

Fraction Bars Project Code

GitHub Copilot

September 29, 2025

Contents

1	Main HTML File	2
1.1	Fraction_Bars.html	2
2	JavaScript Files	7
2.1	Core Fraction Bars Logic	7
2.1.1	fractionBars.js	7
2.1.2	FractionBarsCanvas.js	21
2.2	Object Classes	36
2.2.1	Bar.js	36
2.2.2	Mat.js	46
2.2.3	Point.js	47
2.2.4	Line.js	49
2.2.5	Split.js	49
2.2.6	Blob.js	50
2.3	UI Components	52
2.3.1	CanvasState.js	52
2.3.2	SplitsWidget.js	53
2.4	Utility Functions	54
2.4.1	utilities.js	54
2.4.2	cycle.js	56
3	CSS Files	58
3.1	Main Styles	58
3.1.1	fractionBars.css	58
3.1.2	deneme.css	60
3.2	Language and UI Styles	61
3.2.1	lang_eng.css	61
4	AceofBases Program	62
4.1	HTML Files	62
4.1.1	index.html	62
4.1.2	index_ace_of_bases.html	63
4.2	JavaScript Files	64
4.2.1	script.js	64
4.2.2	script_ace_of_bases.js	69
4.3	CSS Files	74
4.3.1	styles.css	74
4.3.2	styles_ace_of_bases.css	74
5	Configuration Files	75
5.1	Fraction_Bars.code-workspace	75

1 Main HTML File

1.1 Fraction_Bars.html

```
1 <!DOCTYPE html>
2 <!-- saved from url=(0033)https://educn101.siteshost.iu.edu/ -->
3 <html lang="en"><!-- test this will be a conflict --><head><meta http-equiv="Content-Type"
4   ↳ content="text/html; charset=UTF-8">
5
6   <title>HTML 5: Fraction Bars</title>
7   <link rel="stylesheet" href="./Fraction_Bars_files/fractionBars.css" type="text/css">
8   <link rel="stylesheet" href="./Fraction_Bars_files/lang_eng.css" type="text/css">
9   <link rel="stylesheet" href="./Fraction_Bars_files/deneme.css" type="text/css">
10  ↳ type="text/css">
11  <script src="./Fraction_Bars_files/jquery-1.9.1.min.js" type="text/javascript"
12  ↳ language="javascript"></script>
13  <script src="./Fraction_Bars_files/jquery-ui-1.10.3.custom.min.js" type="text/javascript"
14  ↳ language="javascript"></script>
15  <script src="https://cdnjs.cloudflare.com/ajax/libs/jqueryui-touch-punch/0.2.3/jquery.ui.touch-pu
16  ↳ nch.min.js"></script>
17  <script src="./Fraction_Bars_files/cycle.js" type="text/javascript"
18  ↳ language="javascript"></script>
19  <script src="./Fraction_Bars_files/FileSaver.min.js" type="text/javascript"
20  ↳ language="javascript"></script>
21  <script src="./Fraction_Bars_files/Blob.js" type="text/javascript" language="javascript"></script>
22  <script src="./Fraction_Bars_files/utilities.js" type="application/javascript"
23  ↳ language="javascript"></script>
24  <script src="./Fraction_Bars_files/Point.js" type="application/javascript"
25  ↳ language="javascript"></script>
26  <script src="./Fraction_Bars_files/Bar.js" type="application/javascript"
27  ↳ language="javascript"></script>
28  <script src="./Fraction_Bars_files/Mat.js" type="application/javascript"
29  ↳ language="javascript"></script>
30  <script src="./Fraction_Bars_files/Split.js" type="application/javascript"
31  ↳ language="javascript"></script>
32  <script src="./Fraction_Bars_files/SplitsWidget.js" type="application/javascript"
33  ↳ language="javascript"></script>
34  <script src="./Fraction_Bars_files/Line.js" type="application/javascript"
35  ↳ language="javascript"></script>
36  <script src="./Fraction_Bars_files/CanvasState.js" type="application/javascript"
37  ↳ language="javascript"></script>
38  <script src="./Fraction_Bars_files/FractionBarsCanvas.js" type="application/javascript"
39  ↳ language="javascript"></script>
40  <script src="./Fraction_Bars_files/fractionBars.js" type="application/javascript"
41  ↳ language="javascript"></script>
42
43 </head>
44 <body data-new-gr-c-s-check-loaded="14.1231.0" data-gr-ext-installed="">
45 <a class="skip-link" href="https://educn101.siteshost.iu.edu/#main">Skip to main content</a>
46 <h1 class="bar_titles" id="bar_titles">Fraction Bars</h1>
47 <div id="main">
48   <div id="tools">
49     <div class="toolGroup">
50       <a class="c_bar" id="tool_bar">&nbsp;</a>
51       <a class="c_mat" id="tool_mat">&nbsp;</a>
52       <!-- <a id="tool_cover">Cover</a> -->
53     </div>
54     <div class="toolGroup">
55       <a class="c_copy" id="action_copy">&nbsp;</a>
56       <a class="c_repeat" id="tool_repeat">&nbsp;</a>
57       <a class="c_iterate" id="window_iterate">&nbsp;</a>
58       <a class="c_join" id="action_join">&nbsp;</a>
59       <a class="c_delete" id="action_delete">&nbsp;</a>
60     </div>
61     <div class="toolGroup">
62       <a class="c_parts" id="window_split">&nbsp;</a>
63       <a class="c_pieces" id="tool_manualSplit">&nbsp;</a>
64       <a class="c_b_apart" id="action_breakApart">&nbsp;</a>
65       <a class="c_pullout" id="action_pullOutSplit">&nbsp;</a>
66       <a class="c_c_parts" id="action_clearSplits">&nbsp;</a>
67     </div>
68   </div>
69 </div>
```

```

53     <div class="toolGroup">
54         <a class="c_set_unit" id="action_setUnitBar">&nbsp;</a>
55         <a class="c_measure" id="action_measure">&nbsp;</a>
56         <a class="c_make" id="action_make">Make</a>
57         <a class="c_label" id="window_label">&nbsp;</a>
58     </div>
59     <div class="toolGroup">
60         <a class="c_undo" id="action_undo"></a>
61         <a class="c_redo" id="action_redo">&nbsp;</a>
62         <a class="c_save" id="action_save">&nbsp;</a>
63         <a class="c_open" id="action_open">&nbsp;</a>
64         <a class="c_new" id="action_clearAll">&nbsp;</a>
65         <a class="c_print" id="action_print">&nbsp;</a>
66         <a class="c_properties" id="window_properties">&nbsp;</a>
67     </div>
68     <div class="toolGroup" style="text-align:center">
69         <a class="colorBlock color10 colorSelected" id="setColor1">&nbsp;</a>
70         <a class="colorBlock color3" id="setColor2">&nbsp;</a>
71         <a class="colorBlock color7" id="setColor3">&nbsp;</a>
72         <a class="colorBlock color5" id="setColor4">&nbsp;</a>
73         <a class="colorBlock color12" id="setColor5">&nbsp;</a>
74         <a class="colorBlock color9" id="setColor6">&nbsp;</a>
75         <a class="colorBlock color13" id="setColor7">&nbsp;</a>
76         <a class="colorBlock color14" id="setColor8">&nbsp;</a>
77     </div>
78     <div class="toolGroup" style="text-align:center">&nbsp;<select class="c_filetext" id="id_filetext" multiple="multiple" style="display:
79     ↪ none;"></select>
80         <a class="c_previous" id="action_previous" style="display: none;">&nbsp;</a>
81         <a class="c_next" id="action_next" style="display: none;">&nbsp;</a>
82     </div>
83     <div class="toolGroup" style="text-align:center">
84         <a class="hideShow c_hide" id="tool_hide">&nbsp;</a>
85         <a class="hideShow c_show" id="action_show">&nbsp;</a>
86     </div>
87 </div>
88
89 <div hidden="" id="flags">
90     <a id="marked-iterate" data-flag="true">&nbsp;</a>
91 </div>
92
93 <canvas id="fbCanvas" width="700" height="600"></canvas>
94
95 <!--
96     <p style="clear:both">Mouse: (<span id="mouseAction"> </span>) <span id="mouseLoc">
97     ↪ </span></p>
98     -->
99
100 <input name="labelField" id="labelInput" type="text" value="">
101
102
103
104
105
106
107
108
109
110
111 </div>
112
113
114 <div class="ui-dialog ui-widget ui-widget-content ui-corner-all ui-front ui-dialog-buttons ui-draggable"
115 ↪ tabindex="-1" role="dialog" aria-describedby="dialog_splits" aria-labelledby="ui-id-1" style="display:
116 ↪ none;"><div class="ui-dialog-titlebar ui-widget-header ui-corner-all ui-helper-clearfix"><span
117 ↪ id="ui-id-1" class="ui-dialog-title"></span><button class="ui-button ui-widget ui-state-default
118 ↪ ui-corner-all ui-button-icon-only ui-dialog-titlebar-close" role="button" aria-disabled="false"
119 ↪ title="close"><span class="ui-button-icon-primary ui-icon ui-icon-closethick"></span><span
120 ↪ class="ui-button-text">close</span></button></div><div class="c_dialog_splits ui-dialog-content
121 ↪ ui-widget-content" id="dialog_splits">

```

```

119     <div class="radio_vert_horz" id="radio_vert" style="DISPLAY: none">
120         <label class="c_vertical" for="vert">&nbsp;</label>
121         <input type="radio" name="vert_horiz" id="vert" value="Vertical" checked="">
122         <label class="c_horizontal" for="horiz">&nbsp;</label>
123         <input type="radio" name="vert_horiz" id="horiz" value="Horizontal">
124     </div>
125
126     <p>
127         <label class="c_number_part" for="split-slider-field">&nbsp;</label>
128         <input type="text" id="split-slider-field" style="border:0; color:#f6931f;
129             ↪ font-weight:bold;" value="2" readonly="">
130     </p>
131
132     <div id="split-slider" class="ui-slider ui-slider-horizontal ui-widget ui-widget-content
133     ↪ ui-corner-all" aria-disabled="false"><a class="ui-slider-handle ui-state-default
134     ↪ ui-corner-all" href="https://educn101.sitehost.iu.edu/#" style="left: 0%;"></a></div>
135
136     <div id="radio_whole">
137         <label class="c_part_whole" for="whole">&nbsp;</label>
138         <input type="radio" name="whole_part" id="whole" value="Whole" checked="">
139         <br>
140         <label class="c_part_part" for="part">&nbsp;</label>
141         <input type="radio" name="whole_part" id="part" value="Part">
142     </div><div class="ui-dialog-buttonpane ui-widget-content ui-helper-clearfix"><div
143     ↪ class="ui-dialog-buttonset"><button type="button" class="ui-button ui-widget ui-state-default
144     ↪ ui-corner-all ui-button-text-only" role="button" aria-disabled="false"><span
145     ↪ class="ui-button-text">Ok</span></button><button type="button" class="ui-button ui-widget
146     ↪ ui-state-default ui-corner-all ui-button-text-only" role="button" aria-disabled="false"><span
147     ↪ class="ui-button-text">Cancel</span></button></div></div><div class="ui-dialog ui-widget
148     ↪ ui-widget-content ui-corner-all ui-front ui-dialog-buttons ui-draggable" tabindex="-1"
149     ↪ role="dialog" aria-describedby="dialog-properties" aria-labelledby="ui-id-2" style="display:
150     ↪ none;"><div class="ui-dialog-titlebar ui-widget-header ui-corner-all ui-helper-clearfix"><span
151     ↪ id="ui-id-2" class="ui-dialog-title"></span><button class="ui-button ui-widget
152     ↪ ui-state-default ui-corner-all ui-button-icon-only ui-dialog-titlebar-close" role="button"
153     ↪ aria-disabled="false" title="close"><span class="ui-button-icon-primary ui-icon
154     ↪ ui-icon-closethick"></span><span class="ui-button-text">close</span></button></div><div
155     ↪ class="c_dialog_properties ui-dialog-content ui-widget-content" id="dialog-properties">
156
157         <label class="c_iterations" for="same">&nbsp;</label>
158         <div id="radio_iterate_p" title="Iterations">
159             <p>
160                 <label class="c_dont_create" for="same">&nbsp;</label>
161                 <input type="radio" name="create" id="same" value="Same">
162                 <br>
163                 <label class="c_create_new" for="new">&nbsp;</label>
164                 <input type="radio" name="create" id="new" value="New" checked="">
165                 <br>
166                 ---
167                 <br>
168
169                 <label class="c_two_way" for="two_way">&nbsp;</label>
170                 <input type="radio" name="two_ittr" id="two_way" value="Two_way">
171                 <br>
172                 <label class="c_one_way" for="one_way">&nbsp;</label>
173                 <input type="radio" name="two_ittr" id="one_way" value="One_way"
174                 ↪ checked="">
175             </p>
176         </div>
177         <br>
178         <label class="c_splits" for="same">&nbsp;</label>
179         <div id="radio_split_p" title="Split or Manual Split">
180             <p>
181                 <label class="c_vert_horiz" for="two_horiz">&nbsp;</label>
182                 <input type="radio" name="two_split" id="two_horiz"
183                 ↪ value="Two_horiz">
184                 <br>
185                 <label class="c_only_vert" for="one_horiz">&nbsp;</label>
186                 <input type="radio" name="two_split" id="one_horiz"
187                 ↪ value="One_horiz" checked="">
188             </p>
189         </div>
190     </div>

```

```

175         <label class="c_lang" for="lang">&nbsp;</label>
176     <div id="radio_lang_tr" title="Language">
177         <p>
178             <fieldset class="elist">
179
180                 <ul>
181                     <li><input type="radio" name="lang" id="lang_tr"
182                         ↪ value="lang_tur" /><label class="c_lang_tur"
183                         ↪ for="lang_tr"></label></li>
184                     <li><input type="radio" name="lang" id="lang_en"
185                         ↪ value="lang_eng" checked /><label class="c_lang_eng"
186                         ↪ for="lang_en"></label></li>
187                 </ul>
188             </fieldset>
189         </p>
190     </div>
191
192     -->
193     <br>
194
195     <div id="tools2">
196         <div class="toolGroup" style="text-align:center">
197             <p>
198                 <label class="c_color" for="color">&nbsp;</label>
199             </p>
200
201             <a class="colorBlock1 color1"
202                 ↪ id="setColor1_2">&nbsp;</a>
203             <a class="colorBlock1 color2"
204                 ↪ id="setColor2_2">&nbsp;</a>
205             <a class="colorBlock1 color3"
206                 ↪ id="setColor3_2">&nbsp;</a>
207             <a class="colorBlock1 color4"
208                 ↪ id="setColor4_2">&nbsp;</a>
209             <a class="colorBlock1 color5"
210                 ↪ id="setColor5_2">&nbsp;</a>
211             <a class="colorBlock1 color6"
212                 ↪ id="setColor6_2">&nbsp;</a>
213             <a class="colorBlock1 color7"
214                 ↪ id="setColor7_2">&nbsp;</a>
215             <a class="colorBlock1 color8"
216                 ↪ id="setColor8_2">&nbsp;</a>
217             <a class="colorBlock1 color9"
218                 ↪ id="setColor9_2">&nbsp;</a>
219             <a class="colorBlock1 color10 colorSelected"
220                 ↪ id="setColor10">&nbsp;</a>
221             <a class="colorBlock1 color11"
222                 ↪ id="setColor11">&nbsp;</a>
223             <a class="colorBlock1 color12"
224                 ↪ id="setColor12">&nbsp;</a>
225             <a class="colorBlock1 color13"
226                 ↪ id="setColor13">&nbsp;</a>
227             <a class="colorBlock1 color14"
228                 ↪ id="setColor14">&nbsp;</a>
229             <a class="colorBlock1 color15"
230                 ↪ id="setColor15">&nbsp;</a>
231             <a class="colorBlock1 color16"
232                 ↪ id="setColor16">&nbsp;</a>
233
234         </div>
235     </div>
236
237 </div><div class="ui-dialog-buttonpane ui-widget-content ui-helper-clearfix"><div
238 ↪ class="ui-dialog-buttonset"><button type="button" class="ui-button ui-widget ui-state-default
239 ↪ ui-corner-all ui-button-text-only" role="button" aria-disabled="false"><span
240 ↪ class="ui-button-text">Ok</span></button><button type="button" class="ui-button ui-widget
241 ↪ ui-state-default ui-corner-all ui-button-text-only" role="button" aria-disabled="false"><span
242 ↪ class="ui-button-text">Cancel</span></button></div></div></div><div class="ui-dialog ui-widget
243 ↪ ui-widget-content ui-corner-all ui-front ui-dialog-buttons ui-draggable" tabindex="-1"
244 ↪ role="dialog" aria-describedby="dialog-iterate" aria-labelledby="ui-id-3" style="display:
245 ↪ none;"><div class="ui-dialog-titlebar ui-widget-header ui-corner-all ui-helper-clearfix"><span
246 ↪ id="ui-id-3" class="ui-dialog-title"></span><button class="ui-button ui-widget
247 ↪ ui-state-default ui-corner-all ui-button-icon-only ui-dialog-titlebar-close" role="button"
248 ↪ aria-disabled="false" title="close"><span class="ui-button-icon-primary ui-icon
249 ↪ ui-icon-closethick"></span><span class="ui-button-text">close</span></button></div><div
250 ↪ id="dialog-iterate" class="c_dialog_iterate ui-dialog-content ui-widget-content">
251
252     <div class="radio_iterate_vert_horz" id="iterate_vert-horiz" style="DISPLAY: none">

```

```

216         <label class="c_vertical" for="iterate_vert">&nbsp;</label>
217         <input type="radio" name="vert_horiz" id="iterate_vert" value="Vertical"
        ↪ checked="">
218         <label class="c_horizontal" for="iterate_horiz">&nbsp;</label>
219         <input type="radio" name="vert_horiz" id="iterate_horiz" value="Horizontal">
220     </div>
221
222     <p>
223         <label class="c_number_iterations" for="iterate-field">&nbsp;</label>
224         <input type="text" id="iterate-field" style="border:1; color:#000000;
        ↪ font-weight:bold;" value="2">
225     </p>
226
227 </div><div class="ui-dialog-buttonpane ui-widget-content ui-helper-clearfix"><div
    ↪ class="ui-dialog-buttonset"><button type="button" class="ui-button ui-widget ui-state-default
    ↪ ui-corner-all ui-button-text-only" role="button" aria-disabled="false"><span
    ↪ class="ui-button-text">Ok</span></button><button type="button" class="ui-button ui-widget
    ↪ ui-state-default ui-corner-all ui-button-text-only" role="button" aria-disabled="false"><span
    ↪ class="ui-button-text">Cancel</span></button></div></div></div><div class="ui-dialog ui-widget
    ↪ ui-widget-content ui-corner-all ui-front ui-dialog-buttons ui-draggable" tabindex="-1"
    ↪ role="dialog" aria-describedby="dialog-make" aria-labelledby="ui-id-4" style="display:
    ↪ none;"><div class="ui-dialog-titlebar ui-widget-header ui-corner-all ui-helper-clearfix"><span
    ↪ id="ui-id-4" class="ui-dialog-title"></span><button class="ui-button ui-widget
    ↪ ui-state-default ui-corner-all ui-button-icon-only ui-dialog-titlebar-close" role="button"
    ↪ aria-disabled="false" title="close"><span class="ui-button-icon-primary ui-icon
    ↪ ui-icon-closethick"></span><span class="ui-button-text">close</span></button></div><div
    ↪ id="dialog-make" class="c_dialog_make ui-dialog-content ui-widget-content">
228
229     <p>
230         <label class="c_number_whole" for="whole-field">Write Fraction:</label>
231     </p><table style="width:10">
232
233 <tbody>
234     <tr>
235         <td rowspan="2"><input type="text" id="whole-field" style="border:1; color:#000000;
        ↪ font-weight:bold; text-align:right;" size="4" value=" "></td>
236         <td style="border-bottom:solid 1px"><input type="text" id="num-field" style="border:1;
        ↪ color:#000000; font-weight:bold; text-align:center;" size="4"></td>
237     </tr>
238     <tr>
239         <td><input type="text" id="denum-field" style="border:1; color:#000000; font-weight:bold;
        ↪ text-align:center;" size="4"></td>
240     </tr>
241 </tbody>
242 </table>
243
244     &nbsp;&nbsp;&nbsp;
245
246 <p></p>
247
248 </div><div class="ui-dialog-buttonpane ui-widget-content ui-helper-clearfix"><div
    ↪ class="ui-dialog-buttonset"><button type="button" class="ui-button ui-widget ui-state-default
    ↪ ui-corner-all ui-button-text-only" role="button" aria-disabled="false"><span
    ↪ class="ui-button-text">Ok</span></button><button type="button" class="ui-button ui-widget
    ↪ ui-state-default ui-corner-all ui-button-text-only" role="button" aria-disabled="false"><span
    ↪ class="ui-button-text">Cancel</span></button></div></div></div><div class="ui-dialog ui-widget
    ↪ ui-widget-content ui-corner-all ui-front ui-dialog-buttons ui-draggable ui-resizable"
    ↪ tabindex="-1" role="dialog" aria-describedby="dialog-file" aria-labelledby="ui-id-5"
    ↪ style="display: none; position: absolute;"><div class="ui-dialog-titlebar ui-widget-header
    ↪ ui-corner-all ui-helper-clearfix"><span id="ui-id-5" class="ui-dialog-title"></span><button
    ↪ class="ui-button ui-widget ui-state-default ui-corner-all ui-button-icon-only
    ↪ ui-dialog-titlebar-close" role="button" aria-disabled="false" title="close"><span
    ↪ class="ui-button-icon-primary ui-icon ui-icon-closethick"></span><span
    ↪ class="ui-button-text">close</span></button></div><div class="c_choose_file ui-dialog-content
    ↪ ui-widget-content" id="dialog-file">
249         <p class="c_open_file">&nbsp;</p>
250         <p></p>
251         <p></p>
252         <input type="file" id="files" name="files[]" multiple="">
253 <!--         <output id="list"></output> -->

```

```

254     </div><div class="ui-dialog-buttonpane ui-widget-content ui-helper-clearfix"><div
        ↪ class="ui-dialog-buttonset"><button type="button" class="ui-button ui-widget ui-state-default
        ↪ ui-corner-all ui-button-text-only" role="button" aria-disabled="false"><span
        ↪ class="ui-button-text">Cancel</span></button></div></div><div class="ui-resizable-handle
        ↪ ui-resizable-n" style="z-index: 90;"></div><div class="ui-resizable-handle ui-resizable-e"
        ↪ style="z-index: 90;"></div><div class="ui-resizable-handle ui-resizable-s" style="z-index:
        ↪ 90;"></div><div class="ui-resizable-handle ui-resizable-w" style="z-index: 90;"></div><div
        ↪ class="ui-resizable-handle ui-resizable-se ui-icon ui-icon-gripsmall-diagonal-se"
        ↪ style="z-index: 90;"></div><div class="ui-resizable-handle ui-resizable-sw" style="z-index:
        ↪ 90;"></div><div class="ui-resizable-handle ui-resizable-ne" style="z-index: 90;"></div><div
        ↪ class="ui-resizable-handle ui-resizable-nw" style="z-index:
        ↪ 90;"></div></div></body><grammarly-desktop-integration
        ↪ data-grammarly-shadow-root="true"><template shadowrootmode="open"><style>
255     div.grammarly-desktop-integration {
256         position: absolute;
257         width: 1px;
258         height: 1px;
259         padding: 0;
260         margin: -1px;
261         overflow: hidden;
262         clip: rect(0, 0, 0, 0);
263         white-space: nowrap;
264         border: 0;
265         -moz-user-select: none;
266         -webkit-user-select: none;
267         -ms-user-select: none;
268         user-select: none;
269     }
270
271     div.grammarly-desktop-integration:before {
272         content: attr(data-content);
273     }
274 </style><div aria-label="grammarly-integration" role="group" tabindex="-1"
        ↪ class="grammarly-desktop-integration" data-content="{&quot;mode&quot;:&quot;limited&quot;,&quot;i
        ↪ sActive&quot;:&quot;false,&quot;isUserDisabled&quot;:&quot;false,&quot;isAlwaysAvailableAssistantEnabled&quot;
        ↪ :&quot;false&quot;}"></div></template></grammarly-desktop-integration></html>

```

2 JavaScript Files

2.1 Core Fraction Bars Logic

2.1.1 fractionBars.js

```

1  // Copyright University of Massachusetts Dartmouth 2014
2  //
3  // Designed and built by James P. Burke and Jason Orrill
4  // Modified and developed by Hakan Sandir
5  //
6  // This Javascript version of Fraction Bars is based on
7  // the Transparent Media desktop version of Fraction Bars,
8  // which in turn was based on the original TIMA Bars software
9  // by John Olive and Leslie Steffe.
10 // We thank them for allowing us to update that product.
11
12
13
14
15 /*
16 // pull in our other files
17
18 // TODO: figure out if this is really a desirable thing to do. I like it in
19 // that this approach feels more like other languages, but there are issues
20 // with the classes not being available when I expect them to be.
21
22 include_js('class/Point.js', 'js/');
23 include_js('class/Bar.js', 'js/');
24 include_js('class/Mat.js', 'js/');
25 include_js('class/Split.js', 'js/');
26 include_js('class/Line.js', 'js/');
27 include_js('class/FractionBarsCanvas.js', 'js/');
28

```

```

29 */
30
31 var point1 = null ;
32 var point2 = null;
33 var fbContext = null ;
34 var splitWidgetContext = null;
35 var hiddenButtons = [];
36 var hiddenButtonsName=[];
37
38 var fracEvent = null;
39
40 splitWidgetObj = null;
41
42 $(document).ready(function() {
43 //first attempt
44     hideButton("id_filetext");
45     hideButton("action_previous");
46     hideButton("action_next");
47
48
49
50     fbContext = $('#fbCanvas')[0].getContext( '2d' ) ;
51     fbCanvasObj = new FractionBarsCanvas(fbContext);
52     splitWidgetContext = $('#split-display')[0].getContext('2d');
53     var splitWidgetObj = new SplitsWidget(splitWidgetContext);
54
55     // High-DPI/Retina support: scale canvas for crisp display
56     var dpr = 3; // 3x for Retina/HiDPI
57     var $canvas = $('#fbCanvas');
58     var cssWidth = $canvas.attr('width');
59     var cssHeight = $canvas.attr('height');
60     $canvas[0].width = cssWidth * dpr;
61     $canvas[0].height = cssHeight * dpr;
62     $canvas.css({ width: cssWidth + 'px', height: cssHeight + 'px' });
63     fbContext.setTransform(dpr, 0, 0, dpr, 0, 0);
64
65     $("#split-slider").slider({
66         change: function(event,ui) {
67             splitWidgetObj.handleSliderChange(event, ui);
68         }
69     });
70
71     $("#vert,#horiz").change(handleVertHorizChange);
72
73     function handleVertHorizChange(event) {
74         splitWidgetObj.handleVertHorizChange(event);
75     }
76
77
78
79
80     $( "#files" ).change(handleFileSelect);
81     FBFileReader = new FileReader();
82
83
84
85
86 //First attempt
87     $( "#id_filetext" ).change(handleListSelect);
88 //
89
90
91     $('#fbCanvas').dblclick(function() {
92         var fbImg = fbContext.getImageData(0,0,1000,600) ;
93         fbContext.clearRect(0,0,1000,600) ;
94         // fbContext.restore() ;
95         fbContext.putImageData(fbImg,0,0);
96     });
97
98     $('#fbCanvas').mousemove(function(e) {
99         fracEvent = e;
100         updateMouseLoc(e, $(this));
101         updateMouseAction('mousemove');
102
103         var p = Point.createFromMouseEvent(e, $(this)) ;

```



```

104
105     if (fbCanvasObj.currentAction == "manualSplit") {
106         fbCanvasObj.manualSplitPoint = p;
107         fbCanvasObj.refreshCanvas();
108     }
109
110     if(fbCanvasObj.mouseDownLoc !== null) {
111         fbCanvasObj.updateCanvas(p);
112     }
113
114     //         if (fbCanvasObj.currentAction == "manualSplit") {
115     //             fbCanvasObj.manualSplitXORDraw(p);
116     //         }
117
118     });
119
120     $('#fbCanvas').mousedown(function(e) {
121
122         fbCanvasObj.check_for_drag = true;
123         fbCanvasObj.cacheUndoState();
124
125         updateMouseLoc(e, $(this));
126         updateMouseAction('mousedown');
127         fbCanvasObj.mouseDownLoc = Point.createFromMouseEvent(e, $(this)) ;
128         var b = fbCanvasObj.barClickedOn() ;
129         var m = fbCanvasObj.matClickedOn() ;
130
131         if( (fbCanvasObj.currentAction == 'bar') || (fbCanvasObj.currentAction == "mat")) {
132             fbCanvasObj.saveCanvas() ;
133         } else if( fbCanvasObj.currentAction == 'repeat' ) {
134             fbCanvasObj.addUndoState();
135             b.repeat(fbCanvasObj.mouseDownLoc);
136             fbCanvasObj.refreshCanvas();
137         } else {
138             // The click is being used to update the selected bars
139             if( b !== null ) {
140                 if( $.inArray(b, fbCanvasObj.selectedBars) == -1) { // clicked on bar is
141                     ↪ not already selected
142                     if( !Utilities.shiftKeyDown ) {
143                         fbCanvasObj.clearSelection();
144                     }
145                     $.each( fbCanvasObj.selectedBars, function(index, bar) {
146                         bar.clearSplitSelection();
147                     });
148                     fbCanvasObj.barToFront(b);
149                     fbCanvasObj.selectedBars.push(b);
150                     b.isSelected = true;
151                     b.selectSplit(fbCanvasObj.mouseDownLoc);
152                 } else {
153                     ↪ // clicked bar is already selected
154                     $.each( fbCanvasObj.selectedBars, function(index, bar) {
155                         bar.clearSplitSelection();
156                     });
157                     if( !Utilities.shiftKeyDown ) {
158                         b.selectSplit(fbCanvasObj.mouseDownLoc);
159                     } else {
160                         fbCanvasObj.removeBarFromSelection(b);
161                     }
162                     fbCanvasObj.barToFront(b);
163                 }
164                 if (fbCanvasObj.currentAction == "manualSplit") {
165                     fbCanvasObj.clearSelection();
166                 }
167             } else if( m !== null ) {
168                 if( $.inArray(m, fbCanvasObj.selectedMats) == -1) { // clicked on mat is
169                     ↪ not already selected
170                     if( !Utilities.shiftKeyDown ) {
171                         fbCanvasObj.clearSelection();
172                     }
173                     m.isSelected = true;
174                     fbCanvasObj.selectedMats.push(m);
175                 } else { // Clicked on mat is already selected
176                     if( Utilities.shiftKeyDown ) {
177                         fbCanvasObj.removeMatFromSelection(m);
178                     }
179                 }
180             }
181         }
182     });

```

```

176         }
177     } else {
178         fbCanvasObj.clearSelection();
179     }
180     fbCanvasObj.refreshCanvas();
181 }
182 });
183
184 $('#fbCanvas').mouseup(function(e) {
185     updateMouseLoc(e, $(this));
186     updateMouseAction('mouseup');
187
188     fbCanvasObj.mouseUpLoc = Point.createFromMouseEvent(e, $(this)) ;
189
190
191     if( fbCanvasObj.currentAction == 'bar' ) {
192         fbCanvasObj.addUndoState();
193         fbCanvasObj.addBar() ;
194         fbCanvasObj.clear_selection_button () ;
195     } else if (fbCanvasObj.currentAction == 'mat') {
196         fbCanvasObj.addUndoState();
197         fbCanvasObj.addMat();
198         fbCanvasObj.clear_selection_button () ;
199     }
200 }
201
202
203 if (fbCanvasObj.found_a_drag){
204     fbCanvasObj.finalizeCachedUndoState();
205     fbCanvasObj.check_for_drag = false;
206 }
207
208 fbCanvasObj.mouseUpLoc = null ;
209 fbCanvasObj.mouseDownLoc = null ;
210 fbCanvasObj.mouseLastLoc = null ;
211
212 });
213
214 $('#colorBlock').click(function(e) {
215     fbCanvasObj.setFillColor( $(this).css('background-color'));
216     $('#colorBlock').removeClass('colorSelected');
217     $(this).addClass('colorSelected');
218     fbCanvasObj.updateColorsOfSelectedBars();
219     fbCanvasObj.refreshCanvas();
220 });
221
222 //first attempt
223 $('#colorBlock1').click(function(e) {
224     document.getElementById('fbCanvas').style.backgroundColor = $(this).css('background-color');
225     $('#colorBlock1').removeClass('colorSelected');
226     $(this).addClass('colorSelected');
227 });
228 //
229
230
231 $('#a').click(function(e) {
232
233     var thisId = $(this).attr('id') ;
234     if (thisId === null) { return; }
235     var tool_on = false; // just temporarily keeps track of whether we're turning a tool on or
236     ↪ off
237
238     // First, handle any hiding, if we're in that mode
239     if ((fbCanvasObj.currentAction == 'hide') && (thisId.indexOf('hide') == -1) ) {
240         $(this).hide();
241         hiddenButtonsName.push(thisId);
242         hiddenButtons.push($(this));
243         return;
244     }
245
246     if( thisId.indexOf('tool_') > -1 ) {
247
248         var toolName = thisId.substr(5,thisId.length);
249         if( toolName.toString() == fbCanvasObj.currentAction.toString() ) {
250             tool_on = false;

```

```

250         fbCanvasObj.clear_selection_button ();
251     } else {
252         fbCanvasObj.currentAction = thisId.substr(5,thisId.length) ;
253         tool_on = true;
254         $(this).addClass('toolSelected');
255     }
256     fbCanvasObj.handleToolUpdate(toolName, tool_on);
257     fbCanvasObj.refreshCanvas();
258 }
259
260 if( thisId.indexOf('action_') > -1 ) {
261     fbCanvasObj.name=thisId.substr( 7, thisId.length );
262     switch( thisId.substr( 7, thisId.length )) {
263         case 'copy':
264             fbCanvasObj.addUndoState();
265             fbCanvasObj.copyBars() ;
266             fbCanvasObj.refreshCanvas() ;
267             break ;
268         case 'delete':
269             fbCanvasObj.addUndoState();
270             fbCanvasObj.deleteSelectedBars() ;
271             fbCanvasObj.refreshCanvas() ;
272             break ;
273         case 'join':
274             fbCanvasObj.addUndoState();
275             fbCanvasObj.joinSelected() ;
276             fbCanvasObj.refreshCanvas() ;
277             break ;
278         case 'setUnitBar':
279             fbCanvasObj.addUndoState();
280             fbCanvasObj.setUnitBar() ;
281             fbCanvasObj.refreshCanvas() ;
282             break ;
283         case 'measure':
284             fbCanvasObj.addUndoState();
285             fbCanvasObj.measureBars() ;
286             fbCanvasObj.refreshCanvas() ;
287             break ;
288         case 'make':
289             fbCanvasObj.addUndoState();
290             fbCanvasObj.make() ;
291             fbCanvasObj.refreshCanvas() ;
292             break ;
293         case 'breakApart':
294             fbCanvasObj.addUndoState();
295             fbCanvasObj.breakApartBars() ;
296             fbCanvasObj.refreshCanvas() ;
297             break ;
298         case 'clearSplits':
299             fbCanvasObj.addUndoState();
300             fbCanvasObj.clearSplits() ;
301             fbCanvasObj.refreshCanvas();
302             break ;
303         case 'pullOutSplit':
304             fbCanvasObj.addUndoState();
305             fbCanvasObj.pullOutSplit();
306             fbCanvasObj.refreshCanvas();
307             break ;
308         case 'undo':
309             fbCanvasObj.undo();
310             fbCanvasObj.refreshCanvas() ;
311             break ;
312         case 'redo':
313             fbCanvasObj.redo();
314             fbCanvasObj.refreshCanvas();
315             break;
316         case 'save':
317             fbCanvasObj.save();
318             break;
319         case 'open':
320             SaveScreen();
321             resetFormElement($("#files"));
322             fbCanvasObj.openFileDialog();
323             break;
324         case 'print':

```

```

325         fbCanvasObj.print_canvas();
326         break ;
327     case 'clearAll':
328         SaveScreen();
329         location.reload();
330         break;
331     case 'show':
332         showAllButtons();
333         break;
334     case 'previous':
335         previousSelectFile();
336         break;
337         case 'next':
338             nextSelectFile();
339             break;
340
341     }
342
343 }
344
345 if( thisId.indexOf('window_') > -1 ) {
346     switch( thisId.substr( 7, thisId.length )) {
347         case 'label':
348             fbCanvasObj.addUndoState();
349             fbCanvasObj.editLabel() ;
350             break ;
351         case 'split':
352             fbCanvasObj.addUndoState();
353             fbCanvasObj.split(splitWidgetObj) ;
354             break ;
355         case 'iterate':
356             fbCanvasObj.addUndoState();
357             fbCanvasObj.iterate() ;
358             break ;
359         case 'properties':
360             fbCanvasObj.properties();
361             break ;
362     }
363 }
364
365 });
366
367
368 $(document).keydown(function(e) {
369
370     if( e.which == 16 ) {
371         Utilities.shiftKeyDown = true ;
372         fbCanvasObj.refreshCanvas();
373     }
374 });
375 $(document).keyup(function(e) {
376     if( e.which == 16 ) {
377         Utilities.shiftKeyDown = false ;
378         fbCanvasObj.refreshCanvas();
379     }
380
381     if( e.ctrlKey && e.keyCode==80 ) {
382         fbCanvasObj.properties();
383         fbCanvasObj.refreshCanvas();
384     }
385
386     if( e.ctrlKey && e.keyCode==83 ) {
387         fbCanvasObj.save();
388         fbCanvasObj.refreshCanvas();
389     }
390
391     if( e.ctrlKey && e.keyCode==72 ) {
392         //$( "#dialog-hidden" ).dialog('open');
393         if(Utilities.ctrlKeyDown){
394             showButton("tool_hide");
395             showButton("action_show");
396             Utilities.ctrlKeyDown=false;
397         } else {
398             Utilities.ctrlKeyDown=true;
399             hideButton("tool_hide");

```

```

400             hideButton("action_show");
401         }
402         fbCanvasObj.clear_selection_button();
403         fbCanvasObj.refreshCanvas();
404     }
405     if( e.ctrlKey && e.keyCode==46) {
406         fbCanvasObj.addUndoState();
407         fbCanvasObj.deleteSelectedBars() ;
408         fbCanvasObj.refreshCanvas() ;
409     }
410
411 });
412
413 $('#labelInput').keyup( function( e ) {
414     if( e.which == 13 ) {
415         fbCanvasObj.saveLabel( $('#labelInput').val(), Utilities.USE_CURRENT_SELECTION ) ;
416         fbCanvasObj.hideEditLabel() ;
417         fbCanvasObj.refreshCanvas();
418     }
419 });
420
421 // This gets triggered after we have already cleared out the selection,
422 // so we need to have a way to be sure the LAST selection gets the label.
423 $('#labelInput').blur( function() {
424     fbCanvasObj.saveLabel( $('#labelInput').val(), Utilities.USE_LAST_SELECTION ) ;
425     fbCanvasObj.hideEditLabel() ;
426 });
427
428
429
430 $( "#dialog-splits" ).dialog({
431     height: 300,
432     width: 400,
433     resizable: false,
434     modal: true,
435     buttons: [
436         {
437             text: "Ok",
438             click: function() {
439                 var num_splits = $("#split-slider-field").val();
440                 var whole = $("input[type='radio'][name='whole_part']:checked").val();
441                 var direction="Vertical";
442                 if(Utilities.flag[1])
443                 {
444                     direction = $("input[type='radio'][name='vert_ho
445                     ↪ riz']:checked").val();
446                 }
447                 fbCanvasObj.makeSplits(num_splits, direction, whole);
448                 $( this ).dialog( "close" );
449             }
450         },
451         {
452             text: "Cancel",
453             click: function() {
454                 $( this ).dialog( "close" );
455             }
456         }
457     ],
458     autoOpen: false
459 });
460
461 $( "#dialog-properties" ).dialog({
462     height: 500,
463     width: 400,
464     resizable: false,
465     modal: true,
466     buttons: [
467         {
468             text: "Ok",
469             click: function() {
470                 var create_checked =
471                 ↪ $("input[type='radio'][name='create']:checked").val();
472                 splitWidgetObj.vertical=true;

```

```

472         if (create_checked == "Same") {
473             Utilities.flag[0]= true;
474         } else if (create_checked == "New") {
475             Utilities.flag[0]= false;}
476
477         var horiz_checked = $("input[type='radio'] [name='two_spli
478         ↵ t']:checked").val();
479         if (horiz_checked == "One_horiz") {
480             Utilities.flag[1]= false;
481             document.getElementById("radio_vert").style.displ
482             ↵ ay = 'none';
483         } else if (horiz_checked == "Two_horiz") {
484             Utilities.flag[1]= true;
485             document.getElementById("radio_vert").style.displ
486             ↵ ay = 'block';
487         }
488
489         var itterate_way_checked = $("input[type='radio'] [name='t
490         ↵ wo_ittr']:checked").val();
491         if (itterate_way_checked == "One_way") {
492             Utilities.flag[2]= false;
493             document.getElementById("iterate_vert-horiz").sty
494             ↵ le.display = 'none';
495         } else if (itterate_way_checked == "Two_way") {
496             Utilities.flag[2]= true;
497             document.getElementById("iterate_vert-horiz").sty
498             ↵ le.display = 'block';
499         }
500
501         var language_checked =
502         ↵ $("input[type='radio'] [name='lang']:checked").val();
503         switch(language_checked) {
504             case 'lang_eng':
505                 Utilities.flag[3]= false;
506                 document.getElementById('stylesheet').href='css/1
507                 ↵ ang_eng.css';
508                 break ;
509             case 'lang_tur':
510                 Utilities.flag[3]= true;
511                 document.getElementById('stylesheet').href='css/1
512                 ↵ ang_tur.css';
513                 break ;
514         }
515
516         $( this ).dialog( "close" );
517     },
518     {
519         text: "Cancel",
520         click: function() {
521             $( this ).dialog( "close" );
522         }
523     },
524     ],
525     autoOpen: false
526 });
527
528 $( "#dialog-iterate" ).dialog({
529     height: 300,
530     width: 400,
531     resizable: false,
532     modal: true,
533     buttons: [
534         {
535             text: "Ok",
536             click: function() {
537                 var num_iterate = $("#iterate-field").val();
538                 if(!Utilities.flag[2])
539                 {
540                     direction="Horizontal";
541                 }
542                 else

```

```

537         {
538             var direction = $("input[type='radio'][name='ver']
                    ↪ t_horiz']:checked").val();
539         }
540         fbCanvasObj.makeIterations(num_iterate, direction);
541         $( this ).dialog( "close" );
542     }
543 },
544 {
545     text: "Cancel",
546     click: function() {
547         $( this ).dialog( "close" );
548     }
549 },
550 ],
551     autoOpen: false
552 });
553
554 $( "#dialog-make" ).dialog({
555     height: 300,
556     width: 400,
557     resizable: false,
558     modal: true,
559     buttons: [
560         {
561             text: "Ok",
562             click: function() {
563                 var num_whole = parseFloat($("#whole-field").val());
564                 var num_num = parseFloat($("#num-field").val());
565                 var num_denum = parseFloat($("#denum-field").val());
566
567                 if(!num_whole)
568                 {
569                     num_whole=0;
570                 }
571                 if(!num_denum)
572                 {
573                     num_denum=1;
574                 }
575                 if(!num_num)
576                 {
577                     num_num=0;
578                 }
579                 num_frac=num_whole + (num_num/num_denum);
580                 if (!num_frac)
581                 {
582                     alert("Please input fraction!");
583                 }
584                 else
585                 {
586                     fbCanvasObj.makeMake(num_frac);
587                 }
588
589                 document.getElementById('whole-field').value="";
590                 document.getElementById('num-field').value="";
591                 document.getElementById('denum-field').value="";
592                 $( this ).dialog( "close" );
593             }
594         },
595         {
596             text: "Cancel",
597             click: function() {
598                 $( this ).dialog( "close" );
599             }
600         }
601     ],
602     autoOpen: false
603 });
604
605 $( "#split-slider" ).slider({
606     value:2,
607     min: 2,
608     max: 20,
609     step: 1,
610     slide: function( event, ui ) {

```

```

611         $( "#split-slider-field" ).val( ui.value );
612     }
613 });
614
615 $( "#dialog-hidden" ).dialog({
616     height: 250,
617     width: 300,
618     modal: true,
619     buttons: [
620         {
621             text: "Ok",
622             click: function() {
623                 ///////////////
624
625                 $( this ).dialog( "close" );
626             }
627         },
628         {
629             text: "Cancel",
630             click: function() {
631                 $( this ).dialog( "close" );
632             }
633         }
634     ],
635     autoOpen: false
636 });
637
638 $( "#dialog-file" ).dialog({
639     height: 250,
640     width: 300,
641     modal: true,
642     buttons: [
643         {
644             text: "Cancel",
645             click: function() {
646                 $( this ).dialog( "close" );
647             }
648         }
649     ],
650     autoOpen: false
651 });
652
653 // --- Touch event helpers ---
654 function getTouchPos(e, elem) {
655     var touch = e.originalEvent.touches[0] || e.originalEvent.changedTouches[0];
656     return {
657         x: touch.clientX - elem.position().left,
658         y: touch.clientY - elem.position().top
659     };
660 }
661 function normalizeEvent(e, elem) {
662     if (e.type.startsWith('touch')) {
663         var pos = getTouchPos(e, elem);
664         return {
665             clientX: pos.x + elem.position().left,
666             clientY: pos.y + elem.position().top,
667             which: 1,
668             ctrlKey: e.ctrlKey || false,
669             shiftKey: e.shiftKey || false,
670             preventDefault: function() { e.preventDefault(); }
671         };
672     }
673     return e;
674 }
675
676 // --- Canvas touch events ---
677 $('#fbCanvas').on('touchstart', function(e) {
678     e.preventDefault(); // Prevent scrolling/zooming
679     var ne = normalizeEvent(e, $(this));
680     // Simulate mousedown logic
681     fbCanvasObj.check_for_drag = true;
682     fbCanvasObj.cacheUndoState();
683     updateMouseLoc(ne, $(this));
684     updateMouseAction('mousedown');
685     fbCanvasObj.mouseDownLoc = Point.createFromMouseEvent(ne, $(this));

```



```

686     var b = fbCanvasObj.barClickedOn();
687     var m = fbCanvasObj.matClickedOn();
688     // Copy from the mouse handler above
689     if( (fbCanvasObj.currentAction == 'bar') || (fbCanvasObj.currentAction == "mat")) {
690         fbCanvasObj.saveCanvas();
691     } else if( fbCanvasObj.currentAction == 'repeat' ) {
692         fbCanvasObj.addUndoState();
693         b.repeat(fbCanvasObj.mouseDownLoc);
694         fbCanvasObj.refreshCanvas();
695     } else {
696         if( b !== null ) {
697             if( $.inArray(b, fbCanvasObj.selectedBars) == -1) {
698                 if( !Utilities.shiftKeyDown ) {
699                     fbCanvasObj.clearSelection();
700                 }
701                 $.each( fbCanvasObj.selectedBars, function(index, bar) {
702                     bar.clearSplitSelection();
703                 });
704                 fbCanvasObj.barToFront(b);
705                 fbCanvasObj.selectedBars.push(b);
706                 b.isSelected = true;
707                 b.selectSplit(fbCanvasObj.mouseDownLoc);
708             } else {
709                 $.each( fbCanvasObj.selectedBars, function(index, bar) {
710                     bar.clearSplitSelection();
711                 });
712                 if( !Utilities.shiftKeyDown ) {
713                     b.selectSplit(fbCanvasObj.mouseDownLoc);
714                 } else {
715                     fbCanvasObj.removeBarFromSelection(b);
716                 }
717                 fbCanvasObj.barToFront(b);
718             }
719             if( fbCanvasObj.currentAction == "manualSplit" ) {
720                 fbCanvasObj.clearSelection();
721             }
722         } else if( m !== null ) {
723             if( $.inArray(m, fbCanvasObj.selectedMats) == -1) {
724                 if( !Utilities.shiftKeyDown ) {
725                     fbCanvasObj.clearSelection();
726                 }
727                 m.isSelected = true;
728                 fbCanvasObj.selectedMats.push(m);
729             } else {
730                 if( Utilities.shiftKeyDown ) {
731                     fbCanvasObj.removeMatFromSelection(m);
732                 }
733             }
734         } else {
735             fbCanvasObj.clearSelection();
736         }
737         fbCanvasObj.refreshCanvas();
738     }
739     ne.preventDefault();
740 });
741
742 $('#fbCanvas').on('touchmove', function(e) {
743     e.preventDefault(); // Prevent scrolling/zooming
744     var ne = normalizeEvent(e, $(this));
745     fracEvent = ne;
746     updateMouseLoc(ne, $(this));
747     updateMouseAction('mousemove');
748     var p = Point.createFromMouseEvent(ne, $(this));
749     if( fbCanvasObj.currentAction == "manualSplit" ) {
750         fbCanvasObj.manualSplitPoint = p;
751         fbCanvasObj.refreshCanvas();
752     }
753     if(fbCanvasObj.mouseDownLoc !== null) {
754         fbCanvasObj.updateCanvas(p);
755     }
756     ne.preventDefault();
757 });
758
759 $('#fbCanvas').on('touchend', function(e) {
760     e.preventDefault(); // Prevent scrolling/zooming

```

```

761         var ne = normalizeEvent(e, $(this));
762         updateMouseLoc(ne, $(this));
763         updateMouseAction('mouseup');
764         fbCanvasObj.mouseUpLoc = Point.createFromMouseEvent(ne, $(this));
765         if( fbCanvasObj.currentAction == 'bar' ) {
766             fbCanvasObj.addUndoState();
767             fbCanvasObj.addBar() ;
768             fbCanvasObj.clear_selection_button ();
769         } else if (fbCanvasObj.currentAction == 'mat') {
770             fbCanvasObj.addUndoState();
771             fbCanvasObj.addMat();
772             fbCanvasObj.clear_selection_button ();
773         }
774         if (fbCanvasObj.found_a_drag){
775             fbCanvasObj.finalizeCachedUndoState();
776             fbCanvasObj.check_for_drag = false;
777         }
778         fbCanvasObj.mouseUpLoc = null ;
779         fbCanvasObj.mouseDownLoc = null ;
780         fbCanvasObj.mouseLastLoc = null ;
781         ne.preventDefault();
782     });
783
784     // --- Touch for color pickers and tool buttons ---
785     $('<div>.colorBlock, .colorBlock1').on('touchstart', function(e) {
786         $(this).trigger('click');
787         e.preventDefault();
788     });
789     $('a').on('touchstart', function(e) {
790         $(this).trigger('click');
791         e.preventDefault();
792     });
793
794 });
795
796 function showAllButtons() {
797     while(hiddenButtons.length >0) {
798         thing = hiddenButtons.pop();
799         thing.show();
800     }
801     hiddenButtons = [];
802     hiddenButtonsName = [];
803 }
804
805 function SaveScreen() {
806     var r=window.confirm("Do you want to save?");
807     if (r==true)
808     {
809         fbCanvasObj.save();
810     }
811 }
812
813 function showButton(item) {
814     var cnt = 0;
815     while(hiddenButtonsName.length >0) {
816         if (hiddenButtonsName[cnt] === item) {
817             var rem_but1=hiddenButtonsName.splice(cnt, 1);
818             hiddenButtons.splice(cnt, 1);
819         }
820         else {
821             cnt++;
822         }
823         if (hiddenButtonsName.length === cnt) {
824             $(document.getElementById(rem_but1)).show();
825             break;
826         }
827     }
828 }
829
830 function hideButton(item) {
831     if (hiddenButtonsName.indexOf(item)<0) {
832         hidden=document.getElementById(item) ;
833         $(hidden).hide();
834         hiddenButtonsName.push(item);
835         hiddenButtons.push($(hidden));

```

```

836     }
837 }
838
839 function handleFileSelect(event) {
840     $("#dialog-file").dialog("close");
841     var files = event.target.files;
842     if (files.length === 0) {return;}
843
844     //First attempt
845     Utilities.file_list=event.target.files;
846     Utilities.file_index=0;
847
848     var aFile = files[0];
849     readFileOpen(aFile);
850     //
851 }
852
853 //First attempt
854 function handleListSelect(event) {
855     Utilities.file_index= document.getElementById('id_filetext').selectedIndex;
856     a_files = Utilities.file_list;
857
858     // SaveScreen();
859     fbCanvasObj.save();
860
861     var aFileIndex=Utilities.file_index;
862     var aFile = a_files[aFileIndex];
863     readFileOpen(aFile);
864
865 }
866 //
867
868 function nextSelectFile(){
869     // SaveScreen();
870     fbCanvasObj.save();
871
872     var n_files = Utilities.file_list;
873     Utilities.file_index = Utilities.file_index+1;
874     document.getElementById('id_filetext').selectedIndex = Utilities.file_index;
875
876     var nFileIndex=Utilities.file_index;
877     var nFile = n_files[nFileIndex];
878     readFileOpen(nFile);
879 }
880
881 function previousSelectFile(){
882     //SaveScreen();
883     fbCanvasObj.save();
884
885     var p_files = Utilities.file_list;
886     Utilities.file_index = Utilities.file_index-1;
887     document.getElementById('id_filetext').selectedIndex = Utilities.file_index;
888
889     var pFileIndex=Utilities.file_index;
890     var pFile = p_files[pFileIndex];
891     readFileOpen(pFile);
892 }
893
894 //First attempt
895 function readFileOpen(oFile){
896     showAllButtons();
897
898     // reset undo and redo
899     fbCanvasObj.mUndoArray = [];
900     fbCanvasObj.mRedoArray = [];
901
902     FBFileReader.readAsText(oFile);
903     FBFileReader.onload = function (oFile) {
904         fbCanvasObj.handleFileEvent(oFile);
905     }
906     showSelectList();
907 }
908 //
909
910

```

```

911 //First attempt
912 function showSelectList() {
913     f_files = Utilities.file_list;
914     var first = document.getElementById('id_filetext');
915     var b_title= document.getElementById('bar_titles');
916     var file_length = f_files.length;
917     var select_length = document.getElementById('id_filetext').selectedIndex;
918     var s_files = Utilities.file_list[Utilities.file_index];
919     select_length = select_length + 1;
920     document.title = s_files.name;
921     b_title.innerHTML=": "+s_files.name;
922
923     if(file_length===1){
924         hideButton("id_filetext");
925         hideButton("action_previous");
926         hideButton("action_next");
927     }
928     else if (file_length===select_length){
929         showButton("id_filetext");
930         showButton("action_previous");
931         hideButton("action_next");
932     }
933     else if (select_length===1 || select_length===0){
934         showButton("id_filetext");
935         hideButton("action_previous");
936         showButton("action_next");
937     }
938     else {
939         showButton("id_filetext");
940         showButton("action_previous");
941         showButton("action_next");
942     }
943
944     first.innerHTML='';
945     for (var i=0, f1; f1=f_files[i]; i++) {
946         if (s_files.name !== f1.name ) {
947             first.innerHTML=first.innerHTML+'<option value="' + f1.name +
948                 ↳ '>' + f1.name + '</option>';
949         }
950         else {
951             first.innerHTML=first.innerHTML+'<option value="' + f1.name +
952                 ↳ '"selected">' + f1.name + '</option>';
953         }
954     }
955 }
956 //
957
958 function resetFormElement(e) {
959     e.wrap('<form>').closest('form').get(0).reset();
960     e.unwrap();
961 }
962
963 // for debugging
964
965 function updateMouseLoc(e, elem) {
966     x = e.clientX - elem.position().left ;
967     y = e.clientY - elem.position().top ;
968     offsetX = elem.offset().left;
969     offsetY = elem.offset().top;
970     /*
971     $('#mouseLoc').text(x + ', ' + y + ' | ' + offsetX + ', ' + offsetY + ' | ' + window.pageXOffset
972     ↳ + ', ' + window.pageYOffset );
973     */
974 }
975
976 function updateMouseAction(actionName) {
977     /*
978     $('#mouseAction').text(actionName) ;
979     */
980 }

```

2.1.2 FractionBarsCanvas.js

```
1 // Copyright University of Massachusetts Dartmouth 2013
2 //
3 // Designed and built by James P. Burke and Jason Orrill
4 // Modified and developed by Hakan Sandir
5 //
6 // This Javascript version of Fraction Bars is based on
7 // the Transparent Media desktop version of Fraction Bars,
8 // which in turn was based on the original TIMA Bars software
9 // by John Olive and Leslie Steffe.
10 // We thank them for allowing us to update that product.
11
12
13
14 function FractionBarsCanvas(canvasContext) {
15     this.context = canvasContext ;
16     //     this.currentTool = '' ;
17     this.currentAction = '' ;
18     this.canvasState = null ;
19     this.currentFill = '#FFFF66' ;
20     //     this.barFill = '#FFFF66' ;
21     this.matFill = '#888888' ;
22     this.mouseDownLoc = null ;
23     this.mouseUpLoc = null ;
24     this.mouseLastLoc = null ;
25
26     this.bars = [] ;
27     this.mats = [] ;
28     this.selectedBars = [] ;
29     this.selectedMats = [] ;
30     this.lastSelectedBars = [] ;
31     this.lastSelectedMats = [] ;
32     this.unitBar = null ;
33     this.context.fillStyle = this.currentFill ;
34     this.context.font = '9pt Verdana' ;
35
36     this.mUndoArray = [] ;
37     this.mRedoArray = [] ;
38
39     this.check_for_drag = false; // These two values are used to check for a drag so that we can
40     this.found_a_drag = false; // store an undo state before a drag, and register it when we know the
41     ↪ drag happened
42
43     this.manualSplitPoint = null;
44 }
45
46 FractionBarsCanvas.prototype.addBar = function(a_bar) {
47     var b = null;
48     if (a_bar === null | a_bar === undefined) {
49         b = Bar.createFromMouse(this.mouseDownLoc, this.mouseUpLoc, 'bar', this.currentFill) ;
50     } else {
51         b = a_bar;
52     }
53
54     this.bars.push(b);
55     this.clearSelection();
56     this.updateSelectionFromState();
57     this.updateCanvas(this.mouseUpLoc);
58     // this.isSelected = true;
59     this.refreshCanvas();
60
61     // Utilities.Log(this.bars.length);
62 };
63
64 FractionBarsCanvas.prototype.addMat = function() {
65     var m = Mat.createFromMouse(this.mouseDownLoc, this.mouseUpLoc, 'mat', this.matFill) ;
66     this.mats.push(m);
67     this.updateCanvas(this.mouseUpLoc);
68     this.refreshCanvas();
69     // Utilities.Log(this.bars.length);
70 };
71
72 // Also copy mats
```

```

72 FractionBarsCanvas.prototype.copyBars = function() {
73     if( this.selectedBars.length > 0 ) {
74         for( var i = this.selectedBars.length-1; i >= 0; i-- ) {
75             this.bars.push( this.selectedBars[i].copy(true) );
76             this.selectedBars[i].isSelected = false ;
77         }
78     }
79     if( this.selectedMats.length > 0 ) {
80         for(var j = this.selectedMats.length-1; j >= 0; j-- ) {
81             this.mats.push( this.selectedMats[j].copy(true) );
82             this.selectedMats[j].isSelected = false ;
83         }
84     }
85     this.updateSelectionFromState();
86 };
87
88 FractionBarsCanvas.prototype.breakApartBars = function() {
89     var newBars ;
90     if( this.selectedBars.length > 0 ) {
91         for( var i = 0; i < this.selectedBars.length; i++ ) {
92             newBars = this.selectedBars[i].breakApart() ;
93             for( var j = 0; j < newBars.length; j++ ) {
94                 this.bars.push( newBars[j] );
95             }
96         }
97
98         // all splits in bars copied...delete the original selection
99         this.deleteSelectedBars() ;
100     }
101 };
102
103 FractionBarsCanvas.prototype.pullOutSplit = function() {
104     var sel_split = null;
105
106     for (var i = 0; i < this.selectedBars.length; i++) {
107         if (this.selectedBars[i].selectedSplit !== null) {
108             sel_split = this.selectedBars[i].selectedSplit;
109             var newbar = Bar.createFromSplit(sel_split, this.selectedBars[i].x,
110                 ↪ this.selectedBars[i].y);
111             this.addBar(newbar);
112         }
113     }
114 };
115
116 FractionBarsCanvas.prototype.clearSplits = function() {
117     if( this.selectedBars.length > 0 ) {
118         for( var i = 0; i < this.selectedBars.length; i++ ) {
119             this.selectedBars[i].clearSplits() ;
120         }
121     }
122 };
123
124 FractionBarsCanvas.prototype.split = function(sw) {
125     // This function opens the dialog, but doesn't actually perform the splits.
126     // makesplits is called directly from the OK button handler code in the .dialog definition in
127     ↪ fractionbars.js
128
129     if ((this.selectedBars.length > 1) || (this.selectedBars.length === 0)) {
130         ///////////////
131         if (Utilities.flag[3]) {
132             alert("Lütfen ayrıştırılacak bir kesir şeridi
133                 ↪ seçiniz.");
134         } else {
135             alert("Please select a bar to partition.");
136         }
137         //alert("Please select a bar to partition.");
138         // alert(window.getComputedStyle($('.c_split_alert')[0], ':before').getPropertyValue('content'));
139         //alert(getComputedStyle(document.querySelector('.c_split_alert'), ':before').content);
140
141     } else {
142         if( this.selectedBars.length > 0 ) {
143

```

```

144         // Show dialog
145         sw.color = this.selectedBars[0].color;
146         $( "#dialog-splits" ).dialog('open');
147         sw.refreshCanvas();
148         for( var i = 0; i < this.selectedBars.length; i++ ) {
149             // Do something to each bar
150         }
151     }
152 }
153 };
154
155 //değişecek
156 FractionBarsCanvas.prototype.properties = function() {
157     if (Utilities.flag[0] ) {
158         document.getElementById("new").checked = false;
159         document.getElementById("same").checked = true;
160     } else {
161         document.getElementById("new").checked = true;
162         document.getElementById("same").checked = false;
163     }
164     if (Utilities.flag[1] ) {
165         document.getElementById("two_horiz").checked = true;
166         document.getElementById("one_horiz").checked = false;
167     } else {
168         document.getElementById("two_horiz").checked = false;
169         document.getElementById("one_horiz").checked = true;
170     }
171     document.getElementById("vert").checked=true;
172     document.getElementById("horiz").checked=false;
173     $( "#dialog-properties" ).dialog('open');
174 };
175
176
177 FractionBarsCanvas.prototype.makeSplits = function(num_splits, vert_horiz, whole_part) {
178     var vert_truth = (vert_horiz === "Vertical");
179     if( this.selectedBars.length > 0 ) {
180         if (whole_part === "Whole") {
181             for( var i = 0; i < this.selectedBars.length; i++ ) {
182                 // Do something to each bar
183                 this_bar = this.selectedBars[i];
184                 // alert(num_splits);
185                 // this_bar.equalSplits(num_splits);
186
187                 this_bar.wholeBarSplits(num_splits, vert_truth);
188             }
189         } else {
190             if((this.selectedBars[0].splits.length === 0) ||
191                ↪ (this.selectedBars[0].selectedSplit === null)) {
192                 // No splits, or no selected split, so treat this like a whole bar split
193
194                 this.selectedBars[0].wholeBarSplits(num_splits, vert_truth);
195             } else {
196
197                 this.selectedBars[0].splitSelectedSplit(num_splits, vert_truth);
198             }
199         }
200         this.refreshCanvas();
201     }
202 };
203
204
205 FractionBarsCanvas.prototype.iterate = function(iw) {
206     // This function opens the dialog, but doesn't actually perform the iteration.
207     // makeIterations is called directly from the OK button handler code in the .dialog definition in
208     ↪ fractionbars.js
209
210     if ((this.selectedBars.length > 1) || (this.selectedBars.length === 0)) {
211         if (Utilities.flag[3]) {
212             alert("Lütfen yineleme işlemi yapabilmek için bir
213             ↪ kesir şeridi seçiniz.");
214         } else {
215             alert("Please select exactly one bar to
216             ↪ iterate.");
217         }
218     }
219 }

```

```

215         //alert("Please select exactly one bar to iterate.");
216     } else {
217         if( this.selectedBars.length > 0 ) {
218             // Show dialog
219             $( "#dialog-iterate" ).dialog('open');
220             //for( var i = 0; i < this.selectedBars.length; i++ ) {
221             // Do something to each bar
222             //}
223         }
224     }
225 };
226
227
228 FractionBarsCanvas.prototype.make = function(iw) {
229     // This function opens the dialog, but doesn't actually perform the iteration.
230     // makeIterations is called directly from the OK button handler code in the .dialog definition in
231     ↪ fractionbars.js
232
233     if ((this.selectedBars.length > 1) || (this.selectedBars.length === 0)) {
234         if (Utilities.flag[3]) {
235             alert("Lütfen yeni bir şerit yapabilmek için bir
236             ↪ kesir şeridi seçiniz.");
237         } else {
238             alert("Please select exactly one bar to make new
239             ↪ bar.");
240         }
241         //alert("Please select exactly one bar to iterate.");
242     } else {
243         if( this.selectedBars.length > 0 ) {
244             // Show dialog
245             $( "#dialog-make" ).dialog('open');
246         }
247     }
248 };
249
250 FractionBarsCanvas.prototype.makeIterations = function(num_iterations, vert_horiz) {
251     var vert_truth = (vert_horiz === "Vertical");
252     if( this.selectedBars.length > 0 ) {
253         if(!Utilities.flag[0]){this.copyBars();}
254         this.selectedBars[0].iterate(num_iterations, vert_truth);
255         this.refreshCanvas();
256     }
257 };
258
259
260 FractionBarsCanvas.prototype.makeMake = function(num_frac) {
261     if( this.selectedBars.length > 0 ) {
262         this.bars.push( this.selectedBars[0].makeNewCopy(num_frac) );
263         this.refreshCanvas();
264     }
265 };
266
267 FractionBarsCanvas.prototype.measureBars = function() {
268     if( this.selectedBars.length > 0 ) {
269         for( var i = this.selectedBars.length-1; i >= 0; i-- ) {
270             this.selectedBars[i].fraction =
271             ↪ Utilities.createFraction(this.selectedBars[i].size, this.unitBar.size) ;
272         }
273     }
274 };
275
276 FractionBarsCanvas.prototype.clearAllMeasurements = function() {
277     for( var i = 0; i < this.bars.length; i++ ) {
278         this.bars[i].isUnitBar = false ;
279         this.bars[i].fraction = '' ;
280     }
281 };

```



```

286
287
288 FractionBarsCanvas.prototype.setUnitBar = function() {
289     this.clearAllMeasurements() ;
290     if( this.selectedBars.length == 1 ) {
291         this.selectedBars[0].isUnitBar = true ;
292         this.selectedBars[0].fraction = '' ;
293         this.unitBar = this.selectedBars[0] ;
294     }
295 };
296
297 FractionBarsCanvas.prototype.editLabel = function() {
298     var canvasPos = $('#fbCanvas').position() ;
299
300     if( this.selectedBars.length == 1 ) {
301         var labelDiv = $('#labelInput') ;
302         $('#labelInput').css('position', 'absolute') ;
303         $('#labelInput').css('width', this.selectedBars[0].w - 13) ;
304
305         $('#labelInput').css('top', canvasPos.top + this.selectedBars[0].y +
306             ↪ this.selectedBars[0].h - labelDiv.outerHeight() - 4) ;
307         $('#labelInput').css('left', canvasPos.left + this.selectedBars[0].x + 5) ;
308         $('#labelInput').val( this.selectedBars[0].label ) ;
309
310         $('#labelInput').show() ;
311         $('#labelInput').focus() ;
312     }
313 };
314
315 FractionBarsCanvas.prototype.hideEditLabel = function() {
316     $('#labelInput').hide() ;
317 };
318
319 FractionBarsCanvas.prototype.saveLabel = function(labelText, selectionType) {
320     var barSelection = [] ;
321     if( selectionType == Utilities.USE_CURRENT_SELECTION ) {
322         barSelection = this.selectedBars ;
323     } else {
324         barSelection = this.lastSelectedBars ;
325     }
326
327     if( barSelection.length == 1 ) {
328         barSelection[0].label = labelText ;
329     }
330     this.lastSelectedBars = [] ;
331     this.refreshCanvas() ;
332 };
333
334 // Deletes both bars and mats that are selected
335 FractionBarsCanvas.prototype.deleteSelectedBars = function() {
336     var newBars = [] ;
337     var unitBarDeleted = false ;
338
339     for( var i = 0; i < this.bars.length; i++ ) {
340         if( !this.bars[i].isSelected ) {
341             newBars.push( this.bars[i] ) ;
342         } else {
343             if( this.bars[i].isUnitBar ) {
344                 unitBarDeleted = true ;
345             }
346         }
347     }
348     this.bars = newBars ;
349     if( unitBarDeleted ) {
350         this.clearAllMeasurements() ;
351     }
352     var newMats = [] ;
353     for (i = 0; i < this.mats.length; i++) {
354         if( !this.mats[i].isSelected ) {
355             newMats.push( this.mats[i] ) ;
356         }
357     }
358     this.mats = newMats;
359 };

```

```

360
361 // Works on bars and mats together
362 FractionBarsCanvas.prototype.updateSelectionFromState = function() {
363     this.selectedBars = [];
364     for( var i = 0; i < this.bars.length; i++ ) {
365         if( this.bars[i].isSelected ) {
366             this.selectedBars.push( this.bars[i] );
367         }
368     }
369     this.selectedMats = [];
370     for(i = 0; i < this.mats.length; i++ ) {
371         if( this.mats[i].isSelected ) {
372             this.selectedMats.push( this.mats[i] );
373         }
374     }
375 };
376
377 FractionBarsCanvas.prototype.findBarForPoint = function(p) {
378     for( var i = this.bars.length-1; i >= 0; i-- ) {
379         if( p.x > this.bars[i].x &&
380             p.x < this.bars[i].x + this.bars[i].w &&
381             p.y > this.bars[i].y &&
382             p.y < this.bars[i].y + this.bars[i].h ) {
383
384             return(this.bars[i]);
385         }
386     }
387     return null;
388 };
389
390 FractionBarsCanvas.prototype.findSplitForPoint = function(p) {
391     var the_bar = this.findBarForPoint(p);
392     if (the_bar !== null) {
393         return (the_bar.findSplitForPoint(p));
394     } else {
395         return (null);
396     }
397 };
398
399 FractionBarsCanvas.prototype.findSomethingForPoint = function(p) {
400     // Returns either a bar or a split that matches the point. Or null if no match.
401     var the_bar = this.findBarForPoint(p);
402     if (the_bar !== null) {
403         var the_split = the_bar.findSplitForPoint(p);
404         if (the_split !== null) {
405             return (the_split);
406         } else {
407             return (the_bar);
408         }
409     } else {
410         return (null);
411     }
412 };
413
414 FractionBarsCanvas.prototype.barClickedOn = function() {
415     for( var i = this.bars.length-1; i >= 0; i-- ) {
416         // Utilities.log(i);
417         if( this.mouseDownLoc.x > this.bars[i].x &&
418             this.mouseDownLoc.x < this.bars[i].x + this.bars[i].w &&
419             this.mouseDownLoc.y > this.bars[i].y &&
420             this.mouseDownLoc.y < this.bars[i].y + this.bars[i].h)
421         {
422             // this.bars[i].isSelected = true ;
423             if (this.currentAction == "manualSplit" ) {
424                 this.addUndoState();
425                 if (Utilities.flag[1] ) {
426                     split_key=Utilities.shiftKeyDown;
427                 } else{
428                     split_key=false;
429                 }
430
431                 this.bars[i].splitBarAtPoint(this.mouseDownLoc, split_key);
432             } else {
433                 this.bars[i].selectSplit(this.mouseDownLoc);
434             }

```

```

435         }
436         return this.bars[i] ;
437     }
438 }
439 return null ;
440 };
441
442 FractionBarsCanvas.prototype.barToFront = function(bar) {
443
444     var new_list = [];
445
446     for (var i = 0; i < this.bars.length; i++) {
447         if (bar !== this.bars[i]) {
448             new_list.push(this.bars[i]);
449         }
450     }
451     new_list.push(bar);
452     this.bars = new_list;
453
454 };
455
456 FractionBarsCanvas.prototype.matClickedOn = function() {
457     for( var i = this.mats.length-1; i >= 0; i-- ) {
458         // Utilities.log(i);
459         if( this.mouseDownLoc.x > this.mats[i].x &&
460             this.mouseDownLoc.x < this.mats[i].x + this.mats[i].w &&
461             this.mouseDownLoc.y > this.mats[i].y &&
462             this.mouseDownLoc.y < this.mats[i].y + this.mats[i].h)
463         {
464             //         this.mats[i].isSelected = true ;
465             return this.mats[i] ;
466         }
467     }
468     return null ;
469 };
470
471 // Clear for bars and mats
472 FractionBarsCanvas.prototype.clearSelection = function() {
473     $.each( this.bars, function(index, bar) {
474         bar.isSelected = false ;
475         bar.clearSplitSelection();
476     });
477     this.lastSelectedBars = this.selectedBars ;
478     this.selectedBars = [] ;
479
480     $.each( this.mats, function(index, mat) {
481         mat.isSelected = false ;
482     });
483     this.lastSelectedMats = this.selectedMats ;
484     this.selectedMats = [] ;
485 };
486
487 // Clear for bars and mats
488 FractionBarsCanvas.prototype.removeBarFromSelection = function(bar) {
489
490     var new_list = [];
491
492     for (var i = 0; i < this.selectedBars.length; i++) {
493         if (bar !== this.selectedBars[i]) {
494             new_list.push(this.selectedBars[i]);
495         }
496     }
497
498     this.selectedBars = new_list;
499     bar.isSelected = false;
500     bar.clearSplitSelection();
501
502 };
503
504 FractionBarsCanvas.prototype.removeMatFromSelection = function(mat) {
505
506     var new_list = [];
507
508     for (var i = 0; i < this.selectedMats.length; i++) {
509         if (mat !== this.selectedMats[i]) {

```

```

510         new_list.push(this.selectedMats[i]);
511     }
512 }
513
514 this.selectedMats = new_list;
515 mat.isSelected = false;
516
517 };
518
519
520 FractionBarsCanvas.prototype.joinSelected = function() {
521     // TODO: bulletproof this
522     // TODO: update this to allow for more than two bars to be joined.
523     if ((this.selectedBars.length > 2) || (this.selectedBars.length === 1) ||
524         ↪ (this.selectedMats.length > 0)) {
525         if (Utilities.flag[3]) {
526             alert("Birleştirme işlemi yapabilmek için lütfen
527                 ↪ iki kesir şeridi seçiniz.");
528         } else {
529             alert("Please select exactly two bars (and no
530                 ↪ mats) before attempting to Join.");
531         }
532         //alert("Please select exactly two bars (and no mats) before attempting to Join.");
533         {return;}
534     }
535     var success = this.selectedBars[0].join(this.selectedBars[1]);
536
537     if (success) {
538         this.selectedBars[0].isSelected = false ;
539         this.deleteSelectedBars();
540         this.updateSelectionFromState();
541     }
542 };
543
544 FractionBarsCanvas.prototype.setupBarRepeats = function() {
545     // For every bar, jsut set its repeatUnit. So that Repeat can work correctly.
546     for (var i = this.bars.length - 1; i >= 0; i--) {
547         this.bars[i].setRepeatUnit();
548     }
549 };
550
551 FractionBarsCanvas.prototype.unsetBarRepeats = function() {
552     // For every bar, jsut set its repeatUnit. So that Repeat can work correctly.
553     for (var i = this.bars.length - 1; i >= 0; i--) {
554         this.bars[i].repeatUnit = null;
555     }
556 };
557
558 FractionBarsCanvas.prototype.handleToolUpdate = function(tool_name, tool_on) {
559     // This is the Canvas' chance to do something when a tool switched on or off
560     // We are given the name of the tool, and a Boolean value of whether it was turned on or off.
561
562     switch(tool_name) {
563         case 'repeat':
564             if (tool_on) {
565                 this.setupBarRepeats();
566             } else {
567                 this.unsetBarRepeats();
568             }
569     }
570 };
571
572
573 FractionBarsCanvas.prototype.drawRect = function(p1, p2) {
574     if (this.currentAction == "bar")
575         this.context.fillStyle = this.currentFill;
576     else if (this.currentAction == "mat")
577         this.context.fillStyle = this.matFill;
578     var w = Math.abs(p2.x - p1.x) ;
579     var h = Math.abs(p2.y - p1.y) ;
580     var p = Point.min( p1, p2 ) ;

```

```

582     this.context.fillRect(p.x + 0.5, p.y + 0.5, w, h) ;
583     this.context.strokeRect(p.x + 0.5, p.y + 0.5, w, h) ;
584 };
585
586 /*
587 FractionBarsCanvas.prototype.manualSplitXORDraw = function(the_point) {
588     this.context.strokeStyle="#FF0000";
589     this.context.globalCompositeOperation="xor";
590     this.context.strokeRect(the_point.x-50, the_point.y-50, 100,100 ) ;
591     this.context.strokeRect(the_point.x-50, the_point.y-50, 100,100 ) ;
592     this.context.globalCompositeOperation="source-over";
593 }
594 */
595
596 FractionBarsCanvas.prototype.drawBar = function(b) {
597
598     this.context.fillStyle = b.color;
599     this.context.fillRect(b.x + 0.5, b.y + 0.5, b.w, b.h) ;
600
601     this.context.strokeStyle = '#FF0000' ;
602     if( b.splits.length > 0 ) {
603         for( i = 0; i < b.splits.length; i++ ) {
604             this.context.fillStyle = b.splits[i].color;
605             this.context.fillRect( b.x + b.splits[i].x + 0.5, b.y + b.splits[i].y + 0.5,
606                 ↪ b.splits[i].w, b.splits[i].h ) ;
607             this.context.strokeRect( b.x + b.splits[i].x + 0.5, b.y + b.splits[i].y + 0.5,
608                 ↪ b.splits[i].w, b.splits[i].h ) ;
609             if (b.splits[i].isSelected === true) {
610                 var xcenter = b.splits[i].x+(b.splits[i].w /2);
611                 var ycenter = b.splits[i].y+(b.splits[i].h /2);
612                 this.context.strokeRect(b.x+xcenter-2, b.y+ycenter-2, 4, 4);
613             }
614         }
615
616         this.context.fillStyle = b.color;
617
618         this.context.strokeStyle = '#000000' ;
619         if( b.isSelected ) {
620             this.context.lineWidth = 2.5 ;
621         }
622
623         this.context.strokeRect(b.x + 0.5, b.y + 0.5, b.w, b.h) ;
624
625         this.context.lineWidth = 1;
626         this.context.fillStyle = '#000000' ;
627
628         if( b.isUnitBar ) {
629             this.context.fillText('Unit Bar', b.x, b.y + b.h + 15) ;
630         }
631
632         if ((this.currentAction == "manualSplit") && (this.manualSplitPoint !== null)) {
633             var asplit = this.findSplitForPoint(this.manualSplitPoint);
634             var abar = this.findBarForPoint(this.manualSplitPoint);
635             var x_offset = 0;
636             var y_offset = 0;
637             var thing = null;
638
639             if (Utilities.flag[1] ) {
640                 split_key=!Utilities.shiftKeyDown;
641             } else
642             { split_key=true;
643             }
644             if (asplit !== null) {
645                 x_offset = abar.x;
646                 y_offset = abar.y;
647                 thing = asplit;
648             } else {
649                 thing = abar;
650             }
651             if ((thing !== null) && !((asplit === null) && (abar !== null) && (abar.splits.length !==
652                 ↪ 0))) {
653                 // The above statement is complex because it checks for the condition where a user
654                 ↪ can click

```

```

653         // exactly between existing splits.
654         var savestroke = this.context.strokeStyle;
655         this.context.strokeStyle = '#FF0000' ;
656
657         if (!split_key) {
658             this.context.strokeRect( thing.x+x_offset, this.manualSplitPoint.y,
659                                     ↪ thing.w,0 ) ;
660         } else {
661             this.context.strokeRect( this.manualSplitPoint.x, thing.y+y_offset, 0,
662                                     ↪ thing.h ) ;
663         }
664         this.context.strokeStyle = savestroke;
665     }
666 }
667
668 var fractionStringMetrics = this.context.measureText( b.fraction ) ;
669 this.context.fillText( b.fraction, b.x + b.w - fractionStringMetrics.width - 5, b.y - 5) ;
670
671 var labelStringMetrics = this.context.measureText( b.label ) ;
672 this.context.fillText( b.label, b.x + 5, b.y + b.h - 5) ;
673
674 this.context.fillStyle = this.currentFill ;
675 };
676
677 FractionBarsCanvas.prototype.drawMat = function(b) {
678
679     this.context.fillStyle = b.color;
680     this.context.fillRect(b.x + 0.5, b.y + 0.5, b.w, b.h) ;
681
682     this.context.strokeStyle = '#FF0000' ;
683
684     this.context.strokeStyle = '#000000' ;
685     if( b.isSelected ) {
686         this.context.lineWidth = 2.5 ;
687     }
688
689     this.context.strokeRect(b.x + 0.5, b.y + 0.5, b.w, b.h) ;
690
691     this.context.lineWidth = 1;
692     this.context.fillStyle = '#000000' ;
693
694     this.context.fillStyle = this.currentFill ;
695 }
696 };
697
698 FractionBarsCanvas.prototype.updateCanvas = function(currentMouseLoc) {
699
700     if ((this.currentAction == 'bar') || (this.currentAction == 'mat')) {
701         if( this.canvasState !== null ) {
702             this.context.putImageData(this.canvasState,0,0);
703         }
704         if( this.mouseDownLoc !== null ) {
705             this.drawRect(this.mouseDownLoc, currentMouseLoc) ;
706         }
707     } else if (this.currentAction == "manualSplit") {
708         // this.calculateSplitLine(currentMouseLoc);
709         this.manualSplitPoint = currentMouseLoc;
710     } else {
711         // we're dragging stuff around
712         this.drag(currentMouseLoc);
713     }
714 };
715
716 FractionBarsCanvas.prototype.saveCanvas = function() {
717     this.canvasState = this.context.getImageData(0,0,1000,600) ;
718 };
719
720 FractionBarsCanvas.prototype.refreshCanvas = function() {
721     this.context.clearRect(0,0,1000,600);
722     for( var i = 0; i < this.mats.length; i++ ) {
723         this.drawMat(this.mats[i]);
724     }
725     for( i = 0; i < this.bars.length; i++ ) {

```

```

726         this.drawBar(this.bars[i]);
727     }
728 };
729
730 FractionBarsCanvas.prototype.setFillColor = function(fillColor) {
731     this.currentFill = fillColor ;
732     this.context.fillStyle = this.currentFill ;
733 };
734
735 FractionBarsCanvas.prototype.updateColorsOfSelectedBars = function() {
736     var i;
737     if (this.selectedBars.length > 0) {
738         this.addUndoState();
739     }
740     for (i in this.selectedBars) {
741         if (this.selectedBars[i].hasSelectedSplit()) {
742             this.selectedBars[i].updateColorOfSelectedSplit(this.currentFill);
743         } else {
744             this.selectedBars[i].color = this.currentFill;
745         }
746     }
747     this.refreshCanvas();
748 };
749
750 FractionBarsCanvas.prototype.clearMouse = function() {
751     this.mouseDownLoc = null ;
752     this.mouseUpLoc = null ;
753 };
754
755 FractionBarsCanvas.prototype.drag = function(currentLoc) {
756     if( this.mouseLastLoc === null || typeof(this.mouseLastLoc) == 'undefined') {
757         this.mouseLastLoc = this.mouseDownLoc ;
758     }
759
760     for( var i = 0; i < this.selectedBars.length; i++ ) {
761         this.selectedBars[i].x = this.selectedBars[i].x + currentLoc.x - this.mouseLastLoc.x ;
762         this.selectedBars[i].y = this.selectedBars[i].y + currentLoc.y - this.mouseLastLoc.y ;
763     }
764
765     for(i = 0; i < this.selectedMats.length; i++ ) {
766         this.selectedMats[i].x = this.selectedMats[i].x + currentLoc.x - this.mouseLastLoc.x ;
767         this.selectedMats[i].y = this.selectedMats[i].y + currentLoc.y - this.mouseLastLoc.y ;
768     }
769
770 }
771
772 if(this.check_for_drag) {
773     this.found_a_drag = true;
774     this.check_for_drag = false;
775 }
776
777 this.mouseLastLoc = currentLoc ;
778
779 this.refreshCanvas() ;
780
781 };
782
783 FractionBarsCanvas.prototype.addUndoState = function() {
784
785     var newstate = new CanvasState(this);
786     newstate.grabBarsAndMats();
787     this.mUndoArray.push(newstate); // Push new state onto the stack
788
789     while (this.mUndoArray.length > 100) {
790         this.mUndoArray.shift(); // Shift states off the bottom of the undo stack
791     }
792
793     this.mRedoArray = []; // When an undoable event happens, it clears the redo stack.
794
795 };
796
797 FractionBarsCanvas.prototype.clear_selection_button = function() {
798
799     fbCanvasObj.clearMouse();
800     fbCanvasObj.clearSelection();

```

```

801         $(".[id~='tool_']").removeClass('toolSelected');
802         fbCanvasObj.currentAction = '';
803     };
804 };
805 FractionBarsCanvas.prototype.cacheUndoState = function() {
806     this.CachedState = new CanvasState(this);
807     this.CachedState.grabBarsAndMats();
808 };
809
810 };
811
812 FractionBarsCanvas.prototype.finalizeCachedUndoState = function() {
813     if(this.CachedState !== null){
814         this.mUndoArray.push(this.CachedState); // Push new state onto the stack
815     }
816     while (this.mUndoArray.length > 100) {
817         this.mUndoArray.shift(); // Shift states off the bottom of the undo stack
818     }
819     this.mRedoArray = []; // When an undoable event happens, it clears the redo stack.
820 }
821
822 this.check_for_drag = false;
823 this.found_a_drag = false;
824 };
825
826 FractionBarsCanvas.prototype.undo = function() {
827     // Store current state in Redo stack
828     // Pop an undo state off the stack
829     // Restore undo state
830     if (this.mUndoArray.length > 0) {
831         var newstate = new CanvasState(this);
832         newstate.grabBarsAndMats();
833         this.mRedoArray.push(newstate); // Push new state onto the stack
834     }
835     this.restoreAState(this.mUndoArray.pop());
836 }
837 };
838
839 FractionBarsCanvas.prototype.redo = function() {
840     if (this.mRedoArray.length > 0) {
841         var newstate = new CanvasState(this);
842         newstate.grabBarsAndMats();
843         this.mUndoArray.push(newstate); // Push new state onto the stack
844     }
845     this.restoreAState(this.mRedoArray.pop());
846 }
847 };
848
849 FractionBarsCanvas.prototype.restoreAState = function(a_new_state) {
850     // clear the bars and mats
851     // copy bars and mats from the new state
852     // set the unit bar, if any.
853     var temp_bar;
854
855     this.bars = [];
856     this.mats = [];
857     this.selectedBars = [];
858     this.selectedMats = [];
859
860     while (a_new_state.mBars.length > 0) {
861         temp_bar = a_new_state.mBars.shift();
862         this.bars.push(temp_bar);
863     }
864
865     while (a_new_state.mMats.length > 0) {

```



```

876         this.mats.push(a_new_state.mMats.shift());
877     }
878
879     this.unitBar = a_new_state.mUnitBar;
880     if (this.unitBar !== null ) {
881         this.unitBar.isUnitBar = true;
882         this.unitBar.fraction = '1/1' ;
883     }
884     //this.updateSelectionFromState();
885     this.clearSelection();
886
887 };
888
889
890
891 FractionBarsCanvas.prototype.save = function() {
892
893     var newstate = new CanvasState(this);
894     newstate.grabBarsAndMats();
895
896     newstate.mFBCanvas = null;
897
898     var state_string = JSON.stringify(JSON.decycle(newstate));
899
900     // alert(state_string);
901     // Utilities.log(state_string);
902     /*
903     var new_win = window.open("", "_blank", "resizable=yes, scrollbars=yes, titlebar=yes, width=1000,
904     ↪ height=500, top=10, left=10");
905     new_win.document.title = "Save this in a file on your hard drive.";
906     new_win.document.writeln("** Save this text to your hard drive. Right-click here and use 'Save
907     ↪ as...' or 'Save page as...'");
908     new_win.document.writeln("**");
909     new_win.document.writeln(state_string);
910     new_win.document.close();
911     returns false if user does not save
912     */
913     try {
914         var blob = new Blob([state_string], {type: "text/plain;charset=utf-8"});
915         //var filename = window.prompt("File name:", "FractionBarsSave.txt");
916
917         // first attempt
918         var select_length = document.getElementById('id_filetext').selectedIndex;
919         if(select_length<0)
920         {
921             var filename = window.prompt("File name:", "FractionBarsSave.txt");
922         }
923         else
924         {
925             var filename = Utilities.file_list[Utilities.file_index].name;
926         }
927
928         //
929         if (filename!=null)
930         {
931             saveAs(blob, filename);
932         }
933         else
934         {
935             return false;
936         }
937     }
938     catch(e){
939         if (Utilities.flag[3]) {
940
941             alert("Bu tarayıcı kaydetmeyi desteklememektedir.
942             ↪ Tarayıcının \nHTML5 destekli olması
943             ↪ gereklidir. \n\nEn iyi sonuç için lütfen
944             ↪ Firefox, \nChrome, Safari ya da Internet
945             ↪ Explorer tarayıcılarından birini
946             ↪ kullanınız.");
947         } else {

```

```

942         alert("This browser does not support saving.
↪ \nHTML5 support is needed. \n\nFor best
↪ results use the most recent Firefox, \nChrome,
↪ Safari, or Internet Explorer browser.");
943     }
944 }
945 //alert("This browser does not support saving. \nHTML5 support is needed. \n\nFor best
↪ results use the most recent Firefox, \nChrome, Safari, or Internet Explorer browser.");
946 }
947 };
948
949 FractionBarsCanvas.prototype.openFileDialog = function() {
950     // Show dialog
951     $( "#dialog-file" ).dialog('open');
952 };
953
954 FractionBarsCanvas.prototype.openSaveDialog = function() {
955     // Show dialog
956     var r=window.confirm("Do you want to save?");
957     if (r==true)
958     {
959         /*var res=this.save();
960         if (res==false)
961         {
962             break;
963         }*/
964     }
965 };
966 FractionBarsCanvas.prototype.handleFileEvent = function(file_event) {
967
968
969     var file_contents = file_event.target.result;
970     // var lines = file_contents.split("**");
971     // var text_state = lines[2].replace(/(\r\n|\n|\r)/gm, "");
972
973     var text_state = "";
974     var something = null;
975
976     try {
977         text_state = file_contents.replace(/(\r\n|\n|\r)/gm, "");
978         something = JSON.retrocycle(JSON.parse(text_state));
979     } catch (e) {
980         var txt = "An error has occurred. \n\n";
981         txt += "Fraction Bars cannot open this file. \n\n";
982         txt += e.message;
983         alert(txt);
984         return;
985     }
986
987     this.restoreBarsAndMatsFromJSON(something);
988 };
989
990 FractionBarsCanvas.prototype.restoreBarsAndMatsFromJSON = function(JSON_obj) {
991
992     this.bars = [];
993     this.mats = [];
994     this.selectedBars = [];
995     this.selectedMats = [];
996     this.unitBar = null;
997     len = 0;
998
999     if( JSON_obj.mBars.length > 0 ) {
1000         for( var i = 0; i < JSON_obj.mBars.length; i++ ) {
1001             len = this.bars.push( Bar.copyFromJSON(JSON_obj.mBars[i]) );
1002             if (this.bars[len-1].isUnitBar) {
1003                 this.unitBar = this.bars[len-1];
1004                 this.bars[len-1].fraction = "1/1";
1005             }
1006         }
1007     }
1008
1009     if( JSON_obj.mMats.length > 0 ) {
1010         for( var j = 0; j < JSON_obj.mMats.length; j++ ) {
1011             this.mats.push( Mat.copyFromJSON(JSON_obj.mMats[j]) );
1012         }
1013     }

```

```

1013     }
1014
1015     //First attempt
1016     var hiddenButtonsName1 = JSON_obj.mHidden.slice(0);
1017     for( var ii = 0; ii < hiddenButtonsName1.length; ii++ ) {
1018         if (hiddenButtonsName.indexOf(hiddenButtonsName1[ii])<0) {
1019             hidden=document.getElementById(hiddenButtonsName1[ii]) ;
1020
1021             $(hidden).hide();
1022             hiddenButtonsName.push(hiddenButtonsName1[ii]);
1023             hiddenButtons.push($(hidden));
1024         }
1025     }
1026     //
1027
1028     Utilities.ctrlKeyDown=true;
1029     Utilities.ctrlKeyDown=true;
1030     this.clearSelection();
1031     this.refreshCanvas();
1032 };
1033
1034 FractionBarsCanvas.prototype.print_canvas = function () {
1035     var canvas=document.getElementById("fbCanvas");
1036     //var ctx=canvas.canvasContext;
1037     var win=window.open();
1038     win.document.write("<html><br><img src='"+canvas.toDataURL()+"'/></html>");
1039     //win.print();
1040     win.self.print();
1041     win.location.reload();
1042 }
1043
1044 FractionBarsCanvas.prototype.exportHighResPNG = function(filename) {
1045     var scale = 3;
1046     var canvas = document.getElementById('fbCanvas');
1047     var w = canvas.width;
1048     var h = canvas.height;
1049     // Create a high-res offscreen canvas
1050     var exportCanvas = document.createElement('canvas');
1051     exportCanvas.width = w * scale;
1052     exportCanvas.height = h * scale;
1053     var exportCtx = exportCanvas.getContext('2d');
1054     // Scale context
1055     exportCtx.setTransform(scale, 0, 0, scale, 0, 0);
1056     // Redraw everything at high-res
1057     this.drawAllToContext(exportCtx);
1058     // Export PNG
1059     var dataURL = exportCanvas.toDataURL('image/png');
1060     var link = document.createElement('a');
1061     link.href = dataURL;
1062     link.download = filename || 'FractionBars.png';
1063     document.body.appendChild(link);
1064     link.click();
1065     document.body.removeChild(link);
1066 };
1067
1068 // Helper to draw all content to a given context (for export)
1069 FractionBarsCanvas.prototype.drawAllToContext = function(ctx) {
1070     ctx.clearRect(0, 0, ctx.canvas.width, ctx.canvas.height);
1071     for (var i = 0; i < this.mats.length; i++) {
1072         this.drawMatToContext(ctx, this.mats[i]);
1073     }
1074     for (var i = 0; i < this.bars.length; i++) {
1075         this.drawBarToContext(ctx, this.bars[i]);
1076     }
1077 };
1078
1079 FractionBarsCanvas.prototype.drawBarToContext = function(ctx, b) {
1080     ctx.save();
1081     ctx.fillStyle = b.color;
1082     ctx.fillRect(b.x + 0.5, b.y + 0.5, b.w, b.h);
1083     ctx.strokeStyle = '#FF0000';
1084     if (b.splits.length > 0) {
1085         for (var i = 0; i < b.splits.length; i++) {
1086             ctx.fillStyle = b.splits[i].color;

```

```

1087         ctx.fillRect(b.x + b.splits[i].x + 0.5, b.y + b.splits[i].y + 0.5, b.splits[i].w,
1088             ↪ b.splits[i].h);
1089         ctx.strokeStyle = '#000000';
1090         ctx.strokeRect(b.x + b.splits[i].x + 0.5, b.y + b.splits[i].y + 0.5,
1091             ↪ b.splits[i].w, b.splits[i].h);
1092         if (b.splits[i].isSelected === true) {
1093             var xcenter = b.splits[i].x + (b.splits[i].w / 2);
1094             var ycenter = b.splits[i].y + (b.splits[i].h / 2);
1095             ctx.strokeRect(b.x + xcenter - 2, b.y + ycenter - 2, 4, 4);
1096         }
1097     }
1098     ctx.fillStyle = b.color;
1099     ctx.strokeStyle = '#000000';
1100     if (b.isSelected) {
1101         ctx.lineWidth = 2.5;
1102     }
1103     ctx.strokeRect(b.x + 0.5, b.y + 0.5, b.w, b.h);
1104     ctx.lineWidth = 1;
1105     ctx.fillStyle = '#000000';
1106     ctx.font = '9pt Verdana';
1107     if (b.isUnitBar) {
1108         ctx.fillText('Unit Bar', b.x, b.y + b.h + 15);
1109     }
1110     var fractionStringMetrics = ctx.measureText(b.fraction);
1111     ctx.fillText(b.fraction, b.x + b.w - fractionStringMetrics.width - 5, b.y - 5);
1112     var labelStringMetrics = ctx.measureText(b.label);
1113     ctx.fillText(b.label, b.x + 5, b.y + b.h - 5);
1114     ctx.fillStyle = this.currentFill;
1115     ctx.restore();
1116 };
1117 FractionBarsCanvas.prototype.drawMatToContext = function(ctx, b) {
1118     ctx.save();
1119     ctx.fillStyle = b.color;
1120     ctx.fillRect(b.x + 0.5, b.y + 0.5, b.w, b.h);
1121     ctx.strokeStyle = '#000000';
1122     ctx.strokeRect(b.x + 0.5, b.y + 0.5, b.w, b.h);
1123     ctx.restore();
1124 };

```

2.2 Object Classes

2.2.1 Bar.js

```

1  // Copyright University of Massachusetts Dartmouth 2013
2  //
3  // Designed and built by James P. Burke and Jason Orrill
4  // Modified and developed by Hakan Sandir
5  //
6  // This Javascript version of Fraction Bars is based on
7  // the Transparent Media desktop version of Fraction Bars,
8  // which in turn was based on the original TIMA Bars software
9  // by John Olive and Leslie Steffe.
10 // We thank them for allowing us to update that product.
11
12
13 function Bar() {
14     //TODO: convert x, y to a Point?
15     this.x = null ;
16     this.y = null ;
17     this.w = null ;
18     this.h = null ;
19     this.size = null ;
20     this.color = null ;
21     this.splits = [] ;
22     this.label = '' ;
23     this.isUnitBar = false ;
24     this.fraction = '' ;
25     this.type = null ;
26     this.isSelected = false ;
27     this.repeatUnit = null;    // This is a copy of whatever the bar looks like at the moment "repeat"
    ↪ mode is turned on.

```

```

28         this.selectedSplit = null;
29     }
30
31     Bar.prototype.measure = function(targetBar) {
32         this.fraction = Utilities.createFraction( this.size, targetBar.size );
33     };
34     Bar.prototype.clearMeasurement = function() {
35         this.fraction = '' ;
36     };
37     Bar.prototype.drawMeasurement = function() {};
38
39
40     Bar.prototype.addSplit = function(x, y, w, h, c) {
41         this.addSplitToList(this.splits, x, y, w, h, c);
42     };
43
44     Bar.prototype.addSplitToList = function(list, x, y, w, h, c) {
45         var split = new Split(x, y, w, h, c) ;
46         if( this.splits.length > 0 ) {
47             for( i = 0; i < this.splits.length; i++ ) {
48                 if( split.equals( this.splits[i] ) ) {
49                     return ;
50                 }
51             }
52         }
53         list.push( split ) ;
54     };
55
56     Bar.prototype.clearSplits = function() {
57         this.splits = [] ;
58     };
59
60     Bar.prototype.copySplits = function() {
61         var splitsCopy = [] ;
62
63         for( var i = 0; i < this.splits.length; i++ ) {
64             splitsCopy.push( this.splits[i].copy() ) ;
65         }
66
67         return splitsCopy ;
68     };
69
70     Bar.prototype.hasSelectedSplit = function() {
71         for (var i = 0; i < this.splits.length; i++) {
72             if (this.splits[i].isSelected) {
73                 return true;
74             }
75         }
76         return false;
77     };
78
79     Bar.prototype.updateColorOfSelectedSplit = function(in_color) {
80         for (var i = 0; i < this.splits.length; i++) {
81             if (this.splits[i].isSelected) {
82                 this.splits[i].color = in_color;
83             }
84         }
85         return false;
86     };
87
88     Bar.prototype.clearSplitSelection = function() {
89         this.selectedSplit = null;
90         for (var i = 0; i < this.splits.length; i++) {
91             this.splits[i].isSelected = false;
92         }
93     };
94
95
96     Bar.prototype.updateSplitSelectionFromState = function() {
97         this.selectedSplit = null;
98         for( var i = 0; i < this.splits.length; i++ ) {
99             if( this.splits[i].isSelected ) {
100                 this.selectedSplit = this.splits[i];
101             }
102         }

```

```

103 };
104
105
106 Bar.prototype.selectSplit = function(mouse_loc) {
107
108     this.selectedSplit = this.splitClickedOn(mouse_loc);
109 };
110
111 Bar.prototype.findSplitForPoint = function(p) {
112     for( var i = this.splits.length-1; i >= 0; i-- ) {
113         if( p.x > this.splits[i].x+this.x &&
114             p.x < this.splits[i].x+this.x + this.splits[i].w &&
115             p.y > this.splits[i].y+this.y &&
116             p.y < this.splits[i].y+this.y + this.splits[i].h ) {
117
118             return(this.splits[i]);
119         }
120     }
121     return null;
122 };
123
124 Bar.prototype.splitClickedOn = function(mouse_loc) {
125     for( var i = this.splits.length-1; i >= 0; i-- ) {
126         // Utilities.log(i);
127         if( mouse_loc.x > this.x + this.splits[i].x &&
128             mouse_loc.x < this.x + this.splits[i].x + this.splits[i].w &&
129             mouse_loc.y > this.y + this.splits[i].y &&
130             mouse_loc.y < this.y + this.splits[i].y + this.splits[i].h )
131         {
132             this.clearSplitSelection();
133             this.splits[i].isSelected = true ;
134             return this.splits[i] ;
135         }
136     }
137     return null ;
138 };
139
140 Bar.prototype.removeASplit = function(split) {
141     // If this bar has this split, remove it.
142     var newsplits = [];
143     for (var i = this.splits.length - 1; i >= 0; i--) {
144         if (this.splits[i] !== split) {
145             newsplits.push(this.splits[i]);
146         }
147     }
148     this.splits = newsplits;
149 };
150
151
152
153 //////////////////////////////////////////////////
154 Bar.prototype.splitBarAtPoint = function(split_point, vert_split) {
155     var the_split = this.findSplitForPoint(split_point);
156
157     if ((the_split !== null)) {
158
159         // Splitting a single split
160         if (!vert_split) {
161             this.addSplit(the_split.x, the_split.y, split_point.x-(this.x+the_split.x),
162                 ↪ the_split.h, the_split.color);
163             this.addSplit(split_point.x-this.x, the_split.y,
164                 ↪ the_split.w-(split_point.x-(this.x+the_split.x)), the_split.h,
165                 ↪ the_split.color);
166         } else {
167             this.addSplit(the_split.x, the_split.y, the_split.w,
168                 ↪ split_point.y-(this.y+the_split.y), the_split.color);
169             this.addSplit(the_split.x, split_point.y-this.y, the_split.w,
170                 ↪ the_split.h-(split_point.y-(this.y+the_split.y)),
171                 ↪ the_split.color);
172         }
173         this.removeASplit(the_split);
174     } else {
175         // Adding a split to a bar with none
176         if (this.splits.length == 0) {
177             // Make sure we really have no splits before doing this.

```

```

172         if (!vert_split) {
173             this.addSplit(0, 0, split_point.x-this.x, this.h, this.color);
174             this.addSplit(split_point.x-this.x, 0, this.x+this.w-split_point.x,
175                 ↪ this.h, this.color);
176         } else {
177             this.addSplit(0, 0, this.w, split_point.y-this.y, this.color);
178             this.addSplit(0, split_point.y-this.y, this.w,
179                 ↪ this.y+this.h-split_point.y, this.color);
180         }
181     }
182 };
183
184 Bar.prototype.initialSplits = function(num_splits, vert_direction) {
185     // Used for when there are no existing splits in a bar
186     var split_interval = 0;
187     var x = 0;
188     var y = 0;
189     var i = 0;
190
191     if (vert_direction === true) {
192         split_interval = this.w / num_splits;
193         for ( i = 0; i < num_splits; i++) {
194             x = i*split_interval;
195             this.addSplit(x, y, split_interval, this.h, this.color);
196         }
197     } else {
198         split_interval = this.h / num_splits;
199         for ( i = 0; i < num_splits; i++) {
200             y = i*split_interval;
201             this.addSplit(x, y, this.w, split_interval, this.color);
202         }
203     }
204 };
205
206 Bar.prototype.splitSelectedSplit = function(num_splits, vert_direction) {
207
208     this.updateSplitSelectionFromState();
209     if (this.selectedSplit === null) {return;}
210
211     var split_interval = 0;
212     var x = this.selectedSplit.x;
213     var y = this.selectedSplit.y;
214     var i = 0;
215
216     if (vert_direction === true) {
217         split_interval = this.selectedSplit.w / num_splits;
218         for ( i = 0; i < num_splits; i++) {
219             x = i*split_interval + this.selectedSplit.x;
220             this.addSplit(x, y, split_interval, this.selectedSplit.h,
221                 ↪ this.selectedSplit.color);
222         }
223     } else {
224         split_interval = this.selectedSplit.h / num_splits;
225         for ( i = 0; i < num_splits; i++) {
226             y = i*split_interval +this.selectedSplit.y;
227             this.addSplit(x, y, this.selectedSplit.w, split_interval,
228                 ↪ this.selectedSplit.color);
229         }
230     }
231     var place = 0;
232     while (this.splits[place] !== this.selectedSplit) {
233         place++;
234     }
235     this.splits.splice(place, 1);
236 };
237
238 Bar.prototype.wholeBarSubSplit = function(a_split, vert_direction, subsplit_interval) {
239     // Takes a single split and tries to subsplit it based on a whole bar fraction
240     // If there is no subsplit, we just return a list containing the original split.
241     // Otherwise we return the subsplits, but not the original split
242     var new_subsplit_list = [];

```

```

243     var i = 0;
244     var split_hit = false; // Start out assuming we have not hit a split
245     var lower_bound = 0; //storing the lower boundary of a new split
246     var upper_bound = 0; // self explanatory
247     var corrected_interval = 0; // storing the corrected width or height of a new split (in case it is
    ↪ cut off)

248
249     if (vert_direction === true) {
250         for (i = subsplit_interval; Math.floor(i) <= this.w; i = i + subsplit_interval) {
251             if ( ((i > a_split.x) && (i -
    ↪ < a_split.x + a_split.w)) ||
252                 ((i - subsplit_interval > a_split.x) && (i -
    ↪ subsplit_interval < a_split.x + a_split.w)) ) {
253                 split_hit = true;
254                 lower_bound = (a_split.x > (i - subsplit_interval)) ? a_split.x : (i -
    ↪ subsplit_interval);
255                 upper_bound = ((a_split.x + a_split.w) < i) ? a_split.x + a_split.w : i;
256                 corrected_interval = upper_bound - lower_bound;
257                 this.addSplitToList(new_subsplit_list, lower_bound, a_split.y,
    ↪ corrected_interval, a_split.h, a_split.color);
258             }
259         }
260     } else {
261         for (i = subsplit_interval; Math.floor(i) <= this.h; i = i + subsplit_interval) {
262             if ( ((i > a_split.y) && (i -
    ↪ < a_split.y + a_split.h)) ||
263                 ((i - subsplit_interval > a_split.y) && (i -
    ↪ subsplit_interval < a_split.y + a_split.h)) ) {
264                 split_hit = true;
265                 lower_bound = (a_split.y > (i - subsplit_interval)) ? a_split.y : (i -
    ↪ subsplit_interval);
266                 upper_bound = ((a_split.y + a_split.h) < i) ? a_split.y + a_split.h : i;
267                 corrected_interval = upper_bound - lower_bound;
268                 this.addSplitToList(new_subsplit_list, a_split.x, lower_bound, a_split.w,
    ↪ corrected_interval, a_split.color);
269             }
270         }
271     }
272     if (split_hit === false) {
273         new_subsplit_list.push(a_split);
274     }
275     return new_subsplit_list;
276 };

277
278
279
280 Bar.prototype.wholeBarSplits = function(num_splits, vert_direction) {
281     // Tries to split a whole bar, despite subsplits.
282     var new_splits_list = [];
283     var split_interval = 0;
284     var list_passback = [];
285
286     if (this.splits.length === 0) {
287         // Doing initial splits because there are no existing splits
288         this.initialSplits(num_splits, vert_direction);
289     } else {
290         // Doing subsequent splits because we already have some splits
291         if (vert_direction === true) {
292             split_interval = this.w / num_splits;
293         } else {
294             split_interval = this.h / num_splits;
295         }
296         // For every split
297         for (var i = this.splits.length - 1; i >= 0; i--) {
298             // Attempt to subsplit it and concat the result into the new list
299             list_passback = this.wholeBarSubSplit(this.splits[i], vert_direction,
    ↪ split_interval);
300             new_splits_list = new_splits_list.concat(list_passback);
301         }
302         // When complete, use the new list to replace the old list.
303         this.clearSplits();
304         this.splits = new_splits_list;
305     }
306 };
307

```



```

308
309
310 Bar.prototype.breakApart = function() {
311     var newBars = [] ;
312     var aBar ;
313     if( this.splits.length === 0 ) {
314         aBar = this.copy(false) ;
315         aBar.isSelected = false ;
316         newBars.push( aBar ) ;
317     } else {
318         for( var i = 0; i < this.splits.length; i++ ) {
319             newBars.push( Bar.create( this.x + this.splits[i].x, this.y + this.splits[i].y,
320                                     ↪ this.splits[i].w, this.splits[i].h, 'bar', this.splits[i].color )) ;
321         }
322     }
323 };
324
325 Bar.prototype.copy = function(with_offset) {
326     var offset = 10 ;
327     var b = new Bar() ;
328
329     if (with_offset === false) {
330         offset = 0;
331     }
332     if (fbCanvasObj.currentAction == "repeat") {
333         offset=0;
334     }
335     b.x = this.x + offset ;
336     b.y = this.y + offset ;
337     b.w = this.w ;
338     b.h = this.h ;
339     b.size = this.size ;
340     b.color = this.color ;
341     b.splits = this.copySplits() ;
342     b.label = this.label ;
343     b.isUnitBar = false ;
344     if (this.isUnitBar === true) {
345         b.fraction = "" ;
346     } else {
347         b.fraction = this.fraction ;
348     }
349     b.type = this.type ;
350     b.isSelected = true ;
351     b.repeatUnit = this.repeatUnit;
352
353     return b ;
354 };
355
356
357 Bar.prototype.makeCopy = function() {
358
359     // This version of the copy routine does not set isSelected to true.
360     // I a using this to make a copy that is just stored, so there is no reason for
361     // the bar to think it is selected.
362
363     var offset = 10 ;
364     var b = new Bar() ;
365
366     b.x = this.x + offset ;
367     b.y = this.y + offset ;
368     b.w = this.w ;
369     b.h = this.h ;
370     b.size = this.size ;
371     b.color = this.color ;
372     b.splits = this.copySplits() ;
373     b.label = this.label ;
374     b.isUnitBar = false ;
375     b.fraction = this.fraction ;
376     b.type = this.type ;
377     b.isSelected = false ;
378
379     return b ;
380 };
381

```

```

382 Bar.prototype.makeNewCopy = function(with_height) {
383
384     // This version of the copy routine does not set isSelected to true.
385     // I a using this to make a copy that is just stored, so there is no reason for
386     // the bar to think it is selected.
387
388     var offset = 10 ;
389
390     var b = new Bar() ;
391
392     b.x = this.x ;
393     b.y = this.y + this.h + offset ;
394     b.w = this.w * with_height;
395     b.h = this.h ;
396     b.size = this.size * with_height;
397     b.color = this.color ;
398     b.isUnitBar = false ;
399     b.type = this.type ;
400     b.isSelected = false ;
401     this.isSelected = false ;
402     return b ;
403 };
404
405 Bar.prototype.repeat = function(clickLoc) {
406
407     //      alert(clickLoc.x +", "+clickLoc.y);
408
409     govert = false;
410     /*      local_x = clickLoc.x - this.x;
411            local_y = clickLoc.y - this.y;
412            diag_slope = this.h / this.w;
413            // modified by hsandir
414            if (local_y > (local_x * diag_slope)) {
415                govert = true;
416            }*/
417
418     if (this.repeatUnit !== null) {
419         if (gover) {
420             this.repeatUnit.x -= 5;
421         } else {
422             this.repeatUnit.y -= 5;
423         }
424         this.join(this.repeatUnit);
425         if ((this.splits.length === 2) && (this.repeatUnit.splits.length === 0) &&
            ↪ Utilities.getMarkedIterateFlag()) {
426             this.splits[1].color = this.splits[0].color;
427             ///////////////////////////////////////////////////
428             //      this.splits[1].color = Utilities.colorLuminance(this.splits[0].color.toString(),
            ↪ -0.1);
429         }
430     } else {
431         alert("Tried to Repeat when no repeatUnit was set.");
432     }
433 };
434
435
436
437 Bar.prototype.iterate = function(iterate_num, vert) {
438
439     offset = 3;
440     i_iter = 0;
441
442     iterate_unit = this.makeCopy();
443     if (vert === true) {
444         iterate_unit.y += offset;
445     } else {
446         iterate_unit.x += offset;
447     }
448
449     start_split_num = this.splits.length;
450
451     for (i_iter = 1; i_iter < iterate_num; i_iter++) {
452         this.join(iterate_unit);
453     }
454

```

```

455         if((start_split_num === 0) && (this.splits.length >0) && Utilities.getMarkedIterateFlag()) {
456             //this.splits[0].color = Utilities.colorLuminance(this.splits[0].color.toString(), -0.1);
457         }
458     };
459
460     Bar.prototype.join = function(b) {
461         var gap = Bar.distanceBetween(this,b);
462         gap.x = Math.abs(gap.x);
463         gap.y = Math.abs(gap.y);
464         var b1, b2 ;
465         var originalBar = this.copy(true) ;
466         var joinDimension = '' ;
467
468         // TODO: add check for matching dimensions
469
470         var vertmatch = this.h == b.h;
471         var horizmatch = this.w == b.w;
472
473         if (!vertmatch && !horizmatch) {
474             alert("To Join, bars must have a matching dimension in height or width.");
475             return(false);
476         }
477
478         //
479         //     this.x = Math.min(this.x, b.x) ;
480         //     this.y = Math.min(this.y, b.y) ;
481
482         if (vertmatch && horizmatch) { // since both match, determine join dimension
483             //         alert("Both match!");
484
485             if( Math.abs(gap.x) < Math.abs(gap.y) ) {
486                 this.h = this.h + b.h ;
487                 joinDimension = 'w' ;
488             } else {
489                 this.w = this.w + b.w ;
490                 joinDimension = 'h' ;
491             }
492         } else { // just one matched
493             if (vertmatch) {
494                 //         alert("Only h matched!");
495                 this.w = this.w + b.w ;
496                 joinDimension = 'h';
497             } else {
498                 //         alert("Only w matched!");
499                 this.h = this.h + b.h ;
500                 joinDimension = 'w';
501             }
502         }
503
504         this.size = this.w * this.h ;
505
506         var i = 0;
507
508         this.clearSplits();
509
510         // handling will be different for vertical/horizontal joins
511         if( joinDimension == 'w' ) {
512             //         alert("Joining along width");
513             if( originalBar.y < b.y ) {
514                 b1 = originalBar ;
515                 b2 = b ;
516             } else {
517                 b1 = b ;
518                 b2 = originalBar ;
519             }
520
521             this.x = b1.x;
522             this.y = b1.y;
523
524             if (b1.splits.length === 0) {
525                 this.addSplit(0, 0, b1.w, b1.h, b1.color) ;
526             }
527             if (b2.splits.length === 0) {
528                 this.addSplit(0, b1.h, b2.w, b2.h, b2.color ) ;
529             }

```

```

530
531         if( b1.splits.length > 0 ) {
532             for(i = 0; i < b1.splits.length; i++ ) {
533                 this.addSplit(b1.splits[i].x, b1.splits[i].y, b1.splits[i].w,
534                     ↪ b1.splits[i].h, b1.splits[i].color ) ;
535             }
536         if( b2.splits.length > 0 ) {
537             for(i = 0; i < b2.splits.length; i++ ) {
538                 this.addSplit(b2.splits[i].x, b2.splits[i].y + b1.h, b2.splits[i].w,
539                     ↪ b2.splits[i].h, b2.splits[i].color ) ;
540             }
541         }
542     } else {
543         // alert("Joining along height");
544         if( originalBar.x < b.x ) {
545             b1 = originalBar ;
546             b2 = b ;
547         } else {
548             b1 = b ;
549             b2 = originalBar ;
550         }
551
552         this.x = b1.x;
553         this.y = b1.y;
554
555         if (b1.splits.length === 0) {
556             this.addSplit(0, 0, b1.w, b1.h, b1.color) ;
557         }
558         // this.addSplit(0, b1.h, originalBar.w, originalBar.h, b2.c) ;
559         // this.addSplit(0, b1.h, b2.w, b2.h, b2.c) ;
560         if (b2.splits.length === 0) {
561             this.addSplit(b1.w, 0, b2.w, b2.h, b2.color) ;
562         }
563
564         if( b1.splits.length > 0 ) {
565             for(i = 0; i < b1.splits.length; i++ ) {
566                 this.addSplit(b1.splits[i].x, b1.splits[i].y, b1.splits[i].w,
567                     ↪ b1.splits[i].h, b1.splits[i].color ) ;
568             }
569         if( b2.splits.length > 0 ) {
570             for(i = 0; i < b2.splits.length; i++ ) {
571                 this.addSplit(b2.splits[i].x + b1.w, b2.splits[i].y, b2.splits[i].w,
572                     ↪ b2.splits[i].h, b2.splits[i].color ) ;
573             }
574         }
575     }
576 }
577
578 // this.purgeOverlappingSplits() ;
579
580 this.clearMeasurement() ;
581
582 return(true);
583 };
584
585
586 Bar.prototype.nearestEdge = function(p) {
587     // return a string indicating which edge is the closest one to the given point (p)
588     var closestEdge = 'bottom' ;
589     var dl = p.x - this.x ;
590     var dr = this.w - dl ;
591     var dt = p.y - this.y ;
592     var db = this.h - dt ;
593
594     if (dl <= dr && dl <= dt && dl <= db ) {
595         closestEdge = "left" ;
596     } else if ( dr <= dl && dr <= dt && dr <= db ) {
597         closestEdge = "right" ;
598     } else if ( dt <= dl && dt <= dr && dt <= db ) {
599         closestEdge = "top" ;
600     }

```

```

601         return closestEdge ;
602     };
603 };
604
605 Bar.prototype.toggleSelection = function() {};
606
607 Bar.prototype.setRepeatUnit = function() {
608     this.repeatUnit = this.makeCopy(true);
609     this.repeatUnit.unPastel();
610 };
611
612 Bar.prototype.unPastel = function() {
613
614 }
615
616 // static methods
617
618 Bar.create = function(x, y, w, h, type, color) {
619     var b = new Bar() ;
620     b.x = x ;
621     b.y = y ;
622     b.w = w ;
623     b.h = h ;
624     b.size = w * h ;
625     b.color = color ;
626     b.type = type ;
627     return b ;
628 };
629
630 Bar.createFromMouse = function(p1, p2, type, color) {
631     var w = Math.abs(p2.x - p1.x) ;
632     var h = Math.abs(p2.y - p1.y) ;
633     var p = Point.min( p1, p2 ) ;
634     var b = Bar.create(p.x, p.y, w, h, type, color) ;
635     return b ;
636 };
637
638
639 Bar.createFromSplit = function(s, inx, iny) {
640     var b = Bar.create(inx+s.x+10, iny+s.y+10, s.w, s.h, this.type, s.color) ;
641     return b ;
642 };
643
644 Bar.distanceBetween = function(b1, b2) {
645     // Returns the distance vertically and horizontally between the centers
646     // of two bars.
647     // Given as separate dimensions in a Point object, think of the return value as the amount needed
648     // to translate b1 so that the center would be precisely over b2.
649     var p = new Point() ;
650     // var totalDistance = Math.max(b1.x + b1.w, b2.x + b2.w) - Math.min(b1.x, b2.x) ;
651     // p.x = totalDistance - b1.w - b2.w ;
652     // totalDistance = Math.max(b1.y + b1.h, b2.y + b2.h) - Math.min(b1.y, b2.y) ;
653     // p.y = totalDistance - b1.h - b2.h ;
654     p.x = b2.x - b1.x;
655     p.y = b2.y - b1.y;
656     return p ;
657 };
658
659
660 Bar.copyFromJSON = function(JSON_Bar) {
661     var b = new Bar() ;
662
663
664     b.x = JSON_Bar.x ;
665     b.y = JSON_Bar.y ;
666     b.w = JSON_Bar.w ;
667     b.h = JSON_Bar.h ;
668     b.size = JSON_Bar.size ;
669     b.color = JSON_Bar.color ;
670     b.makeSplitsFromJSON(JSON_Bar.splits) ;
671     b.label = JSON_Bar.label ;
672     b.isUnitBar = JSON_Bar.isUnitBar ;
673     b.fraction = JSON_Bar.fraction ;
674     b.type = JSON_Bar.type ;
675     b.isSelected = false ;

```

```

676
677     return b ;
678 };
679
680 Bar.prototype.makeSplitsFromJSON = function(JSON_splits) {
681
682     this.clearSplits();
683     for (var i = 0; i < JSON_splits.length; i++) {
684         this.addSplit(JSON_splits[i].x,JSON_splits[i].y,JSON_splits[i].w,JSON_splits[i].h,JSON_sp
        ↪ lits[i].color);
685     }
686 };
687
688

```

2.2.2 Mat.js

```

1  // Copyright University of Massachusetts Dartmouth 2013
2  //
3  // Designed and built by James P. Burke and Jason Orrill
4  // Modified and developed by Hakan Sandir
5  //
6  // This Javascript version of Fraction Bars is based on
7  // the Transparent Media desktop version of Fraction Bars,
8  // which in turn was based on the original TIMA Bars software
9  // by John Olive and Leslie Steffe.
10 // We thank them for allowing us to update that product.
11
12
13
14 function Mat() {
15     //TODO: convert x, y to a Point?
16     this.x = null ;
17     this.y = null ;
18     this.w = null ;
19     this.h = null ;
20     this.size = null ;
21     this.color = null ;
22     //     this.splits = [] ;
23     //     this.label = '' ;
24     //     this.fraction = '' ;
25     this.type = null ;
26     this.isSelected = false ;
27 }
28
29
30
31 Mat.prototype.copy = function(with_offset) {
32     var offset = 10 ;
33     var b = new Bar() ;
34
35     if (with_offset === false) {
36         offset = 0;
37     }
38
39     b.x = this.x + offset ;
40     b.y = this.y + offset ;
41     b.w = this.w ;
42     b.h = this.h ;
43     b.size = this.size ;
44     b.color = this.color ;
45     //     b.splits = this.copySplits() ;
46     //     b.label = this.label ;
47     //     b.isUnitBar = false ;
48     //     b.fraction = this.fraction ;
49     b.type = this.type ;
50     b.isSelected = true ;
51
52     return b ;
53 };
54
55

```

```

56 Mat.prototype.nearestEdge = function(p) {
57     // return a string indicating which edge is the closest one to the given point (p)
58     var closestEdge = 'bottom' ;
59     var dl = p.x - this.x ;
60     var dr = this.w - dl ;
61     var dt = p.y - this.y ;
62     var db = this.h - dt ;
63
64     if (dl <= dr && dl <= dt && dl <= db ) {
65         closestEdge = "left" ;
66     } else if ( dr <= dl && dr <= dt && dr <= db ) {
67         closestEdge = "right" ;
68     } else if ( dt <= dl && dt <= dr && dt <= db ) {
69         closestEdge = "top" ;
70     }
71     return closestEdge ;
72 };
73
74 Mat.prototype.toggleSelection = function() {};
75
76 // static methods
77
78 Mat.create = function(x, y, w, h, type, color) {
79     var b = new Mat() ;
80     b.x = x ;
81     b.y = y ;
82     b.w = w ;
83     b.h = h ;
84     b.size = w * h ;
85     b.color = color ;
86     b.type = type ;
87     return b ;
88 };
89
90 Mat.createFromMouse = function(p1, p2, type, color) {
91     var w = Math.abs(p2.x - p1.x) ;
92     var h = Math.abs(p2.y - p1.y) ;
93     var p = Point.min( p1, p2 ) ;
94     var b = Mat.create(p.x, p.y, w, h, type, color) ;
95     return b ;
96 };
97
98 Mat.distanceBetween = function(b1, b2) {
99     var p = new Point() ;
100     var totalDistance = Math.max(b1.x + b1.w, b2.x + b2.w) - Math.min(b1.x, b2.x) ;
101     p.x = totalDistance - b1.w - b2.w ;
102     totalDistance = Math.max(b1.y + b1.h, b2.y + b2.h) - Math.min(b1.y, b2.y) ;
103     p.y = totalDistance - b1.h - b2.h ;
104     return p ;
105 };
106
107 Mat.copyFromJSON = function(JSON_Mat) {
108     var b = new Mat() ;
109
110
111     b.x = JSON_Mat.x ;
112     b.y = JSON_Mat.y ;
113     b.w = JSON_Mat.w ;
114     b.h = JSON_Mat.h ;
115     b.size = JSON_Mat.size ;
116     b.color = JSON_Mat.color ;
117
118     b.type = JSON_Mat.type ;
119     b.isSelected = false ;
120
121     return b ;
122 };

```

2.2.3 Point.js

```

1 // Copyright University of Massachusetts Dartmouth 2013
2 //
3 // Designed and built by James P. Burke and Jason Orrill

```

```

4 // Modified and developed by Hakan Sandir
5 //
6 // This Javascript version of Fraction Bars is based on
7 // the Transparent Media desktop version of Fraction Bars,
8 // which in turn was based on the original TIMA Bars software
9 // by John Olive and Leslie Steffe.
10 // We thank them for allowing us to update that product.
11
12
13
14 function Point() {
15     this.x = null ;
16     this.y = null ;
17 }
18
19 Point.prototype.equals = function(p) {
20     var _output = false ;
21     if( p ) {
22         _output = (p.x == this.x && p.y == this.y) ;
23     }
24     return _output ;
25 };
26
27 Point.prototype.isOnLine = function(line) {
28     var onLine ;
29     if (line.x1 == line.x2) {
30         isOnLine = (this.x == line.x1) && (this.y >= Math.min(line.y1, line.y2)) && (this.y <=
            ↪ Math.max(line.y1, line.y2)) ;
31     } else {
32         isOnLine = (this.y == line.y1) && (this.x >= Math.min(line.x1, line.x2)) && (this.x <=
            ↪ Math.max(line.x1, line.x2)) ;
33     }
34     return isOnLine ;
35 };
36
37 // static methods
38
39 Point.createFromMouseEvent = function(e, elem) {
40     var p = new Point();
41     // Support both mouse and normalized touch events
42     var x = (typeof e.clientX !== 'undefined') ? e.clientX : (e.touches && e.touches[0] ?
        ↪ e.touches[0].clientX : 0);
43     var y = (typeof e.clientY !== 'undefined') ? e.clientY : (e.touches && e.touches[0] ?
        ↪ e.touches[0].clientY : 0);
44     p.x = Math.round((x - elem.position().left) + window.pageXOffset);
45     p.y = Math.round((y - elem.position().top) + window.pageYOffset);
46     return p;
47 }
48
49 Point.subtract = function(p1, p2) {
50     var p = new Point() ;
51     p.x = p1.x - p2.x ;
52     p.y = p1.y - p2.y ;
53     return p ;
54 }
55
56 Point.add = function(p1, p2) {
57     var p = new Point() ;
58     p.x = p1.x + p2.x ;
59     p.y = p1.y + p2.y ;
60     return p ;
61 }
62 }
63
64 Point.multiply = function(p1, p2) {
65     var p = new Point() ;
66     p.x = p1.x * p2.x ;
67     p.y = p1.y * p2.y ;
68     return p ;
69 }
70
71 Point.min = function( p1, p2 ) {
72     var p = new Point() ;
73     p.x = Math.min(p1.x, p2.x) ;
74     p.y = Math.min(p1.y, p2.y) ;

```



```

75         return p ;
76     }

```

2.2.4 Line.js

```

1  // Copyright University of Massachusetts Dartmouth 2013
2  //
3  // Designed and built by James P. Burke and Jason Orrill
4  // Modified and developed by Hakan Sandir
5  //
6  // This Javascript version of Fraction Bars is based on
7  // the Transparent Media desktop version of Fraction Bars,
8  // which in turn was based on the original TIMA Bars software
9  // by John Olive and Leslie Steffe.
10 // We thank them for allowing us to update that product.
11
12
13 function Line(x1, y1, x2, y2) {
14     this.x1 = x1 ;
15     this.y1 = y1 ;
16     this.x2 = x2 ;
17     this.y2 = y2 ;
18 }
19
20 Line.prototype.equals = function(line) {
21     var _output ;
22     if( line ) {
23         _output = (this.x1 == line.x1 && this.y1 == line.y1 && this.x2 == line.x2 && this.y2 ==
                ↪ line.y2) ;
24     }
25     return _output ;
26 }

```

2.2.5 Split.js

```

1  // Copyright University of Massachusetts Dartmouth 2013
2  //
3  // Designed and built by James P. Burke and Jason Orrill
4  // Modified and developed by Hakan Sandir
5  //
6  // This Javascript version of Fraction Bars is based on
7  // the Transparent Media desktop version of Fraction Bars,
8  // which in turn was based on the original TIMA Bars software
9  // by John Olive and Leslie Steffe.
10 // We thank them for allowing us to update that product.
11
12
13 function Split(x, y, w, h, c) {
14     this.x = x ;
15     this.y = y ;
16     this.w = w ;
17     this.h = h ;
18     this.color = c ;
19     this.isSelected = false;
20 }
21
22 Split.prototype.equals = function(s) {
23     var _output = false ;
24     if( s ) {
25         _output = (s.x == this.x && s.y == this.y && s.w == this.w && s.h == this.h) ;
26     }
27     return _output ;
28 };
29
30 Split.prototype.copy = function() {
31     newsplit = new Split(this.x, this.y, this.w, this.h, this.color);
32     return newsplit;
33 };

```

2.2.6 Blob.js

```
1  /* Blob.js
2  * A Blob implementation.
3  * 2013-06-20
4  *
5  * By Eli Grey, http://eligrey.com
6  * By Devin Samarin, https://github.com/eboyjr
7  * License: X11/MIT
8  * See LICENSE.md
9  */
10
11 /*global self, unescape */
12 /*jslint bitwise: true, regexp: true, confusion: true, es5: true, vars: true, white: true,
13    plusplus: true */
14
15 /*! @source http://purl.eligrey.com/github/Blob.js/blob/master/Blob.js */
16
17 if (typeof Blob !== "function" || typeof URL === "undefined")
18 if (typeof Blob === "function" && typeof webkitURL !== "undefined") self.URL = webkitURL;
19 else var Blob = (function (view) {
20     "use strict";
21
22     var BlobBuilder = view.BlobBuilder || view.WebKitBlobBuilder || view.MozBlobBuilder ||
23         ↪ view.MSBlobBuilder || (function(view) {
24         var
25             get_class = function(object) {
26                 return
27                 ↪ Object.prototype.toString.call(object).match(/\[object\s(.*)\]$/)[1];
28             },
29             FakeBlobBuilder = function BlobBuilder() {
30                 this.data = [];
31             },
32             FakeBlob = function Blob(data, type, encoding) {
33                 this.data = data;
34                 this.size = data.length;
35                 this.type = type;
36                 this.encoding = encoding;
37             },
38             FBB_proto = FakeBlobBuilder.prototype,
39             FB_proto = FakeBlob.prototype,
40             FileReaderSync = view.FileReaderSync,
41             FileException = function(type) {
42                 this.code = this[this.name = type];
43             },
44             file_ex_codes = (
45                 "NOT_FOUND_ERR SECURITY_ERR ABORT_ERR NOT_READABLE_ERR ENCODING_ERR "
46                 + "NO_MODIFICATION_ALLOWED_ERR INVALID_STATE_ERR SYNTAX_ERR"
47             ).split(" "),
48             file_ex_code = file_ex_codes.length,
49             real_URL = view.URL || view.webkitURL || view,
50             real_create_object_URL = real_URL.createObjectURL,
51             real_revoke_object_URL = real_URL.revokeObjectURL,
52             URL = real_URL,
53             btoa = view.btoa,
54             atob = view.atob,
55
56             , ArrayBuffer = view.ArrayBuffer
57             , Uint8Array = view.Uint8Array
58
59 ;
60 FakeBlob.fake = FB_proto.fake = true;
61 while (file_ex_code--) {
62     FileException.prototype[file_ex_codes[file_ex_code]] = file_ex_code + 1;
63 }
64 if (!real_URL.createObjectURL) {
65     URL = view.URL = {};
66 }
67 URL.createObjectURL = function(blob) {
68     var
69         type = blob.type
70         , data_URI_header
71 ;
72 if (type === null) {
73     type = "application/octet-stream";
74 }
```

```

71     }
72     if (blob instanceof FakeBlob) {
73         data_URI_header = "data:" + type;
74         if (blob.encoding === "base64") {
75             return data_URI_header + ";base64," + blob.data;
76         } else if (blob.encoding === "URI") {
77             return data_URI_header + "," + decodeURIComponent(blob.data);
78         } if (btoa) {
79             return data_URI_header + ";base64," + btoa(blob.data);
80         } else {
81             return data_URI_header + "," + encodeURIComponent(blob.data);
82         }
83     } else if (real_create_object_URL) {
84         return real_create_object_URL.call(real_URL, blob);
85     }
86 };
87 URL.revokeObjectURL = function(object_URL) {
88     if (object_URL.substring(0, 5) !== "data:" && real_revoke_object_URL) {
89         real_revoke_object_URL.call(real_URL, object_URL);
90     }
91 };
92 FBB_proto.append = function(data/*, endings*/) {
93     var bb = this.data;
94     // decode data to a binary string
95     if (Uint8Array && (data instanceof ArrayBuffer || data instanceof Uint8Array)) {
96         var
97             str = ""
98             , buf = new Uint8Array(data)
99             , i = 0
100             , buf_len = buf.length
101         ;
102         for (; i < buf_len; i++) {
103             str += String.fromCharCode(buf[i]);
104         }
105         bb.push(str);
106     } else if (get_class(data) === "Blob" || get_class(data) === "File") {
107         if (FileReaderSync) {
108             var fr = new FileReaderSync;
109             bb.push(fr.readAsBinaryString(data));
110         } else {
111             // async FileReader won't work as BlobBuilder is sync
112             throw new FileException("NOT_READABLE_ERR");
113         }
114     } else if (data instanceof FakeBlob) {
115         if (data.encoding === "base64" && atob) {
116             bb.push(atob(data.data));
117         } else if (data.encoding === "URI") {
118             bb.push(decodeURIComponent(data.data));
119         } else if (data.encoding === "raw") {
120             bb.push(data.data);
121         }
122     } else {
123         if (typeof data !== "string") {
124             data += ""; // convert unsupported types to strings
125         }
126         // decode UTF-16 to binary string
127         bb.push(unescape(encodeURIComponent(data)));
128     }
129 };
130 FBB_proto.getBlob = function(type) {
131     if (!arguments.length) {
132         type = null;
133     }
134     return new FakeBlob(this.data.join(""), type, "raw");
135 };
136 FBB_proto.toString = function() {
137     return "[object BlobBuilder]";
138 };
139 FBB_proto.slice = function(start, end, type) {
140     var args = arguments.length;
141     if (args < 3) {
142         type = null;
143     }
144     return new FakeBlob(
145         this.data.slice(start, args > 1 ? end : this.data.length)

```

```

146         , type
147         , this.encoding
148     );
149 };
150 FB_proto.toString = function() {
151     return "[object Blob]";
152 };
153 return FakeBlobBuilder;
154 }(view));
155
156 return function Blob(blobParts, options) {
157     var type = options ? (options.type || "") : "";
158     var builder = new BlobBuilder();
159     if (blobParts) {
160         for (var i = 0, len = blobParts.length; i < len; i++) {
161             builder.append(blobParts[i]);
162         }
163     }
164     return builder.getBlob(type);
165 };
166 }(self));

```

2.3 UI Components

2.3.1 CanvasState.js

```

1 // A CanvasState object is used to contain a copy of the hierarchy of all the contents of the
2 // FractionBars Canvas. Essentially, it is kind of an epty husk of a canvas with just the bars,
3 // mats, and minimum required noformation for performing an "undo" or a "redo"
4
5 // Copyright University of Massachusetts Dartmouth 2013
6 //
7 // Designed and built by James P. Burke and Jason Orrill
8 // Modified and developed by Hakan Sandir
9 //
10 // This Javascript version of Fraction Bars is based on
11 // the Transparent Media desktop version of Fraction Bars,
12 // which in turn was based on the original TIMA Bars software
13 // by John Olive and Leslie Steffe.
14 // We thank them for allowing us to update that product.
15
16
17
18 function CanvasState(FBCanvas) {
19     this.mFBCanvas = FBCanvas ;
20     this.canvasState = null ;
21
22     this.mBars = [] ;
23     this.mMats = [] ;
24     this.mUnitBar = null ;
25     this.mHidden= [] ;
26 }
27
28
29
30 // Also copy mats
31 CanvasState.prototype.grabBarsAndMats = function() {
32
33     var mBars = [] ;
34     var mMats = [] ;
35     var aBar = null;
36     var mHidden=[];
37
38     for( var i = 0; i < this.mFBCanvas.bars.length; i++ ) {
39         aBar = this.mFBCanvas.bars[i].copy(false);
40         this.mBars.push( aBar ) ;
41         if (this.mFBCanvas.bars[i] === this.mFBCanvas.unitBar) {
42             this.mUnitBar = aBar; // Remember which copy is a copy of the unit bar, if any.
43         }
44         if (this.mFBCanvas.bars[i].isSelected) {
45             aBar.isSelected = true;
46         } else {

```

```

47         aBar.isSelected = false;
48     }
49     if (this.mFBCanvas.bars[i].isUnitBar) {
50         aBar.isUnitBar = true;
51     }
52     //         aBar.clearSplitSelection();
53 }
54
55 for(var j = 0; j < this.mFBCanvas.mats.length; j++ ) {
56     this.mMats.push( this.mFBCanvas.mats[j].copy(false) );
57 }
58 this.mHidden=hiddenButtonsName.slice(0);
59 if (hiddenButtonsName.indexOf("tool_hide")<0) {
60     this.mHidden.push('tool_hide') ;
61 }
62 if (hiddenButtonsName.indexOf("action_show")<0) {
63     this.mHidden.push('action_show') ;
64 }
65
66 };
67
68

```

2.3.2 SplitsWidget.js

```

1  // Copyright University of Massachusetts Dartmouth 2013
2  //
3  // Designed and built by James P. Burke and Jason Orrill
4  // Modified and developed by Hakan Sandir
5  //
6  // This Javascript version of Fraction Bars is based on
7  // the Transparent Media desktop version of Fraction Bars,
8  // which in turn was based on the original TIMA Bars software
9  // by John Olive and Leslie Steffe.
10 // We thank them for allowing us to update that product.
11
12
13 function SplitsWidget(canvasContext) {
14     this.context = canvasContext ;
15     this.images = [];
16     this.vertical = true;
17     this.num_splits = 2;
18     this.color = "yellow";
19 }
20
21
22
23
24 SplitsWidget.prototype.handleSliderChange = function(event, ui) {
25     // var aslider = event.target;
26     // this.num_splits = aslider.slider( "value" );
27     // alert(this.num_splits);
28
29
30     //     this.num_splits = $("#split-slider").slider("value");
31     //     this.num_splits = $("#split-slider-field").val();
32
33     this.num_splits = ui.value;
34     this.refreshCanvas();
35
36 };
37
38
39 SplitsWidget.prototype.handleVertHorizChange = function(event) {
40
41
42     var the_checked = $("input:checked").val();
43     if (the_checked == "Vertical") {
44         this.vertical = true;
45     } else {
46         if(Utilities.flag[1]){
47
48             this.vertical = false;}

```

```

49         else {
50             this.vertical = true;}
51     }
52     this.refreshCanvas();
53 };
54
55 SplitsWidget.prototype.refreshCanvas = function() {
56
57     this.context.strokeStyle = "#FF3333";
58     this.context.fillStyle = this.color;
59
60     //     this.num_splits = $("#split-slider").slider("value");
61     //     this.num_splits = $("#split-slider-field").val();
62     //     this.num_splits = document.getElementById("split-slider").slider();
63
64     var width = $("#split-display").attr("width");
65     var height = $("#split-display").attr("height");
66
67     this.context.fillRect(0,0,width,height);
68
69     //     this.context.strokeText(this.num_splits,10,10);
70
71     if (this.vertical) {
72         width = width / this.num_splits;
73         for (var i = 0; i < this.num_splits; i++) {
74             this.context.strokeRect(i*width,0,width,height);
75         }
76     } else {
77         height = height / this.num_splits;
78         for (var j = 0; j < this.num_splits; j++) {
79             this.context.strokeRect(0,j*height,width,height);
80         }
81     }
82
83     this.refreshed = true;
84 };

```

2.4 Utility Functions

2.4.1 utilities.js

```

1  // Copyright University of Massachusetts Dartmouth 2013
2  //
3  // Designed and built by James P. Burke and Jason Orrill
4  // Modified and developed by Hakan Sandir
5  //
6  // This Javascript version of Fraction Bars is based on
7  // the Transparent Media desktop version of Fraction Bars,
8  // which in turn was based on the original TIMA Bars software
9  // by John Olive and Leslie Steffe.
10 // We thank them for allowing us to update that product.
11
12
13
14 function Utilities() {
15     this.shiftKeyDown = false ;
16     this.ctrlKeyDown = false ;
17 }
18
19 //First attempt
20 Utilities.file_list="";
21 Utilities.file_index=0;
22 //
23
24 Utilities.flag=['it,sp,rpt,lng'];
25 Utilities.flag[0]=false;
26 Utilities.flag[1]=false;
27 Utilities.flag[2]=false;
28 Utilities.flag[3]=false;
29 Utilities.USE_CURRENT_SELECTION = 'useCurrent' ;
30 Utilities.USE_LAST_SELECTION = 'useLast' ;
31

```

```

32
33 Utilities.include_js = function (file,path) {
34     if (typeof(path) !== 'undefined' && path !== null) {
35         file = path + file ;
36     }
37     var include_file = document.createElement('script');
38     include_file.type = 'text/javascript';
39     include_file.src = file;
40     document.getElementsByTagName('head')[0].appendChild(include_file);
41 };
42
43 Utilities.createFraction = function(numerator, denominator) {
44     // Calculate the (approximate) fraction for this measurement.
45     // Basic algorigm taken from Dr. Math at the Math Forum...
46     // http://mathforum.org/library/drmath/view/51910.html
47
48     var max_terms = 30 ;
49     var min_divisor = 0.000001 ;
50     var max_error = 0.00001 ;
51
52     var v = numerator / denominator ;
53     var f = v ;
54
55     var n1 = 1 ;
56     var d1 = 0 ;
57     var n2 = 0 ;
58     var d2 = 1 ;
59
60     var a ;
61
62     for (i = 0; i < max_terms; i++) {
63         a = Math.round(f) ;
64         f = f - a ;
65         n = n1 * a + n2 ;
66         d = d1 * a + d2 ;
67
68         n2 = n1 ;
69         d2 = d1 ;
70
71         n1 = n ;
72         d1 = d ;
73
74         if (f < min_divisor && Math.abs(v-n/d) < max_error) {
75             break ;
76         }
77
78         f = 1/f ;
79     }
80
81     if (Math.floor(v) == v) {
82         return v ;
83     }
84     else{
85         return Math.abs(n) + "/" + Math.abs(d) ;
86     }
87 };
88
89 Utilities.log = function(msg) {
90     if( window.console ) {
91         console.log( msg ) ;
92     }
93 };
94
95 Utilities.colorLuminance = function(hex, lum) {
96
97     // validate hex string
98     hex = String(hex).replace(/[^0-9a-f]/gi, '');
99     if (hex.length < 6) {
100         hex = hex[0]+hex[0]+hex[1]+hex[1]+hex[2]+hex[2];
101     }
102     lum = lum || 0;
103
104     // convert to decimal and change luminosity
105     var rgb = "#", c, i;
106     for (i = 0; i < 3; i++) {

```

```

107     c = parseInt(hex.substr(i*2,2), 16);
108     c = Math.round(Math.min(Math.max(0, c + (c * lum)), 255)).toString(16);
109     rgb += ("00"+c).substr(c.length);
110 }
111
112 return rgb;
113 };
114
115 Utilities.getMarkedIterateFlag = function() {
116     // Returns false by default
117     state = ($('#marked-iterate').attr('data-flag') === "true");
118     return state;
119 };

```

2.4.2 cycle.js

```

1  /*
2     cycle.js
3     2013-02-19
4
5     Public Domain.
6
7     NO WARRANTY EXPRESSED OR IMPLIED. USE AT YOUR OWN RISK.
8
9     This code should be minified before deployment.
10    See http://javascript.crockford.com/jsmin.html
11
12    USE YOUR OWN COPY. IT IS EXTREMELY UNWISE TO LOAD CODE FROM SERVERS YOU DO
13    NOT CONTROL.
14 */
15
16 /*jshint evil: true, regexp: true */
17
18 /*members $ref, apply, call, decycle, hasOwnProperty, length, prototype, push,
19    retrocycle, stringify, test, toString
20 */
21
22 if (typeof JSON.decycle !== 'function') {
23     JSON.decycle = function decycle(object) {
24         'use strict';
25
26         // Make a deep copy of an object or array, assuring that there is at most
27         // one instance of each object or array in the resulting structure. The
28         // duplicate references (which might be forming cycles) are replaced with
29         // an object of the form
30         //     {$ref: PATH}
31         // where the PATH is a JSONPath string that locates the first occurrence.
32         // So,
33         //     var a = [];
34         //     a[0] = a;
35         //     return JSON.stringify(JSON.decycle(a));
36         // produces the string '[{"$ref": "$"}]'.
37
38         // JSONPath is used to locate the unique object. $ indicates the top level of
39         // the object or array. [NUMBER] or [STRING] indicates a child member or
40         // property.
41
42         var objects = [], // Keep a reference to each unique object or array
43             paths = [];   // Keep the path to each unique object or array
44
45         return (function derez(value, path) {
46             // The derez recurses through the object, producing the deep copy.
47
48             var i,        // The loop counter
49                 name,    // Property name
50                 nu;      // The new object or array
51
52             // typeof null === 'object', so go on if this value is really an object but not
53             // one of the weird builtin objects.
54
55             if (typeof value === 'object' && value !== null &&
56                 !(value instanceof Boolean) &&

```



```

58         !(value instanceof Date)    &&
59         !(value instanceof Number)  &&
60         !(value instanceof RegExp)  &&
61         !(value instanceof String)) {
62
63     // If the value is an object or array, look to see if we have already
64     // encountered it. If so, return a $ref/path object. This is a hard way,
65     // linear search that will get slower as the number of unique objects grows.
66
67     for (i = 0; i < objects.length; i += 1) {
68         if (objects[i] === value) {
69             return {$ref: paths[i]};
70         }
71     }
72
73     // Otherwise, accumulate the unique value and its path.
74
75     objects.push(value);
76     paths.push(path);
77
78     // If it is an array, replicate the array.
79
80     if (Object.prototype.toString.apply(value) === '[object Array]') {
81         nu = [];
82         for (i = 0; i < value.length; i += 1) {
83             nu[i] = derez(value[i], path + '[' + i + ']');
84         }
85     } else {
86
87     // If it is an object, replicate the object.
88
89         nu = {};
90         for (name in value) {
91             if (Object.prototype.hasOwnProperty.call(value, name)) {
92                 nu[name] = derez(value[name],
93                     path + '[' + JSON.stringify(name) + ']');
94             }
95         }
96     }
97     return nu;
98 }
99 return value;
100 }(object, '$'));
101 };
102 }
103
104
105 if (typeof JSON.retrocycle !== 'function') {
106     JSON.retrocycle = function retrocycle($) {
107         'use strict';
108
109         // Restore an object that was reduced by decycle. Members whose values are
110         // objects of the form
111         //     {$ref: PATH}
112         // are replaced with references to the value found by the PATH. This will
113         // restore cycles. The object will be mutated.
114
115         // The eval function is used to locate the values described by a PATH. The
116         // root object is kept in a $ variable. A regular expression is used to
117         // assure that the PATH is extremely well formed. The regexp contains nested
118         // * quantifiers. That has been known to have extremely bad performance
119         // problems on some browsers for very long strings. A PATH is expected to be
120         // reasonably short. A PATH is allowed to belong to a very restricted subset of
121         // Goessner's JSONPath.
122
123         // So,
124         //     var s = '[$ref:"$"]';
125         //     return JSON.retrocycle(JSON.parse(s));
126         // produces an array containing a single element which is the array itself.
127
128         var px =
129             /\$(?:\[?(?:\d+|\"(?:\\\"|u0000-u001f|\\(?:[\\\/bfnrt]|u[0-9a-zA-Z]{4}))*)\])?$/;
130
131         (function rez(value) {

```

```

133 // The rez function walks recursively through the object looking for $ref
134 // properties. When it finds one that has a value that is a path, then it
135 // replaces the $ref object with a reference to the value that is found by
136 // the path.
137
138     var i, item, name, path;
139
140     if (value && typeof value === 'object') {
141         if (Object.prototype.toString.apply(value) === '[Object Array]') {
142             for (i = 0; i < value.length; i += 1) {
143                 item = value[i];
144                 if (item && typeof item === 'object') {
145                     path = item.$ref;
146                     if (typeof path === 'string' && px.test(path)) {
147                         value[i] = eval(path);
148                     } else {
149                         rez(item);
150                     }
151                 }
152             }
153         } else {
154             for (name in value) {
155                 if (typeof value[name] === 'object') {
156                     item = value[name];
157                     if (item) {
158                         path = item.$ref;
159                         if (typeof path === 'string' && px.test(path)) {
160                             value[name] = eval(path);
161                         } else {
162                             rez(item);
163                         }
164                     }
165                 }
166             }
167         }
168     }
169     }($);
170     return $;
171 };
172 }

```

3 CSS Files

3.1 Main Styles

3.1.1 fractionBars.css

```

1
2 /*
3 // Copyright University of Massachusetts Dartmouth 2013
4 //
5 // Designed and built by James P. Burke and Jason Orrill
6 //
7 // This Javascript version of Fraction Bars is based on
8 // the Transparent Media desktop version of Fraction Bars,
9 // which in turn was based on the original TIMA Bars software
10 // by John Olive and Leslie Steffe.
11 // We thank them for allowing us to update that product.
12 */
13
14
15 body, p, td {
16     font-family: Helvetica, Arial, sans-serif ;
17     font-size: 12px ;
18 }
19 a {
20     cursor: pointer;
21 }
22
23 .skip-link {display:none;}
24

```

```

25 #labelInput {
26     display: none ;
27 }
28
29 #fbCanvas {
30     background-color: #EFEFEF;
31     border: 1px gray groove;
32 }
33 #tools {
34     float:left ;
35     width: 100px ;
36 }
37 #tools .toolGroup {
38     margin-bottom: 12px ;
39 }
40 #tools .toolGroup a {
41     border: 1px solid #999999 ;
42     display: block ;
43     width: 80px ;
44     text-align: center ;
45     margin: 4px ;
46     padding: 2px ;
47 }
48
49 #tools .toolGroup a.colorBlock {
50     float: left;
51     border: 1px solid gray;
52     width: 10px;
53     height: 10px;
54     display: inline;
55     margin: 3px 3px 4px 4px;
56 }
57
58 #tools .toolGroup a.colorBlock1 {
59     float: left;
60     border: 1px solid gray;
61     width: 10px;
62     height: 10px;
63     display: inline;
64     margin: 3px 3px 4px 4px;
65 }
66
67 #tools .toolGroup a.hideShow {
68     float: left;
69     border: 1px solid gray;
70     width: 35px;
71     display: inline;
72     margin: 3px 3px 4px 4px;
73 }
74
75 #tools .toolGroup a.colorSelected {
76     border: 1px solid black ;
77 }
78
79 #tools .toolGroup a.toolSelected {
80     border: 1px solid black ;
81     background-color: #EEEEEE ;
82 }
83
84 #split-display {
85     float: right;
86     background-color: #EFEFEF;
87     border: 1px gray groove;
88 }
89
90
91 #split-slider {
92     width: 250px;
93 }
94
95 .color10 {background-color: #FFFF66;}
96 .color3 {background-color: #ACBEFF;}
97 .color7 {background-color: #E6E6E6;}
98 .color5 {background-color: #FFFFFF;}
99 .color12 {background-color: #CCFF66;}

```

```

100 .color9 {background-color: #FFCC66;}
101 .color13 {background-color: #DD99FF;}
102 .color14 {background-color: #FF92DA;}
103 .color1 {background-color: #A8FFF4;}
104 .color6 {background-color: #EFEFEF;}
105 .color2 {background-color: #DDFFF0;}
106 .color4 {background-color: #707EFF;}
107 .color8 {background-color: #D6D6D6;}
108 .color16 {background-color: #FF8C8C;}
109 .color11 {background-color: #E9FF66;}
110 .color15 {background-color: #F56ED0;}

```

3.1.2 deneme.css

```

1 fieldset {
2     border: 2px #aa3333 ;
3     border-radius: 10px;
4 }
5
6 /* Editable [pseudo]select (i.e. fieldsets with [class=elist]) */
7
8 fieldset.elist {
9     display: block;
10    position: relative;
11    vertical-align: bottom;
12    overflow: visible;
13    width: 200px;
14    padding: 0;
15    margin: 0;
16    border: none;
17 }
18
19 fieldset.elist ul {
20     position: absolute;
21     width: 100%;
22     max-height: 320px;
23     padding: 0;
24     margin: 0;
25     overflow: hidden;
26     background-color: #ffffff;
27 }
28
29 fieldset.elist:hover ul {
30     background-color: #ffffff;
31     border: 2px #aaaaaa solid;
32     left: 2px;
33     overflow: auto;
34 }
35
36 fieldset.elist ul > li {
37     list-style-type: none;
38     background-color: transparent;
39 }
40
41 fieldset.elist label {
42     display: none;
43     width: 100%;
44 }
45
46 fieldset.elist ul input[type="radio"] {
47     display: none;
48 }
49
50
51 fieldset.elist input[type="radio"]:checked ~ label {
52     display: block;
53     width: 292px;
54     background-color: #ffffff;
55 }
56
57
58 fieldset.elist input[type="radio"]:checked ~ label:after {
59     content: " \2335";

```

```

60 }
61
62 fieldset.elist:hover label {
63     display: block;
64     height: 100%;
65 }
66
67
68 fieldset.elist:hover input[type="radio"]:checked ~ label {
69     background-color: #aaaaaa;
70 }

```

3.2 Language and UI Styles

3.2.1 lang_eng.css

```

1  .bar_title:before{content: "Fraction Bar";}
2  .c_bar:before{content: "Bar";}
3  .c_mat:before{content: "Mat";}
4  .c_copy:before{content: "Copy";}
5  .c_repeat:before{content: "Repeat";}
6  .c_iterate:before{content: "Iterate";}
7  .c_join:before{content: "Join";}
8  .c_delete:before{content: "Delete";}
9  .c_parts:before{content: "Parts";}
10 .c_pieces:before{content: "Line";}
11 .c_b_apart:before{content: "Break Apart";}
12 .c_pullout:before{content: "Pull Out Parts";}
13 .c_c_parts:before{content: "Clear Parts";}
14 .c_set_unit:before{content: "Set Unit Bar";}
15 .c_measure:before{content: "Measure";}
16 .c_label:before{content: "Label";}
17 .c_undo:before{content: "Undo";}
18 .c_redo:before{content: "Redo";}
19 .c_save:before{content: "Save";}
20 .c_open:before{content: "Open";}
21 .c_new:before{content: "New";}
22 .c_print:before{content: "Print";}
23 .c_properties:before{content: "Properties";}
24 .c_hide:before{content: "Hide";}
25 .c_show:before{content: "Show";}
26 .c_previous:before{content: "Previous";}
27 .c_next:before{content: "Next";}
28 .c_dialog_splits title:before {content: "Partition part of bar";}
29 .c_vertical:before{content: "Vertical";}
30 .c_horizontal:before{content: "Horizontal";}
31 .c_number_part:before{content: "Number of parts";}
32 .c_part_whole:before{content: "Partition whole bar";}
33 .c_part_part:before{content: "Partition part of bar";}
34 .c_dialog_properties title:before {content: "Partition part of bar";}
35 .c_iterations:before{content: "Iterations";}
36 .c_dont_create:before{content: "Don't Create New Bar";}
37 .c_create_new:before{content: "Create New Bar";}
38 .c_two_way:before{content: "Two way Iterate";}
39 .c_one_way:before{content: "Only one way Iterate";}
40 .c_splits:before{content: "Splits (Parts)";}
41 .c_vert_horiz:before{content: "Vertical or Horizontal Split";}
42 .c_only_vert:before{content: "Only vertical split";}
43 .c_lang:before{content: "Language";}
44 .c_lang_tur:before{content: "Turkish";}
45 .c_lang_eng:before{content: "English";}
46 .c_color:before{content: "Background Color";}
47 .c_dialog_iterate title:before {content: "Iterate";}
48 .c_number_iterations:before{content: "Number of iterations";}
49 .c_c_choose_file title:before {content: "Choose a File";}
50 .c_c_open_file:before{content: "Use the button below to choose a FractionBars file to open.";}
51 .c_split_alert:before{content: "Please select a bar to partition.";}

```

4 AceofBases Program

4.1 HTML Files

4.1.1 index.html

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Ace of Base Arithmetic</title>
7   <style>
8     body {
9       font-family: Arial, sans-serif;
10      display: flex;
11      justify-content: center;
12      align-items: center;
13      height: 100vh;
14      margin: 0;
15      background-color: #f9f9f9;
16    }
17
18    .container {
19      text-align: center;
20      border: 1px solid #ccc;
21      padding: 20px;
22      border-radius: 10px;
23      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
24      background-color: white;
25    }
26
27    canvas {
28      border: 1px solid #000;
29    }
30
31    .controls {
32      margin-top: 20px;
33    }
34
35    button {
36      margin: 5px;
37    }
38
39    .results {
40      margin-top: 20px;
41    }
42
43    .popup {
44      display: none;
45      position: fixed;
46      left: 50%;
47      top: 50%;
48      transform: translate(-50%, -50%);
49      padding: 20px;
50      background-color: white;
51      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
52      border-radius: 10px;
53      z-index: 1000;
54    }
55
56    .popup .close {
57      cursor: pointer;
58      background-color: red;
59      color: white;
60      border: none;
61      border-radius: 5px;
62      padding: 5px 10px;
63      margin-top: 10px;
64    }
65  </style>
66 </head>
67 <body>
68   <div class="container">
```

```

69     <h1>Ace of Base Arithmetic</h1>
70     <p>Circle the number of squares you want to be your grouping unit:</p>
71     <div class="instructions">
72         <p>Drag to select between 2 and 15 cubes:</p>
73         <canvas id="cubeCanvas" width="600" height="400"></canvas>
74     </div>
75     <div class="controls">
76         <p id="selectedUnits">Selected Units: 0</p>
77         <button id="composeButton">Compose</button>
78         <button id="decomposeButton">Decompose</button>
79         <button id="addCubeButton">Add Cube</button>
80         <button id="removeCubeButton">Remove Cube</button>
81     </div>
82     <div class="results">
83         <p id="baseConversion"></p>
84         <p id="baseTenCount"></p>
85     </div>
86 </div>
87 <div id="explanationPopup" class="popup">
88     <p>With bases larger than ten, we need different symbols for the numbers called ten, eleven,
89     ↪ twelve, thirteen, fourteen, etc., in base ten. Here is how they are represented:</p>
90     <ul>
91         <li>What is called ten in base ten will be represented with the digit T.</li>
92         <li>What is called eleven in base ten will be represented with the digit E.</li>
93         <li>What is called twelve in base ten will be represented with the digit D.</li>
94         <li>What is called thirteen in base ten will be represented with the digit R.</li>
95         <li>What is called fourteen in base ten will be represented with the digit F.</li>
96     </ul>
97     <button class="close"
98     ↪ onclick="document.getElementById('explanationPopup').style.display='none'">Close</button>
99 </div>
100 <div id="overflowPopup" class="popup">
101     <p>Please choose a larger grouping unit.</p>
102     <button class="close"
103     ↪ onclick="document.getElementById('overflowPopup').style.display='none'">Close</button>
104 </div>
105 <script src="script.js"></script>
106 </body>
107 </html>

```

4.1.2 index_ace_of_bases.html

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Ace of Base Arithmetic</title>
7      <link rel="stylesheet" href="styles_ace_of_bases.css">
8  </head>
9  <body>
10     <div class="container">
11         <h1>Ace of Base Arithmetic</h1>
12         <p>Circle the number of squares you want to be your grouping unit:</p>
13         <div class="instructions">
14             <p>Drag to select between 2 and 15 cubes:</p>
15             <canvas id="cubeCanvas" width="600" height="400"></canvas>
16         </div>
17         <div class="controls">
18             <p id="selectedUnits">Selected Units: 0</p>
19             <button id="composeButton">Compose</button>
20             <button id="decomposeButton">Decompose</button>
21             <button id="addCubeButton">Add Cube</button>
22             <button id="removeCubeButton">Remove Cube</button>
23         </div>
24         <div class="results">
25             <p id="baseConversion"></p>
26             <p id="baseTenCount"></p>
27         </div>
28     </div>
29     <div id="explanationPopup" class="popup">
30         <p>With bases larger than ten, we need different symbols for the numbers called ten, eleven,
31         ↪ twelve, thirteen, fourteen, etc., in base ten. Here is how they are represented:</p>

```

```

31     <ul>
32         <li>What is called ten in base ten will be represented with the digit T.</li>
33         <li>What is called eleven in base ten will be represented with the digit E.</li>
34         <li>What is called twelve in base ten will be represented with the digit D.</li>
35         <li>What is called thirteen in base ten will be represented with the digit R.</li>
36         <li>What is called fourteen in base ten will be represented with the digit F.</li>
37     </ul>
38     <button class="close"
39         ↳ onclick="document.getElementById('explanationPopup').style.display='none'">Close</button>
40 </div>
41 <div id="overflowPopup" class="popup">
42     <p>Please choose a larger grouping unit.</p>
43     <button class="close"
44         ↳ onclick="document.getElementById('overflowPopup').style.display='none'">Close</button>
45 </div>
46 <script src="script_ace_of_bases.js"></script>
47 </body>
48 </html>

```

4.2 JavaScript Files

4.2.1 script.js

```

1 document.addEventListener('DOMContentLoaded', () => {
2     const canvas = document.getElementById('cubeCanvas');
3     const ctx = canvas.getContext('2d');
4     const composeButton = document.getElementById('composeButton');
5     const decomposeButton = document.getElementById('decomposeButton');
6     const addCubeButton = document.getElementById('addCubeButton');
7     const removeCubeButton = document.getElementById('removeCubeButton');
8     const selectedUnitsDisplay = document.getElementById('selectedUnits');
9     const baseConversionDisplay = document.getElementById('baseConversion');
10    const baseTenCountDisplay = document.getElementById('baseTenCount');
11    const explanationPopup = document.getElementById('explanationPopup');
12    const overflowPopup = document.getElementById('overflowPopup');
13
14    let cubes = [];
15    let selectedUnits = 0;
16    let modulus = 0;
17    let isDragging = false;
18    let dragOffsetX, dragOffsetY;
19    let draggedCube = null;
20    let isSelecting = false;
21    let selectionStartX, selectionStartY;
22
23    const generateRandomCubes = () => {
24        const count = Math.floor(Math.random() * 14) + 2;
25        cubes = Array.from({ length: count }, (_, i) => ({
26            x: Math.random() * (canvas.width - 20),
27            y: Math.random() * (canvas.height - 20),
28            size: 20,
29        }));
30        drawCubes();
31    };
32
33    const drawCubes = () => {
34        ctx.clearRect(0, 0, canvas.width, canvas.height);
35        cubes.forEach(cube => {
36            ctx.fillStyle = 'blue';
37            ctx.fillRect(cube.x, cube.y, cube.size, cube.size);
38        });
39    };
40
41    const handleMouseDown = (e) => {
42        const rect = canvas.getBoundingClientRect();
43        const startX = e.clientX - rect.left;
44        const startY = e.clientY - rect.top;
45
46        draggedCube = cubes.find(cube => (
47            startX >= cube.x && startX <= cube.x + cube.size &&
48            startY >= cube.y && startY <= cube.y + cube.size
49        ));

```



```

50
51     if (draggedCube) {
52         isDragging = true;
53         dragOffsetX = startX - draggedCube.x;
54         dragOffsetY = startY - draggedCube.y;
55     } else {
56         isSelecting = true;
57         selectionStartX = startX;
58         selectionStartY = startY;
59         canvas.addEventListener('mousemove', handleMouseMove);
60         canvas.addEventListener('mouseup', handleMouseUp);
61     }
62 };
63
64 const handleMouseMove = (e) => {
65     if (isSelecting) {
66         const rect = canvas.getBoundingClientRect();
67         const currentX = e.clientX - rect.left;
68         const currentY = e.clientY - rect.top;
69         ctx.clearRect(0, 0, canvas.width, canvas.height);
70         drawCubes();
71         ctx.strokeStyle = 'red';
72         ctx.strokeRect(selectionStartX, selectionStartY, currentX - selectionStartX, currentY -
        ↪ selectionStartY);
73     }
74 };
75
76 const handleMouseUp = (e) => {
77     if (isSelecting) {
78         const rect = canvas.getBoundingClientRect();
79         const endX = e.clientX - rect.left;
80         const endY = e.clientY - rect.top;
81         const selected = cubes.filter(cube => (
82             cube.x >= Math.min(selectionStartX, endX) && cube.x <= Math.max(selectionStartX, endX) &&
83             cube.y >= Math.min(selectionStartY, endY) && cube.y <= Math.max(selectionStartY, endY)
84         ));
85         selectedUnits = Math.min(selected.length, 15);
86         selectedUnitsDisplay.textContent = `Selected Units: ${selectedUnits}`;
87         modulus = selectedUnits;
88
89         if (selectedUnits > 10) {
90             explanationPopup.style.display = 'block';
91         } else {
92             explanationPopup.style.display = 'none';
93         }
94
95         isSelecting = false;
96         canvas.removeEventListener('mousemove', handleMouseMove);
97         canvas.removeEventListener('mouseup', handleMouseUp);
98     }
99 };
100
101 const handleTouchStart = (e) => {
102     const touch = e.touches[0];
103     const rect = canvas.getBoundingClientRect();
104     const startX = touch.clientX - rect.left;
105     const startY = touch.clientY - rect.top;
106
107     draggedCube = cubes.find(cube => (
108         startX >= cube.x && startX <= cube.x + cube.size &&
109         startY >= cube.y && startY <= cube.y + cube.size
110     ));
111
112     if (draggedCube) {
113         isDragging = true;
114         dragOffsetX = startX - draggedCube.x;
115         dragOffsetY = startY - draggedCube.y;
116     } else {
117         isSelecting = true;
118         selectionStartX = startX;
119         selectionStartY = startY;
120         canvas.addEventListener('touchmove', handleTouchMove);
121         canvas.addEventListener('touchend', handleTouchEnd);
122     }
123 };

```

```

124
125 const handleTouchMove = (e) => {
126   if (isSelecting) {
127     const touch = e.touches[0];
128     const rect = canvas.getBoundingClientRect();
129     const currentX = touch.clientX - rect.left;
130     const currentY = touch.clientY - rect.top;
131     ctx.clearRect(0, 0, canvas.width, canvas.height);
132     drawCubes();
133     ctx.strokeStyle = 'red';
134     ctx.strokeRect(selectionStartX, selectionStartY, currentX - selectionStartX, currentY -
      ↪ selectionStartY);
135   } else if (isDragging && draggedCube) {
136     const touch = e.touches[0];
137     const rect = canvas.getBoundingClientRect();
138     draggedCube.x = touch.clientX - rect.left - dragOffsetX;
139     draggedCube.y = touch.clientY - rect.top - dragOffsetY;
140     drawCubes();
141   }
142 };
143
144 const handleTouchEnd = (e) => {
145   if (isSelecting) {
146     const rect = canvas.getBoundingClientRect();
147     const touch = e.changedTouches[0];
148     const endX = touch.clientX - rect.left;
149     const endY = touch.clientY - rect.top;
150     const selected = cubes.filter(cube => (
151       cube.x >= Math.min(selectionStartX, endX) && cube.x <= Math.max(selectionStartX, endX) &&
152       cube.y >= Math.min(selectionStartY, endY) && cube.y <= Math.max(selectionStartY, endY)
153     ));
154     selectedUnits = Math.min(selected.length, 15);
155     selectedUnitsDisplay.textContent = `Selected Units: ${selectedUnits}`;
156     modulus = selectedUnits;
157
158     if (selectedUnits > 10) {
159       explanationPopup.style.display = 'block';
160     } else {
161       explanationPopup.style.display = 'none';
162     }
163
164     isSelecting = false;
165     canvas.removeEventListener('touchmove', handleTouchMove);
166     canvas.removeEventListener('touchend', handleTouchEnd);
167   }
168 };
169
170 const handleDrag = (e) => {
171   if (isDragging && draggedCube) {
172     const rect = canvas.getBoundingClientRect();
173     draggedCube.x = e.clientX - rect.left - dragOffsetX;
174     draggedCube.y = e.clientY - rect.top - dragOffsetY;
175     drawCubes();
176   }
177 };
178
179 const handleDragEnd = (e) => {
180   isDragging = false;
181   draggedCube = null;
182 };
183
184 const updateBaseConversion = () => {
185   if (modulus > 1) {
186     const base = modulus;
187     const baseStr = convertToBase(cubes.length, base);
188
189     if (baseStr.length > 4) {
190       overflowPopup.style.display = 'block';
191       return;
192     }
193
194     const baseComponents = { rods: 0, flats: 0, cubes3D: 0, units: 0 };
195     let count = cubes.length;
196
197     while (count > 0) {

```

```

198         if (count >= base * base * base) {
199             baseComponents.cubes3D++;
200             count -= base * base * base;
201         } else if (count >= base * base) {
202             baseComponents.flats++;
203             count -= base * base;
204         } else if (count >= base) {
205             baseComponents.rods++;
206             count -= base;
207         } else {
208             baseComponents.units++;
209             count--;
210         }
211     }
212
213     baseConversionDisplay.textContent = `Base ${base}: ${baseStr}`;
214     baseTenCountDisplay.textContent = `Base 10: ${cubes.length}`;
215     drawBaseComponents(base, baseComponents);
216 }
217 };
218
219 const convertToBase = (number, base) => {
220     const digitMap = { 10: 'T', 11: 'E', 12: 'D', 13: 'R', 14: 'F' };
221     let result = '';
222     while (number > 0) {
223         let digit = number % base;
224         if (digit >= 10 && digit <= 14) {
225             digit = digitMap[digit];
226         }
227         result = digit.toString() + result;
228         number = Math.floor(number / base);
229     }
230     return result;
231 };
232
233 const drawBaseComponents = (base, baseComponents) => {
234     ctx.clearRect(0, 0, canvas.width, canvas.height);
235     let xOffset = 0;
236     let yOffset = 0;
237
238     // Draw units
239     for (let i = 0; i < baseComponents.units; i++) {
240         ctx.fillStyle = 'blue';
241         ctx.fillRect(xOffset, yOffset, 20, 20);
242         xOffset += 25;
243         if (xOffset > canvas.width - 20) {
244             xOffset = 0;
245             yOffset += 25;
246         }
247     }
248
249     // Draw rods
250     for (let i = 0; i < baseComponents.rods; i++) {
251         ctx.fillStyle = 'green';
252         ctx.fillRect(xOffset, yOffset, 20 * base, 20);
253         for (let j = 0; j < base; j++) {
254             ctx.strokeStyle = 'black';
255             ctx.strokeRect(xOffset + j * 20, yOffset, 20, 20);
256         }
257         xOffset += 20 * base + 5;
258         if (xOffset > canvas.width - 20 * base) {
259             xOffset = 0;
260             yOffset += 25;
261         }
262     }
263
264     // Draw flats
265     for (let i = 0; i < baseComponents.flats; i++) {
266         ctx.fillStyle = 'yellow';
267         ctx.fillRect(xOffset, yOffset, 20 * base, 20 * base);
268         for (let j = 0; j < base; j++) {
269             for (let k = 0; k < base; k++) {
270                 ctx.strokeStyle = 'black';
271                 ctx.strokeRect(xOffset + j * 20, yOffset + k * 20, 20, 20);
272             }

```

```

273     }
274     xOffset += 20 * base + 5;
275     if (xOffset > canvas.width - 20 * base) {
276         xOffset = 0;
277         yOffset += 20 * base + 5;
278     }
279 }
280
281 // Draw 3D cubes
282 for (let i = 0; i < baseComponents.cubes3D; i++) {
283     ctx.fillStyle = 'red';
284     const cubeSize = 20 * base;
285     const depth = cubeSize / 3;
286
287     // Draw front face
288     ctx.fillRect(xOffset, yOffset, cubeSize, cubeSize);
289     ctx.strokeStyle = 'black';
290     for (let j = 0; j < base; j++) {
291         for (let k = 0; k < base; k++) {
292             ctx.strokeRect(xOffset + j * 20, yOffset + k * 20, 20, 20);
293         }
294     }
295
296     // Draw top face
297     ctx.beginPath();
298     ctx.moveTo(xOffset, yOffset);
299     ctx.lineTo(xOffset + depth, yOffset - depth);
300     ctx.lineTo(xOffset + cubeSize + depth, yOffset - depth);
301     ctx.lineTo(xOffset + cubeSize, yOffset);
302     ctx.closePath();
303     ctx.fillStyle = 'rgba(255, 0, 0, 0.8)';
304     ctx.fill();
305     ctx.stroke();
306
307     // Draw right face
308     ctx.beginPath();
309     ctx.moveTo(xOffset + cubeSize, yOffset);
310     ctx.lineTo(xOffset + cubeSize + depth, yOffset - depth);
311     ctx.lineTo(xOffset + cubeSize + depth, yOffset + cubeSize - depth);
312     ctx.lineTo(xOffset + cubeSize, yOffset + cubeSize);
313     ctx.closePath();
314     ctx.fillStyle = 'rgba(255, 0, 0, 0.6)';
315     ctx.fill();
316     ctx.stroke();
317
318     xOffset += cubeSize + depth + 5;
319     if (xOffset > canvas.width - cubeSize) {
320         xOffset = 0;
321         yOffset += cubeSize + depth + 5;
322     }
323 }
324 };
325
326 const decomposeCubes = () => {
327     cubes.forEach(cube => {
328         cube.x = Math.random() * (canvas.width - 20);
329         cube.y = Math.random() * (canvas.height - 20);
330     });
331     drawCubes();
332     selectedUnits = 0;
333     selectedUnitsDisplay.textContent = `Selected Units: ${selectedUnits}`;
334     baseConversionDisplay.textContent = '';
335     baseTenCountDisplay.textContent = '';
336 };
337
338 const addCube = () => {
339     cubes.push({
340         x: Math.random() * (canvas.width - 20),
341         y: Math.random() * (canvas.height - 20),
342         size: 20,
343     });
344     drawCubes();
345 };
346
347 const removeCube = () => {

```

```

348     if (cubes.length > 0) {
349         cubes.pop();
350         drawCubes();
351     }
352 };
353
354 canvas.addEventListener('mousedown', handleMouseDown);
355 canvas.addEventListener('touchstart', handleTouchStart);
356 canvas.addEventListener('mousemove', handleDrag);
357 canvas.addEventListener('mouseup', handleDragEnd);
358 canvas.addEventListener('touchmove', handleTouchMove);
359 canvas.addEventListener('touchend', handleDragEnd);
360
361 composeButton.addEventListener('click', updateBaseConversion);
362 decomposeButton.addEventListener('click', decomposeCubes);
363 addCubeButton.addEventListener('click', addCube);
364 removeCubeButton.addEventListener('click', removeCube);
365
366 generateRandomCubes();
367 });

```

4.2.2 script_ace_of_bases.js

```

1 document.addEventListener('DOMContentLoaded', () => {
2     const canvas = document.getElementById('cubeCanvas');
3     const ctx = canvas.getContext('2d');
4     const composeButton = document.getElementById('composeButton');
5     const decomposeButton = document.getElementById('decomposeButton');
6     const addCubeButton = document.getElementById('addCubeButton');
7     const removeCubeButton = document.getElementById('removeCubeButton');
8     const selectedUnitsDisplay = document.getElementById('selectedUnits');
9     const baseConversionDisplay = document.getElementById('baseConversion');
10    const baseTenCountDisplay = document.getElementById('baseTenCount');
11    const explanationPopup = document.getElementById('explanationPopup');
12    const overflowPopup = document.getElementById('overflowPopup');
13
14    let cubes = [];
15    let selectedUnits = 0;
16    let modulus = 0;
17    let isDragging = false;
18    let dragOffsetX, dragOffsetY;
19    let draggedCube = null;
20    let isSelecting = false;
21    let selectionStartX, selectionStartY;
22
23    const generateRandomCubes = () => {
24        const count = Math.floor(Math.random() * 14) + 2;
25        cubes = Array.from({ length: count }, (_, i) => ({
26            x: Math.random() * (canvas.width - 20),
27            y: Math.random() * (canvas.height - 20),
28            size: 20,
29        }));
30        drawCubes();
31    };
32
33    const drawCubes = () => {
34        ctx.clearRect(0, 0, canvas.width, canvas.height);
35        cubes.forEach(cube => {
36            ctx.fillStyle = 'blue';
37            ctx.fillRect(cube.x, cube.y, cube.size, cube.size);
38        });
39    };
40
41    const handleMouseDown = (e) => {
42        const rect = canvas.getBoundingClientRect();
43        const startX = e.clientX - rect.left;
44        const startY = e.clientY - rect.top;
45
46        draggedCube = cubes.find(cube => (
47            startX >= cube.x && startX <= cube.x + cube.size &&
48            startY >= cube.y && startY <= cube.y + cube.size
49        ));
50    };

```

```

51     if (draggedCube) {
52         isDragging = true;
53         dragOffsetX = startX - draggedCube.x;
54         dragOffsetY = startY - draggedCube.y;
55     } else {
56         isSelecting = true;
57         selectionStartX = startX;
58         selectionStartY = startY;
59         canvas.addEventListener('mousemove', handleMouseMove);
60         canvas.addEventListener('mouseup', handleMouseUp);
61     }
62 };
63
64 const handleMouseMove = (e) => {
65     if (isSelecting) {
66         const rect = canvas.getBoundingClientRect();
67         const currentX = e.clientX - rect.left;
68         const currentY = e.clientY - rect.top;
69         ctx.clearRect(0, 0, canvas.width, canvas.height);
70         drawCubes();
71         ctx.strokeStyle = 'red';
72         ctx.strokeRect(selectionStartX, selectionStartY, currentX - selectionStartX, currentY -
        ↪ selectionStartY);
73     }
74 };
75
76 const handleMouseUp = (e) => {
77     if (isSelecting) {
78         const rect = canvas.getBoundingClientRect();
79         const endX = e.clientX - rect.left;
80         const endY = e.clientY - rect.top;
81         const selected = cubes.filter(cube => (
82             cube.x >= Math.min(selectionStartX, endX) && cube.x <= Math.max(selectionStartX, endX) &&
83             cube.y >= Math.min(selectionStartY, endY) && cube.y <= Math.max(selectionStartY, endY)
84         ));
85         selectedUnits = Math.min(selected.length, 15);
86         selectedUnitsDisplay.textContent = `Selected Units: ${selectedUnits}`;
87         modulus = selectedUnits;
88
89         if (selectedUnits > 10) {
90             explanationPopup.style.display = 'block';
91         } else {
92             explanationPopup.style.display = 'none';
93         }
94
95         isSelecting = false;
96         canvas.removeEventListener('mousemove', handleMouseMove);
97         canvas.removeEventListener('mouseup', handleMouseUp);
98     }
99 };
100
101 const handleTouchStart = (e) => {
102     const touch = e.touches[0];
103     const rect = canvas.getBoundingClientRect();
104     const startX = touch.clientX - rect.left;
105     const startY = touch.clientY - rect.top;
106
107     draggedCube = cubes.find(cube => (
108         startX >= cube.x && startX <= cube.x + cube.size &&
109         startY >= cube.y && startY <= cube.y + cube.size
110     ));
111
112     if (draggedCube) {
113         isDragging = true;
114         dragOffsetX = startX - draggedCube.x;
115         dragOffsetY = startY - draggedCube.y;
116     } else {
117         isSelecting = true;
118         selectionStartX = startX;
119         selectionStartY = startY;
120         canvas.addEventListener('touchmove', handleTouchMove);
121         canvas.addEventListener('touchend', handleTouchEnd);
122     }
123 };
124

```

```

125     const handleTouchMove = (e) => {
126         if (isSelecting) {
127             const touch = e.touches[0];
128             const rect = canvas.getBoundingClientRect();
129             const currentX = touch.clientX - rect.left;
130             const currentY = touch.clientY - rect.top;
131             ctx.clearRect(0, 0, canvas.width, canvas.height);
132             drawCubes();
133             ctx.strokeStyle = 'red';
134             ctx.strokeRect(selectionStartX, selectionStartY, currentX - selectionStartX, currentY -
                ↪ selectionStartY);
135         } else if (isDragging && draggedCube) {
136             const touch = e.touches[0];
137             const rect = canvas.getBoundingClientRect();
138             draggedCube.x = touch.clientX - rect.left - dragOffsetX;
139             draggedCube.y = touch.clientY - rect.top - dragOffsetY;
140             drawCubes();
141         }
142     };
143
144     const handleTouchEnd = (e) => {
145         if (isSelecting) {
146             const rect = canvas.getBoundingClientRect();
147             const touch = e.changedTouches[0];
148             const endX = touch.clientX - rect.left;
149             const endY = touch.clientY - rect.top;
150             const selected = cubes.filter(cube => (
151                 cube.x >= Math.min(selectionStartX, endX) && cube.x <= Math.max(selectionStartX, endX) &&
152                 cube.y >= Math.min(selectionStartY, endY) && cube.y <= Math.max(selectionStartY, endY)
153             ));
154             selectedUnits = Math.min(selected.length, 15);
155             selectedUnitsDisplay.textContent = `Selected Units: ${selectedUnits}`;
156             modulus = selectedUnits;
157
158             if (selectedUnits > 10) {
159                 explanationPopup.style.display = 'block';
160             } else {
161                 explanationPopup.style.display = 'none';
162             }
163
164             isSelecting = false;
165             canvas.removeEventListener('touchmove', handleTouchMove);
166             canvas.removeEventListener('touchend', handleTouchEnd);
167         }
168     };
169
170     const handleDrag = (e) => {
171         if (isDragging && draggedCube) {
172             const rect = canvas.getBoundingClientRect();
173             draggedCube.x = e.clientX - rect.left - dragOffsetX;
174             draggedCube.y = e.clientY - rect.top - dragOffsetY;
175             drawCubes();
176         }
177     };
178
179     const handleDragEnd = (e) => {
180         isDragging = false;
181         draggedCube = null;
182     };
183
184     const updateBaseConversion = () => {
185         if (modulus > 1) {
186             const base = modulus;
187             const baseStr = convertToBase(cubes.length, base);
188
189             if (baseStr.length > 4) {
190                 overflowPopup.style.display = 'block';
191                 return;
192             }
193
194             const baseComponents = { rods: 0, flats: 0, cubes3D: 0, units: 0 };
195             let count = cubes.length;
196
197             while (count > 0) {
198                 if (count >= base * base * base) {

```

```

199         baseComponents.cubes3D++;
200         count -= base * base * base;
201     } else if (count >= base * base) {
202         baseComponents.flats++;
203         count -= base * base;
204     } else if (count >= base) {
205         baseComponents.rods++;
206         count -= base;
207     } else {
208         baseComponents.units++;
209         count--;
210     }
211 }
212
213 baseConversionDisplay.textContent = `Base ${base}: ${baseStr}`;
214 baseTenCountDisplay.textContent = `Base 10: ${cubes.length}`;
215 drawBaseComponents(base, baseComponents);
216 }
217 };
218
219 const convertToBase = (number, base) => {
220     const digitMap = { 10: 'T', 11: 'E', 12: 'D', 13: 'R', 14: 'F' };
221     let result = '';
222     while (number > 0) {
223         let digit = number % base;
224         if (digit >= 10 && digit <= 14) {
225             digit = digitMap[digit];
226         }
227         result = digit.toString() + result;
228         number = Math.floor(number / base);
229     }
230     return result;
231 };
232
233 const drawBaseComponents = (base, baseComponents) => {
234     ctx.clearRect(0, 0, canvas.width, canvas.height);
235     let xOffset = 0;
236     let yOffset = 0;
237
238     // Draw units
239     for (let i = 0; i < baseComponents.units; i++) {
240         ctx.fillStyle = 'blue';
241         ctx.fillRect(xOffset, yOffset, 20, 20);
242         xOffset += 25;
243         if (xOffset > canvas.width - 20) {
244             xOffset = 0;
245             yOffset += 25;
246         }
247     }
248
249     // Draw rods
250     for (let i = 0; i < baseComponents.rods; i++) {
251         ctx.fillStyle = 'green';
252         ctx.fillRect(xOffset, yOffset, 20 * base, 20);
253         for (let j = 0; j < base; j++) {
254             ctx.strokeStyle = 'black';
255             ctx.strokeRect(xOffset + j * 20, yOffset, 20, 20);
256         }
257         xOffset += 20 * base + 5;
258         if (xOffset > canvas.width - 20 * base) {
259             xOffset = 0;
260             yOffset += 25;
261         }
262     }
263
264     // Draw flats
265     for (let i = 0; i < baseComponents.flats; i++) {
266         ctx.fillStyle = 'yellow';
267         ctx.fillRect(xOffset, yOffset, 20 * base, 20 * base);
268         for (let j = 0; j < base; j++) {
269             for (let k = 0; k < base; k++) {
270                 ctx.strokeStyle = 'black';
271                 ctx.strokeRect(xOffset + j * 20, yOffset + k * 20, 20, 20);
272             }
273         }

```



```

274     xOffset += 20 * base + 5;
275     if (xOffset > canvas.width - 20 * base) {
276         xOffset = 0;
277         yOffset += 20 * base + 5;
278     }
279 }
280
281 // Draw 3D cubes
282 for (let i = 0; i < baseComponents.cubes3D; i++) {
283     ctx.fillStyle = 'red';
284     const cubeSize = 20 * base;
285     const depth = cubeSize / 3;
286
287     // Draw front face
288     ctx.fillRect(xOffset, yOffset, cubeSize, cubeSize);
289     ctx.strokeStyle = 'black';
290     for (let j = 0; j < base; j++) {
291         for (let k = 0; k < base; k++) {
292             ctx.strokeRect(xOffset + j * 20, yOffset + k * 20, 20, 20);
293         }
294     }
295
296     // Draw top face
297     ctx.beginPath();
298     ctx.moveTo(xOffset, yOffset);
299     ctx.lineTo(xOffset + depth, yOffset - depth);
300     ctx.lineTo(xOffset + cubeSize + depth, yOffset - depth);
301     ctx.lineTo(xOffset + cubeSize, yOffset);
302     ctx.closePath();
303     ctx.fillStyle = 'rgba(255, 0, 0, 0.8)';
304     ctx.fill();
305     ctx.stroke();
306
307     // Draw right face
308     ctx.beginPath();
309     ctx.moveTo(xOffset + cubeSize, yOffset);
310     ctx.lineTo(xOffset + cubeSize + depth, yOffset - depth);
311     ctx.lineTo(xOffset + cubeSize + depth, yOffset + cubeSize - depth);
312     ctx.lineTo(xOffset + cubeSize, yOffset + cubeSize);
313     ctx.closePath();
314     ctx.fillStyle = 'rgba(255, 0, 0, 0.6)';
315     ctx.fill();
316     ctx.stroke();
317
318     xOffset += cubeSize + depth + 5;
319     if (xOffset > canvas.width - cubeSize) {
320         xOffset = 0;
321         yOffset += cubeSize + depth + 5;
322     }
323 }
324 };
325
326 const decomposeCubes = () => {
327     cubes.forEach(cube => {
328         cube.x = Math.random() * (canvas.width - 20);
329         cube.y = Math.random() * (canvas.height - 20);
330     });
331     drawCubes();
332     selectedUnits = 0;
333     selectedUnitsDisplay.textContent = `Selected Units: ${selectedUnits}`;
334     baseConversionDisplay.textContent = '';
335     baseTenCountDisplay.textContent = '';
336 };
337
338 const addCube = () => {
339     cubes.push({
340         x: Math.random() * (canvas.width - 20),
341         y: Math.random() * (canvas.height - 20),
342         size: 20,
343     });
344     drawCubes();
345 };
346
347 const removeCube = () => {
348     if (cubes.length > 0) {

```

```

349         cubes.pop();
350         drawCubes();
351     }
352 };
353
354 canvas.addEventListener('mousedown', handleMouseDown);
355 canvas.addEventListener('touchstart', handleTouchStart);
356 canvas.addEventListener('mousemove', handleDrag);
357 canvas.addEventListener('mouseup', handleDragEnd);
358 canvas.addEventListener('touchmove', handleTouchMove);
359 canvas.addEventListener('touchend', handleDragEnd);
360
361 composeButton.addEventListener('click', updateBaseConversion);
362 decomposeButton.addEventListener('click', decomposeCubes);
363 addCubeButton.addEventListener('click', addCube);
364 removeCubeButton.addEventListener('click', removeCube);
365
366 generateRandomCubes();
367 });

```

4.3 CSS Files

4.3.1 styles.css

```

1 body {
2     font-family: Arial, sans-serif;
3     display: flex;
4     justify-content: center;
5     align-items: center;
6     height: 100vh;
7     margin: 0;
8     background-color: #f9f9f9;
9 }
10
11 .container {
12     text-align: center;
13     border: 1px solid #ccc;
14     padding: 20px;
15     border-radius: 10px;
16     box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
17     background-color: white;
18 }
19
20 canvas {
21     border: 1px solid #000;
22 }
23
24 .controls {
25     margin-top: 20px;
26 }
27
28 button {
29     margin: 5px;
30 }
31
32 .results {
33     margin-top: 20px;
34 }

```

4.3.2 styles__ace_of_bases.css

```

1 body {
2     font-family: Arial, sans-serif;
3     display: flex;
4     justify-content: center;
5     align-items: center;
6     height: 100vh;
7     margin: 0;
8     background-color: #f9f9f9;
9 }

```

```

10
11 .container {
12     text-align: center;
13     border: 1px solid #ccc;
14     padding: 20px;
15     border-radius: 10px;
16     box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
17     background-color: white;
18 }
19
20 canvas {
21     border: 1px solid #000;
22 }
23
24 .controls,
25 button,
26 .results {
27     margin-top: 20px;
28 }
29
30 button {
31     margin: 5px;
32 }

```

5 Configuration Files

5.1 Fraction_Bars.code-workspace

```

1 {
2     "folders": [
3         {
4             "path": "."
5         },
6         {
7             "path": "../../../Desktop/Spring 2025/GPT4_1/Fraction_Bars_files"
8         }
9     ],
10    "settings": {
11        "liveServer.settings.multiRootWorkspaceName": "Fraction_Bars"
12    }
13 }

```
