



《软件界面设计》课程设计报告

UI Design Experiment Report

学生所在学院：软件学院

学生所在班级：22软件 班

学生姓名：

学 号：

教 务 处

2024年 9 月

实验题目：Candy Crush游戏设计与实现

1. **项目内容**

**用户界面**

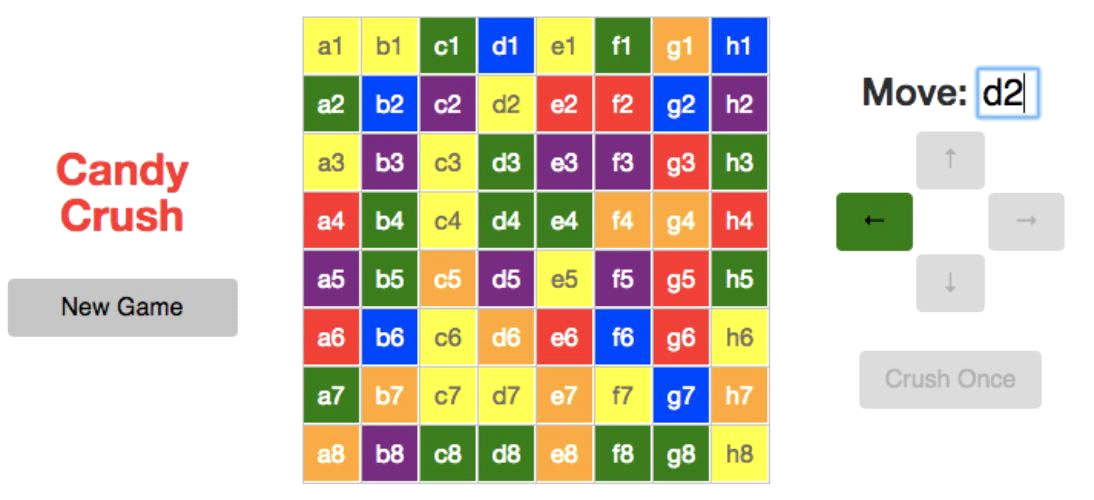


图1 使用标准HTML元素的简单消消乐UI

**基本布局**

首先，构建如图 1 所示的界面。需要注意：

* 左列（firstColumn）按钮(buttun)、消消乐网格（candy crush grid）和右列（lastColumn）按钮(buttun)之间的间距（margin)应至少为10px。
* 屏幕上的所有元素都应使用三列（Column）布局。index.html中提供了一个模板，使用了bootstrap元素。Bootstrap是一个有助于格式化的CSS框架。修改模板，使中间列占页面的50%。

1. 第一列（firstColumn）应包含两个元素：

* 游戏标题<title>（粗体和彩色）
* 一个“New Game”按钮
* 按钮应该是<input>元素。因为我们已经包含了Bootstrap，你可以添加class="btn" 添加到你的按钮以获得现代风格的按钮。

1. 第二列（mainColumn）应该包含游戏面板（candy crush grid）：

* 游戏板应具有320x320像素的固定尺寸。它应该有8行，每行应该有8个正方形（board.square)。
* 你应该使用<table>构建游戏板，每个“糖果”都是表中的一个单元格<td>。你可能需要编写javascript代码来自动构建此表。
* 在本实验中，“糖果”将是彩色方格。每种糖果都有不同的背景颜色。在消消乐2和消消乐3中，你将使用糖果图像而不是彩色方格。
* 若要生成糖果，首先，你调用rules.js中的rules.prepareNewGame方法，此方法生成一个新的有效游戏面板。接着为了将糖果添加（add）到游戏面板中，它会生成对add事件监听器的调用，你需要在index.html中实现该监听器。
* 为了完成此游戏，我们按行（Row 1-8）和列（Column a-h）命名每种糖果。用文本坐标（a1、b3等）标注每个彩色方格，确保这些坐标位于方格（square）的中心。为确保文本坐标清晰易读，可以将文本坐标设为白色，而对于黄色糖果，可以使用更深的颜色，比如灰色。

1. 第三列（lastColumn）应包含一个面板（candy crush grid），以允许用户移动糖果：

* 包括标签“Move：”和一个文本框，玩家可以在文本框输入应该移动糖果的坐标（例如，左上角的糖果为a1）。此时，你还不必添加点击事件，你将在下一步中添加。
* 在文本框下方，包括一个由四个箭头按钮（arrow button)组成的网格(grid)：向上(up)、向下(down)、向左(left)、和向右(right)。
* 在网格下方，添加一个名为“Crush Once”的按钮(button)。
* 在消消乐3中，你将实现糖果的拖曳。但是，在消消乐1和消消乐2中，你只需实现用按钮触发器和文本框来控制糖果移动。

**添加基本行为**

现在添加基本用户交互：

* 单击“New Game”按钮时，将生成一个新的游戏面板。
* 当用户单击向上/向下/向左/向右按钮（up/down/left/right buttons）时，程序读取文本框中的字段，如果根据规则移动有效（即移动后可以消去一些糖果），则移动相应的糖果。
* 若要测试移动是否有效，请调用rules.isMoveTypeValid方法。如果移动有效，请调用board.flipCandies方法，它会实际交换糖果位置。flipCandies方法会发出一个移动事件，因此你还需要实现“移动”事件侦听器。你可能会发现board.getCandyInDirection方法很有用。你不应该在这一步把糖果粉碎，你将在步骤4中实现糖果粉碎。
* 假设目前用户输入有效，我们将在下一步中实现输入检查。
* 用户单击其中一个按钮（上/下/左/右）后，文本框应被清除，光标应自动聚焦（focus）到文本框，以便用户可以在糖果粉碎后立即开始键入。

有关Board.js和Rules.js如何交互的更多信息:

Board是只保存状态的“哑”数据对象，Rules是不保存状态但知道一切如何变化的“智能”对象。因为所有的状态变化都要经过Board，任何想要知道Board变化的代码都可以监听Board事件。Rules对象拥有所有复杂的方法。Rules对象持有一个Board, Board不知道Rules。

**添加基本输入检查**

在此步骤中，你将添加基本功能来验证用户输入（input），并通过防止无效的用户输入来提高实用性。为此：

* 仅当用户在文本框中键入有效文本内容时才应启用向上/向下/向左/向右按钮（例如，a5有效，但ZZZ8是无效的，空文本框也是无效的）。比如，你可以选择将大写字母（A-H）视为有效的输入字符，但这不是必需的。如果输入无效，则应禁用这些按钮（向上/向下/向左/向右按钮），也就是这些按钮不能点击。
* 根据游戏规则，只启用表示有效移动的按钮：当翻转没有导致一排三个或更多相同颜色的糖果时，相应的上/下/左/右按钮也应该被禁用。你需要调用rules.js中的rules.isMoveTypeValid方法来确定四个潜在移动方向中每个方向的有效性。
* 要使启用的按钮更加醒目，请将其背景色更改为灰色以外的颜色，并在禁用时恢复为默认颜色。
* 一旦用户在文本框中输入了至少两个字符，则最好执行这些检查。

**Crush Once 按钮**

在这个版本的消消乐中，粉碎事件不会自动发生。要粉碎糖果，用户可以按下Crush Once 按钮（button），这会导致一次粉碎。

* Crush Once按钮应该只粉碎可立即粉碎的糖果。你需要多次单击它才能产生一连串的粉碎效应。
* 当你按下“Crush Once”按钮时，它应该做两件用户可以看到的事情：它应该移除粉碎的糖果，然后重新填充游戏面板。这些操作应按顺序进行，从粉碎糖果到向下移动糖果的时间间隔应为500ms。
* 你可能会发现以下方法很有用：rules.removeCrushes（它将移除rules.getCandyCrushes()允许消除的糖果）、setTimeout和rules.moveCandiesDown()。你需要在index.html中实现“粉碎糖果”监听器，该监听器由这些方法调用。
* 在这一步中，你不需要执行动画(animate)来平滑地向下移动糖果。你只需移除它们，等待0.5秒，然后重新填充面板。你将在实验2中实现平滑的粉碎动画。
* “Crush Once”按钮应该像箭头一样进行验证。当没有什么可粉碎的东西时，应该禁用它。当有东西要粉碎时，它应该被启用（此时箭头按钮应该被禁用）。与箭头按钮一样，“Crush Once”按钮在启用时应更改为灰色以外的其他颜色，在禁用时恢复为默认颜色。
* 重要的是，不能通过UI进行非法移动。特别是，在500ms的延迟期间，你不能允许用户在粉碎完成之前移动糖果。

**进一步（可选）改进**

在实验2和实验3 中，你将为游戏实现更多功能，包括移除粉碎的糖果和跟踪分数。如果你迫不及待地想了解更多，这里有一些改进的想法：

* 通过使文本框不区分大小写（a1 和 A1 都有效）并忽略空格来改进输入检查。
* 让玩家在文本框中输入一个名字，或者让玩家从上一个玩家的下拉菜单中选择一个名字。在标签中显示玩家名称。
* 用按键（可能是n键和空格键）触发“New Game”按钮和“Crush Once”按钮。
* 让你的页面对移动设备友好！检查以确保布局在多种屏幕尺寸上都能正确渲染，包括宽屏幕和窄屏幕的桌面浏览器以及大约360像素宽的智能手机。Google Chrome浏览器中的Web Inspector有一个很好的功能，可以模拟各种设备的视图。如果你找不到，就用Google搜索一下！思考使你的页面具有响应性，例如，自动切换到小屏幕的纵向布局。

1. **程序设计**
2. 游戏初始化逻辑和相关实用函数：
3. // #2
4. // Final initialization entry point: the Javascript code inside this block
5. // runs at the end of start-up when the page has finished loading.
6. $(document).ready(function () {
7. createNewTable();
8. originalButtonColors.up = $('#upButton').css('backgroundColor');
9. originalButtonColors.down = $('#downButton').css('backgroundColor');
10. originalButtonColors.left = $('#leftButton').css('backgroundColor');
11. originalButtonColors.right = $('#rightButton').css('backgroundColor');
12. originalButtonColors.crush = $('#crushOnceButton').css('backgroundColor');
13. disableButtons(); // 禁用按钮
14. // 监听输入框的输入事件
15. $('#moveInput').on('input', function () {
16. // 获取输入框的值并去除首尾空格
17. let input = $(this).val().trim();
19. // 如果输入长度至少为两个字符，执行检查操作
20. if (input.length **>**= 2) {
21. validateAndEnableButtons(); // 执行检查和按钮启用操作
22. } else {
23. disableButtons(); // 否则禁用按钮
24. }
25. });
26. });
28. // #3
29. //创建游戏板
30. function createNewTable() {
31. board.clear();
32. rules.prepareNewGame();
33. let table = document.getElementById("table");
34. table.innerHTML = '';
35. for (var i = 0; i **<** **DEFAULT\_BOARD\_SIZE**; i++) {
36. let tr = document.createElement('tr');
37. for (var j = 0; j **<** **DEFAULT\_BOARD\_SIZE**; j++) {
38. let td = document.createElement('td');
39. td.style.backgroundColor = board.square[i][j].color;
40. td.style.border = "1px solid white";
41. // 设置单元格的ID，按照格式 cell\_row\_col
42. td.id = 'cell\_' + i + '\_' + j;
44. // 创建span元素来显示文本坐标
45. let span = document.createElement('span');
46. span.textContent = String.fromCharCode(97 + j) + (i + 1); // 转换为a1, b2等
47. span.style.color = (board.square[i][j].color === 'yellow') ? 'gray' : 'white';
48. span.style.position = 'absolute'; // 确保span在单元格中心
49. span.style.top = '50%';
50. span.style.left = '50%';
51. span.style.transform = 'translate(-50%, -50%)';
52. td.style.position = 'relative'; // 确保td是相对定位，以便span能正确定位
54. td.appendChild(span);
55. tr.appendChild(td);
56. }
57. table.appendChild(tr);
58. }
59. }
61. //获取用户输入的坐标，例如 'a1', 'b2'
62. function getTargetObject() {
63. // 获取文本框中的输入值
64. let input = $('#moveInput').val().trim();
65. // 如果输入为空，则直接返回，不执行后续操作
66. if (!input) {
67. console.error('文本框中没有输入坐标');
68. return null;
69. }
71. // 解析输入的坐标，例如 'a1', 'b2'
72. let column = input.charAt(0).toLowerCase();
73. let row = parseInt(input.slice(1), 10);
74. if (!isValidInput(input)) {
75. console.error('输入的坐标无效');
76. return null;
77. }
79. // 计算当前位置的坐标，转换为数组索引格式（从0开始）
80. let x = row - 1;
81. let y = column.charCodeAt(0) - 97;
82. return {x: x, y: y};
83. // 这里使用了ASCII码进行字符转换：
84. // - 'a'.charCodeAt(0) 获取字符 'a' 的ASCII码值，即 97
85. // - column.charCodeAt(0) 获取输入列字符的ASCII码值，例如输入 'a'，则 column.charCodeAt(0) 为 97
86. // - 相减后得到该列在数组中的索引位置，例如 'a' 对应索引 0，'b' 对应索引 1，以此类推
88. // parseInt(row) - 1 是为了将输入的行号转换为数组索引格式，例如输入 '1'，则转换为索引 0
90. // currentPosition 现在包含了移动操作的起始位置，可以在接下来的代码中使用
91. }
93. // 根据坐标获取Candy对象
94. function getCandyByIndex({x, y}) {
95. // 确保索引在有效范围内
96. if (x **>**= 0 && x **<** **DEFAULT\_BOARD\_SIZE** && y **>**= 0 && y **<** **DEFAULT\_BOARD\_SIZE**) {
97. return board.square[x][y];
98. } else {
99. console.error("输入的坐标超出范围 " + x + "," + y);
100. return null;
101. }
102. }
104. //清空文本输入框并设置焦点并禁用按钮
105. function clearAndFocusTextBox() {
106. let textBox = $('#moveInput');
107. textBox.val('');
108. textBox.focus();
109. disableButtons(); // 禁用按钮
110. }
112. //实现视觉上的移动单元格
113. function moveCandyOnBoard(candy, fromRow, fromCol, toRow, toCol) {
114. var fromCellId = 'cell\_' + fromRow + '\_' + fromCol;  // 获取起始单元格的ID
115. var toCellId = 'cell\_' + toRow + '\_' + toCol;        // 获取目标单元格的ID
116. console.log(fromCellId, toCellId);
117. var fromCell = document.getElementById(fromCellId);  // 获取起始单元格的DOM元素
118. var toCell = document.getElementById(toCellId);      // 获取目标单元格的DOM元素
119. console.log(fromCell, toCell);
120. if (fromCell && toCell) {
121. var tempColor = fromCell.style.backgroundColor;    // 临时存储起始单元格的背景颜色
122. fromCell.style.backgroundColor = toCell.style.backgroundColor;  // 将目标单元格的背景颜色赋给起始单元格
123. toCell.style.backgroundColor = tempColor;          // 将临时存储的背景颜色赋给目标单元格
124. // 更新起始单元格的文字颜色
125. fromCell.querySelector('span').style.color = (fromCell.style.backgroundColor === 'yellow') ? 'gray' : 'white';
126. // 更新目标单元格的文字颜色
127. toCell.querySelector('span').style.color = (toCell.style.backgroundColor === 'yellow') ? 'gray' : 'white';
128. } else {
129. console.error('未找到单元格ID:', fromCellId, toCellId);  // 如果未找到单元格ID，则输出错误信息到控制台
130. }
131. }
133. // 禁用按钮的函数
134. function disableButtons() {
135. disableButton('#upButton');
136. disableButton('#downButton');
137. disableButton('#leftButton');
138. disableButton('#rightButton');
139. }
141. //验证输入是否为有效坐标
142. function isValidInput(input) {
143. let column = input.charAt(0).toLowerCase();
144. let row = parseInt(input.slice(1), 10);
145. return !isNaN(row) && row **>**= 1 && row **<**= DEFAULT\_BOARD\_SIZE && column **>**= 'a' && column **<** **String.fromCharCode**(97 + DEFAULT\_BOARD\_SIZE);
146. }
148. // 启用按钮并修改背景色
149. function enableButton(buttonId, originalColor) {
150. $(buttonId).prop('disabled', false);
151. $(buttonId).css('backgroundColor', '#ffff00'); // 修改为启用时的颜色
152. }
154. // 禁用按钮并恢复原始背景色
155. function disableButton(buttonId) {
156. $(buttonId).prop('disabled', true);
157. // 恢复为禁用前的原始颜色
158. switch (buttonId) {
159. case '#upButton':
160. $(buttonId).css('backgroundColor', originalButtonColors.up);
161. break;
162. case '#downButton':
163. $(buttonId).css('backgroundColor', originalButtonColors.down);
164. break;
165. case '#leftButton':
166. $(buttonId).css('backgroundColor', originalButtonColors.left);
167. break;
168. case '#rightButton':
169. $(buttonId).css('backgroundColor', originalButtonColors.right);
170. break;
171. case '#crushOnceButton':
172. $(buttonId).css('backgroundColor', originalButtonColors.crush);
173. default:
174. break;
175. }
176. }
178. // 验证输入并根据结果启用或禁用按钮的函数
179. function validateAndEnableButtons() {
180. let input = $('#moveInput').val().trim();
182. // 如果输入为空或无效
183. if (!input || !isValidInput(input)) {
184. console.error('输入的坐标无效');
185. disableButtons(); // 禁用按钮
186. } else {
187. let targetObject = getTargetObject();
188. let initCandy = getCandyByIndex(targetObject);
189. if (rules.isMoveTypeValid(initCandy, 'up')) {
190. enableButton('#upButton', originalButtonColors.up);
191. } else {
192. disableButton('#upButton');
193. }
194. if (rules.isMoveTypeValid(initCandy, 'down')) {
195. enableButton('#downButton', originalButtonColors.down);
196. } else {
197. disableButton('#downButton');
198. }
199. if (rules.isMoveTypeValid(initCandy, 'left')) {
200. enableButton('#leftButton', originalButtonColors.left);
201. } else {
202. disableButton('#leftButton');
203. }
204. if (rules.isMoveTypeValid(initCandy, 'right')) {
205. enableButton('#rightButton', originalButtonColors.right);
206. } else {
207. disableButton('#rightButton');
208. }
210. }
211. }
213. // 判断CrushOnceButton是否应该禁用
214. function validateCrushOnceButton() {
215. let crushes = rules.getCandyCrushes(false);
216. if (crushes.length **>** 0) {
217. console.log(crushes);
218. enableButton('#crushOnceButton', originalButtonColors.crush);
219. } else {
220. disableButton('#crushOnceButton');
221. }
222. }

225. function crushOnce() {
226. disableButtons(); // 禁用所有按钮以防止用户在500ms延迟期间进行非法移动
227. let crushes = rules.getCandyCrushes(false);// 获取所有可以粉碎的糖果
228. if (crushes.length **>** 0) {
229. rules.removeCrushes(crushes, false);// 触发remove事件，移除可以粉碎的糖果
230. } else {
231. console.error('没有可以粉碎的糖果');
232. validateAndEnableButtons(); // 没有可以粉碎的糖果时，重新启用有效的按钮
233. }
235. }
236. 事件监听器实现：
237. /\* Event Handlers \*/
238. // access the candy object with info.candy
240. // #4
241. // add a candy to the board
242. $(board).on('add', function (e, info) {
243. var cellId = 'cell\_' + info.toRow + '\_' + info.toCol;
244. var cell = document.getElementById(cellId);
246. if (cell) {
247. cell.style.backgroundColor = info.candy.color;
248. cell.querySelector('span').style.color = (info.candy.color === 'yellow') ? 'gray' : 'white';
249. }
250. });
252. // #5
253. // move a candy on the board
254. $(board).on('move', function (e, info) {
255. var candy = info.candy;
256. console.log(candy);
257. var fromRow = info.fromRow;
258. var fromCol = info.fromCol;
259. var toRow = info.toRow;
260. var toCol = info.toCol;
261. moveCandyOnBoard(candy, fromRow, fromCol, toRow, toCol);
262. validateCrushOnceButton();
263. });
265. // #6
266. // remove a candy from the board
267. $(board).on('remove', function (e, info) {
268. var cellId = 'cell\_' + info.fromRow + '\_' + info.fromCol;
269. var cell = document.getElementById(cellId);
271. if (cell) {
272. cell.style.backgroundColor = '';  // 清空背景颜色
273. }
275. setTimeout(function () {
276. rules.moveCandiesDown();
277. validateCrushOnceButton();
278. }, 500);
279. });
281. // #7
282. // move a candy on the board
283. $(board).on('scoreUpdate', function (e, info) {
284. // Your code here. To be implemented in pset 2.
285. });
287. // Button Events
288. $(document).on('click', "#newGameButton", function (evt) {
289. // 创建全新游戏板
290. createNewTable();
291. });
293. $(document).on('click', "#upButton", function (evt) {
294. let targetObject = getTargetObject();
295. let initCandy = getCandyByIndex(targetObject);
296. let targetCandy = getCandyByIndex({x: targetObject.x - 1, y: targetObject.y});
297. if (initCandy != null) {
298. if (rules.isMoveTypeValid(initCandy, 'up')) {
299. board.flipCandies(initCandy, targetCandy);
300. }
301. clearAndFocusTextBox();
302. }
303. });
305. $(document).on('click', "#downButton", function (evt) {
306. let targetObject = getTargetObject();
307. let initCandy = getCandyByIndex(targetObject);
308. let targetCandy = getCandyByIndex({x: targetObject.x + 1, y: targetObject.y});
309. if (initCandy != null) {
310. if (rules.isMoveTypeValid(initCandy, 'down')) {
311. board.flipCandies(initCandy, targetCandy);
312. }
313. clearAndFocusTextBox();
314. }
315. });
317. $(document).on('click', "#leftButton", function (evt) {
318. let targetObject = getTargetObject();
319. let initCandy = getCandyByIndex(targetObject);
320. let targetCandy = getCandyByIndex({x: targetObject.x, y: targetObject.y - 1});
321. if (initCandy != null) {
322. if (rules.isMoveTypeValid(initCandy, 'left')) {
323. board.flipCandies(initCandy, targetCandy);
324. }
325. clearAndFocusTextBox();
326. }
328. });
330. // #8
331. $(document).on('click', "#rightButton", function (evt) {
332. let targetObject = getTargetObject();
333. let initCandy = getCandyByIndex(targetObject);
334. let targetCandy = getCandyByIndex({x: targetObject.x, y: targetObject.y + 1});
335. if (initCandy != null) {
336. if (rules.isMoveTypeValid(initCandy, 'right')) {
337. board.flipCandies(initCandy, targetCandy);
338. }
339. clearAndFocusTextBox();
340. }
341. });
343. // #9
344. $(document).on('click', "#crushOnceButton", function (evt) {
345. if (!$(this).prop('disabled')) { // 确保按钮未禁用
346. crushOnce();
347. }
348. });

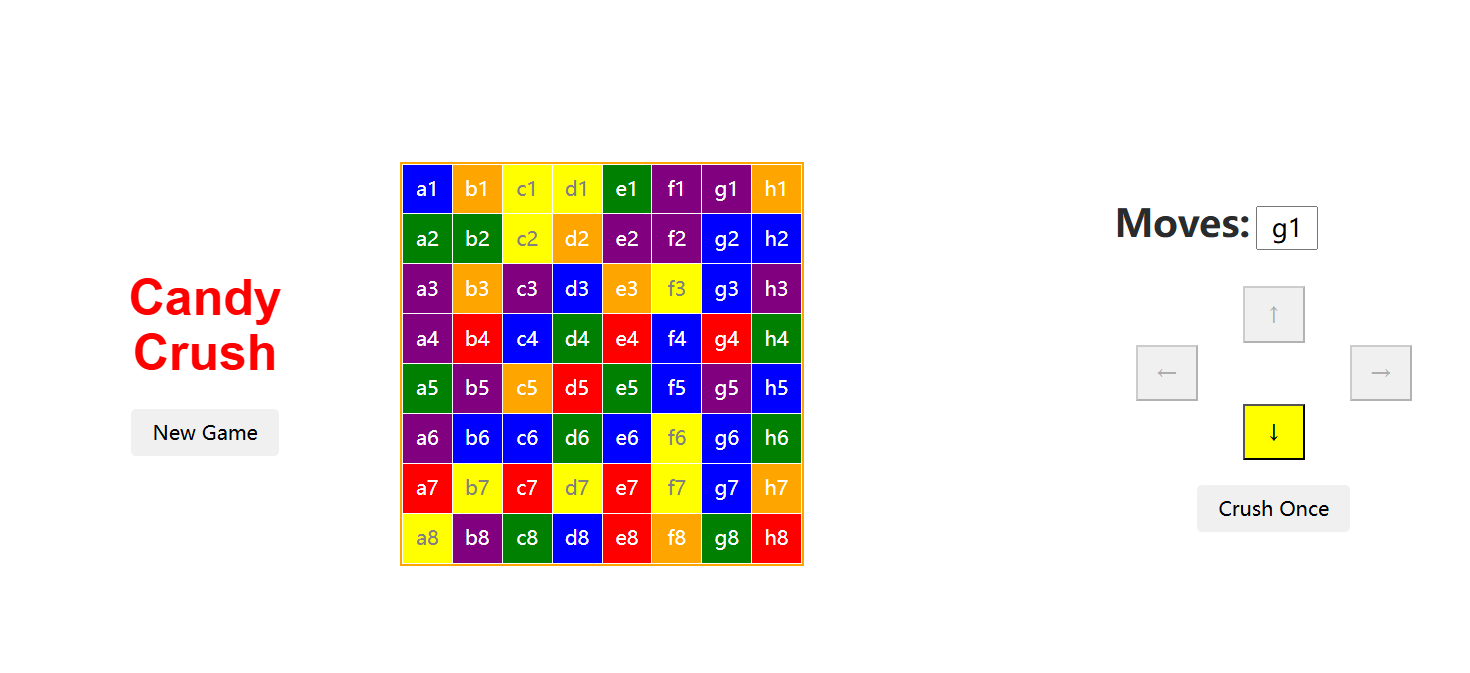
351. // keyboard events arrive here
352. $(document).on('keydown', function (evt) {
354. });
355. 游戏页面结构HTML实现代码：
356. **<body>**
357. **<div** class="container"**>**
358. <!-- justify-content-center 水平居中列中元素 align-items-center 垂直居中列中元素 -->
359. **<div** class="row justify-content-center align-items-center" style="height: 100vh;"**>**
360. **<div** class="col-md-3 text-center" id="firstColumn"**>**
361. **<div>**
362. <!-- Column 1 Code Here -->
363. **<h1** style="font-family: Arial ;font-weight: bold;color: #FF0000;"**>**Candy**</br>**Crush**</h1>**
364. **<input** type="button" class="btn" id="newGameButton" value="New Game" style="margin-top: 15px;"**/>**
366. **</div>**
367. **</div>**
369. **<div** class="col-md-6" id="mainColumn"**>**
371. <!-- Column 2 Code Here -->
372. **<table** id="table" style="width: 320px; height: 320px;"**>**
373. **</table>**
375. **</div>**
377. **<div** class="col-md-3" id="lastColumn"**>**
379. <!-- Column 3 Code Here -->
380. **<div** id="move"**>**
381. **<label** for="moveInput" class="move-lable"**>**Moves:**</label>**
382. **<input** type="text" id="moveInput" placeholder="a1" class="move-text"**>**
383. **</div>**
384. **<div** class="arrow-grid"**>**
385. **<button** id="leftButton"**>**←**</button>**
386. **<button** id="upButton"**>**↑**</button>**
387. **<button** id="rightButton"**>**→**</button>**
388. **<button** id="downButton"**>**↓**</button>**
389. **</div>**
390. **<div** style="display: flex; justify-content: center;"**>**
391. **<button** class="btn" id="crushOnceButton"**>**Crush Once**</button>**
392. **</div>**
394. **</div>**
395. **</div>** <!-- class="row" -->
396. **</div>** <!-- class="container" -->
397. **</body>**
398. 样式设计CSS代码：
399. /\* Copyright (c) 2017 MIT 6.813/6.831 course staff, all rights reserved.
400. \* Redistribution of original or derived work requires permission of course staff.
401. \*/
402. #mainColumn {
403. **display**: flex;
404. flex-wrap: wrap;
405. **height**: 320px;
407. }
408. #table {
409. **outline**: 2px solid orange;
410. }
411. .move-lable {
412. font-size : 30px;
413. **font-weight**: bold;
415. }
416. .move-text {
417. font-size : 20px;
418. **width**: 50px;
419. **text-align**: center;
420. }
421. .arrow-grid {
422. **display**: grid;
423. grid-template-columns: repeat(3, 1fr);        /\*grid-template-columns: 这是一个CSS属性，用于定义网格容器中列的大小和数量。\*/
424. gap: 2px;                                     /\*gap属性用于定义网格容器中的列和行之间的间距。\*/
425. **margin-top**: 20px;
426. justify-items: center;                        /\* 中心对齐箭头按钮 \*/
427. }
428. .arrow-grid button {
429. **width**: 50px;
430. **height**: 45px;
431. }
432. .arrow-grid button:nth-child(2) {               /\* Up button \*/
433. grid-column: 2 / 3;
434. grid-row: 1 / 2;
435. }
436. .arrow-grid button:nth-child(4) {               /\* Down button 开始于第二行，结束于第三行 \*/
437. grid-column: 2 / 3;
438. grid-row: 3 / 4;
439. }
440. .arrow-grid button:nth-child(1) {               /\* Left button \*/
441. grid-column: 1 / 2;
442. grid-row: 2 / 3;
443. }
444. .arrow-grid button:nth-child(3) {               /\* Right button \*/
445. grid-column: 3 / 4;
446. grid-row: 2 / 3;
447. }
448. .btn{
449. **margin-top**: 20px;
450. }
451. **程序详解及运行结果**

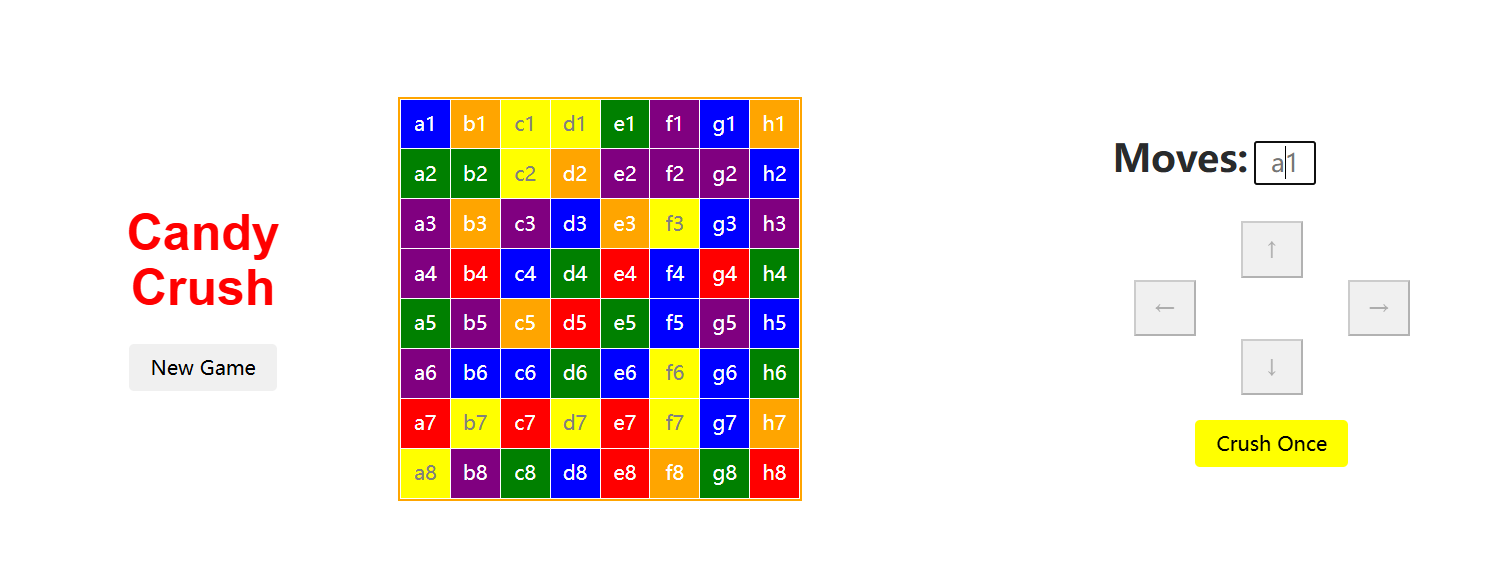
程序详解：

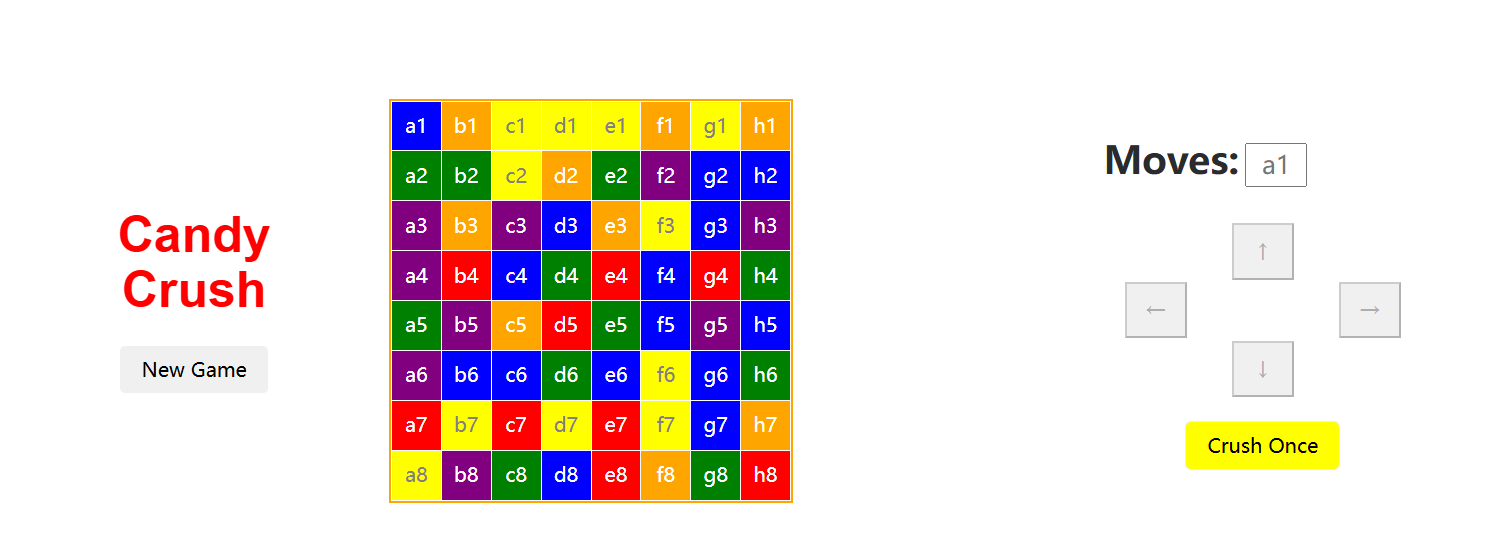
1. 定义了一些全局变量来保存按钮的原始颜色以及禁用时的颜色：

1. let originalButtonColors = {
2. up: **null**,
3. down: **null**,
4. left: **null**,
5. right: **null**,
6. crush: **null**
7. };
8. 在文档加载完成后，创建了一个新的游戏板，保存了按钮的原始颜色，并禁用了这些按钮。你还添加了一个事件监听器来监听输入框的输入事件。如果输入长度至少为两个字符，就会执行检查操作并启用按钮；否则，按钮将被禁用。
9. $(document).ready(**function** () {
10. createNewTable();
11. originalButtonColors.up = $('#upButton').css('backgroundColor');
12. originalButtonColors.down = $('#downButton').css('backgroundColor');
13. originalButtonColors.left = $('#leftButton').css('backgroundColor');
14. originalButtonColors.right = $('#rightButton').css('backgroundColor');
15. originalButtonColors.crush = $('#crushOnceButton').css('backgroundColor');
16. disableButtons(); // 禁用按钮
17. // 监听输入框的输入事件
18. $('#moveInput').on('input', **function** () {
19. // 获取输入框的值并去除首尾空格
20. let input = $(**this**).val().trim();
21. // 如果输入长度至少为两个字符，执行检查操作
22. **if** (input.length >= 2) {
23. validateAndEnableButtons(); // 执行检查和按钮启用操作
24. } **else** {
25. disableButtons(); // 否则禁用按钮
26. }
27. });
28. });
29. 实现了一个 createNewTable 函数来创建游戏板，其中包括了清空游戏板、准备新游戏、创建游戏板的 HTML 表格等操作。
30. 为 add 事件添加了事件处理器，当糖果被添加到游戏板时，会更新相应单元格的背景颜色。
31. $(board).on('add', **function** (e, info) {
32. **var** cellId = 'cell\_' + info.toRow + '\_' + info.toCol;
33. **var** cell = document.getElementById(cellId);
34. **if** (cell) {
35. cell.style.backgroundColor = info.candy.color;
36. cell.querySelector('span').style.color = (info.candy.color === 'yellow') ? 'gray' : 'white';
37. }
38. });
39. move事件处理器，当糖果在游戏板上移动时，会更新相应单元格的背景颜色，并检查是否有可粉碎的糖果。（代码参考上一部分）
40. remove事件处理器，当糖果从游戏板上移除时，会清空相应单元格的背景颜色，并在一段延迟后将糖果下移，并检查是否有可粉碎的糖果。（代码参考上一部分）
41. rightButton 点击事件处理器，当点击这个按钮时，会将选中的糖果向右移动。（代码参考上一部分）
42. crushOnceButton 点击事件处理器，当点击这个按钮时，会粉碎一次糖果。（代码参考上一部分）

运行结果截图：

****

****

****

1. **心得体会**

在设计和开发消消乐游戏的过程中，我积累了许多的前端开发和UI设计的经验。以下是我的一些领悟：

在这个项目中，我掌握了HTML、CSS和JavaScript的基础应用。尤其是JavaScript的事件处理、DOM操作和动画效果的实现，赋予我更强大的页面元素控制和交互能力。实现消消乐游戏逻辑无疑是这个项目的核心挑战之一。从处理游戏棋盘的设计到识别和处理糖果的匹配规则，我学习并掌握了如何设计和实现复杂的游戏算法。这包括糖果的移动、消除、下落等过程的编程实现，以及游戏状态的管理和更新。

同时，我领悟到了用户体验的重要性：在设计用户界面和交互过程中，我尤其注重用户体验的细节。例如，通过动态更新按钮状态、保证游戏操作的流畅性以及提供清晰的反馈信息，从而提升了玩家的整体游戏体验。这个过程让我深刻理解，在设计任何应用程序时，用户体验始终应该是首要考虑的因素。

这个项目还让我认识到前端开发是一个持续学习和改进的过程。随着技术的不断进步和用户需求的变化，我需要保持对新知识和技能的学习，以保持自己的竞争力并不断提升自己的开发能力。

前端技术不是我课上的唯一收获，打个比方程序是一个机器，每个版本的机器对应一个状态。每加一行代码进去，这个机器都会变化，那么对应的这个状态也就变化了。“写一行测一行”保证我们知道这个状态变化是哪部分代码带来的，这是一种有效的代码实践方式。

通过这个消消乐项目，我不仅获得了宝贵的技术经验，还锻炼了解决问题的一种通用的能力和思想。这些经历将成为我未来学习和发展的重要基石，我期待能够将这些经验和技能应用到更多挑战中。