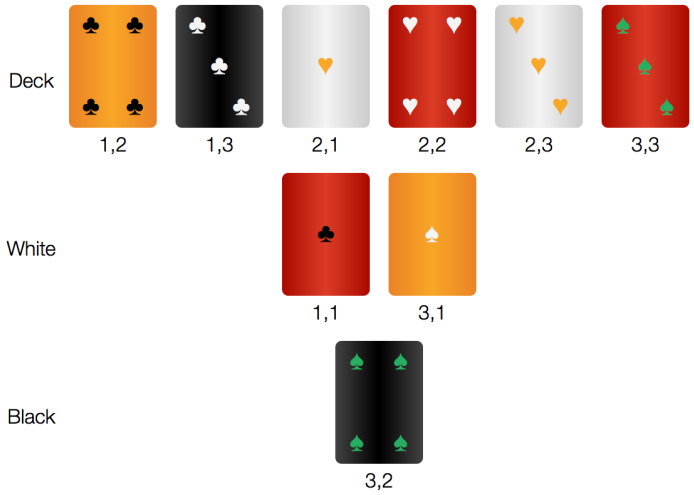
Trifecta



http://worksheets.stanford.edu/examples/trifecta/peertopeer.html?room=cs151

Tic Tac Toe - Trifecta





Remote Collaboration

Dataset Sharing

- Easy to implement and debug
- May move lots of data
- Allows all users to see and modify all data

Message Passing (Communication Channels)

- Difficult to implement and debug
- Moves minimal data
- Privacy and security assured

Backend Server (MySQL, PHP, etc.)

- Moderate effort to implement and debug
- Development and maintenance of backend infrastructure
- Moves minimal data
- Privacy and security assured