```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class AI_Ell1_FindSpot : MonoBehaviour
6  {
7      private bool downed = false;
8      private bool checkedRow1 = false;
9      private bool checkedRow2 = false;
10     private bool checkedRow3 = false;
11     private bool checkedRow4 = false;
12     private bool checkedRow5 = false;
13     private bool checkedRow6 = false;
14     private bool checkedRow7 = false;
15     private bool checkedRow8 = false;
16     private bool checkedRow9 = false;
17     private bool checkedRow10 = false;
18     private bool checkedRow11 = false;
19     private bool checkedRow12 = false;
20     private bool checkedRow13 = false;
21     private bool checkedRow14 = false;
22     private bool checkedRow15 = false;
23     private bool checkedRow16 = false;
24     private bool checkedRow17 = false;
25     private bool checkedRow18 = false;
26     private bool checkedRow19 = false;
27     private bool checkedRow20 = false;
28
29     private int chosenRot;
30     private int myRot;
31     private int mySpot = 0;
32     private int spot;
33
34     private float timerTime;
35
36     private Vector3 cubeQuart = new Vector3(0.25f, 0.25f, 0.25f);
37
38     private List<Vector3> row1 = new List<Vector3>();
39     private List<Vector3> row2 = new List<Vector3>();
40     private List<Vector3> row3 = new List<Vector3>();
41     private List<Vector3> row4 = new List<Vector3>();
42     private List<Vector3> row5 = new List<Vector3>();
43     private List<Vector3> row6 = new List<Vector3>();
44     private List<Vector3> row7 = new List<Vector3>();
45     private List<Vector3> row8 = new List<Vector3>();
46     private List<Vector3> row9 = new List<Vector3>();
47     private List<Vector3> row10 = new List<Vector3>();
48     private List<Vector3> row11 = new List<Vector3>();
49     private List<Vector3> row12 = new List<Vector3>();
```

```
50      private List<Vector3> row13 = new List<Vector3>();
51      private List<Vector3> row14 = new List<Vector3>();
52      private List<Vector3> row15 = new List<Vector3>();
53      private List<Vector3> row16 = new List<Vector3>();
54      private List<Vector3> row17 = new List<Vector3>();
55      private List<Vector3> row18 = new List<Vector3>();
56      private List<Vector3> row19 = new List<Vector3>();
57      private List<Vector3> row20 = new List<Vector3>();
58      private List<Vector3> row1Vacancies = new List<Vector3>();
59      private List<Vector3> row2Vacancies = new List<Vector3>();
60      private List<Vector3> row3Vacancies = new List<Vector3>();
61      private List<Vector3> row4Vacancies = new List<Vector3>();
62      private List<Vector3> row5Vacancies = new List<Vector3>();
63      private List<Vector3> row6Vacancies = new List<Vector3>();
64      private List<Vector3> row7Vacancies = new List<Vector3>();
65      private List<Vector3> row8Vacancies = new List<Vector3>();
66      private List<Vector3> row9Vacancies = new List<Vector3>();
67      private List<Vector3> row10Vacancies = new List<Vector3>();
68      private List<Vector3> row11Vacancies = new List<Vector3>();
69      private List<Vector3> row12Vacancies = new List<Vector3>();
70      private List<Vector3> row13Vacancies = new List<Vector3>();
71      private List<Vector3> row14Vacancies = new List<Vector3>();
72      private List<Vector3> row15Vacancies = new List<Vector3>();
73      private List<Vector3> row16Vacancies = new List<Vector3>();
74      private List<Vector3> row17Vacancies = new List<Vector3>();
75      private List<Vector3> row18Vacancies = new List<Vector3>();
76      private List<Vector3> row19Vacancies = new List<Vector3>();
77      private List<Vector3> row20Vacancies = new List<Vector3>();
78      private List<int> row1VacancyInts = new List<int>();
79      private List<int> row2VacancyInts = new List<int>();
80      private List<int> row3VacancyInts = new List<int>();
81      private List<int> row4VacancyInts = new List<int>();
82      private List<int> row5VacancyInts = new List<int>();
83      private List<int> row6VacancyInts = new List<int>();
84      private List<int> row7VacancyInts = new List<int>();
85      private List<int> row8VacancyInts = new List<int>();
86      private List<int> row9VacancyInts = new List<int>();
87      private List<int> row10VacancyInts = new List<int>();
88      private List<int> row11VacancyInts = new List<int>();
89      private List<int> row12VacancyInts = new List<int>();
90      private List<int> row13VacancyInts = new List<int>();
91      private List<int> row14VacancyInts = new List<int>();
92      private List<int> row15VacancyInts = new List<int>();
93      private List<int> row16VacancyInts = new List<int>();
94      private List<int> row17VacancyInts = new List<int>();
95      private List<int> row18VacancyInts = new List<int>();
96      private List<int> row19VacancyInts = new List<int>();
97      private List<int> row20VacancyInts = new List<int>();
98      private List<int> openSpots = new List<int>();
```

```
 99      private List<int> preferredSpots = new List<int>();
100
101      private Vector3 boxA1 = new Vector3(0.5f, 0.3f, -3.6f);
102      private Vector3 boxB1 = new Vector3(0.5f, 0.3f, -2.6f);
103      private Vector3 boxC1 = new Vector3(0.5f, 0.3f, -1.6f);
104      private Vector3 boxD1 = new Vector3(0.5f, 0.3f, -0.6f);
105      private Vector3 boxE1 = new Vector3(0.5f, 0.3f, 0.3f);
106      private Vector3 boxF1 = new Vector3(0.5f, 0.3f, 1.3f);
107      private Vector3 boxG1 = new Vector3(0.5f, 0.3f, 2.3f);
108      private Vector3 boxH1 = new Vector3(0.5f, 0.3f, 3.3f);
109      private Vector3 boxI1 = new Vector3(0.5f, 0.3f, 4.3f);
110      private Vector3 boxJ1 = new Vector3(0.5f, 0.3f, 5.3f);
111
112      private Vector3 boxA2 = new Vector3(0.5f, 1.3f, -3.6f);
113      private Vector3 boxB2 = new Vector3(0.5f, 1.3f, -2.6f);
114      private Vector3 boxC2 = new Vector3(0.5f, 1.3f, -1.6f);
115      private Vector3 boxD2 = new Vector3(0.5f, 1.3f, -0.6f);
116      private Vector3 boxE2 = new Vector3(0.5f, 1.3f, 0.3f);
117      private Vector3 boxF2 = new Vector3(0.5f, 1.3f, 1.3f);
118      private Vector3 boxG2 = new Vector3(0.5f, 1.3f, 2.3f);
119      private Vector3 boxH2 = new Vector3(0.5f, 1.3f, 3.3f);
120      private Vector3 boxI2 = new Vector3(0.5f, 1.3f, 4.3f);
121      private Vector3 boxJ2 = new Vector3(0.5f, 1.3f, 5.3f);
122
123      private Vector3 boxA3 = new Vector3(0.5f, 2.3f, -3.6f);
124      private Vector3 boxB3 = new Vector3(0.5f, 2.3f, -2.6f);
125      private Vector3 boxC3 = new Vector3(0.5f, 2.3f, -1.6f);
126      private Vector3 boxD3 = new Vector3(0.5f, 2.3f, -0.6f);
127      private Vector3 boxE3 = new Vector3(0.5f, 2.3f, 0.3f);
128      private Vector3 boxF3 = new Vector3(0.5f, 2.3f, 1.3f);
129      private Vector3 boxG3 = new Vector3(0.5f, 2.3f, 2.3f);
130      private Vector3 boxH3 = new Vector3(0.5f, 2.3f, 3.3f);
131      private Vector3 boxI3 = new Vector3(0.5f, 2.3f, 4.3f);
132      private Vector3 boxJ3 = new Vector3(0.5f, 2.3f, 5.3f);
133
134      private Vector3 boxA4 = new Vector3(0.5f, 3.3f, -3.6f);
135      private Vector3 boxB4 = new Vector3(0.5f, 3.3f, -2.6f);
136      private Vector3 boxC4 = new Vector3(0.5f, 3.3f, -1.6f);
137      private Vector3 boxD4 = new Vector3(0.5f, 3.3f, -0.6f);
138      private Vector3 boxE4 = new Vector3(0.5f, 3.3f, 0.3f);
139      private Vector3 boxF4 = new Vector3(0.5f, 3.3f, 1.3f);
140      private Vector3 boxG4 = new Vector3(0.5f, 3.3f, 2.3f);
141      private Vector3 boxH4 = new Vector3(0.5f, 3.3f, 3.3f);
142      private Vector3 boxI4 = new Vector3(0.5f, 3.3f, 4.3f);
143      private Vector3 boxJ4 = new Vector3(0.5f, 3.3f, 5.3f);
144
145      private Vector3 boxA5 = new Vector3(0.5f, 4.3f, -3.6f);
146      private Vector3 boxB5 = new Vector3(0.5f, 4.3f, -2.6f);
147      private Vector3 boxC5 = new Vector3(0.5f, 4.3f, -1.6f);
```

```
148     private Vector3 boxD5 = new Vector3(0.5f, 4.3f, -0.6f);
149     private Vector3 boxE5 = new Vector3(0.5f, 4.3f, 0.3f);
150     private Vector3 boxF5 = new Vector3(0.5f, 4.3f, 1.3f);
151     private Vector3 boxG5 = new Vector3(0.5f, 4.3f, 2.3f);
152     private Vector3 boxH5 = new Vector3(0.5f, 4.3f, 3.3f);
153     private Vector3 boxI5 = new Vector3(0.5f, 4.3f, 4.3f);
154     private Vector3 boxJ5 = new Vector3(0.5f, 4.3f, 5.3f);
155
156     private Vector3 boxA6 = new Vector3(0.5f, 5.3f, -3.6f);
157     private Vector3 boxB6 = new Vector3(0.5f, 5.3f, -2.6f);
158     private Vector3 boxC6 = new Vector3(0.5f, 5.3f, -1.6f);
159     private Vector3 boxD6 = new Vector3(0.5f, 5.3f, -0.6f);
160     private Vector3 boxE6 = new Vector3(0.5f, 5.3f, 0.3f);
161     private Vector3 boxF6 = new Vector3(0.5f, 5.3f, 1.3f);
162     private Vector3 boxG6 = new Vector3(0.5f, 5.3f, 2.3f);
163     private Vector3 boxH6 = new Vector3(0.5f, 5.3f, 3.3f);
164     private Vector3 boxI6 = new Vector3(0.5f, 5.3f, 4.3f);
165     private Vector3 boxJ6 = new Vector3(0.5f, 5.3f, 5.3f);
166
167     private Vector3 boxA7 = new Vector3(0.5f, 6.3f, -3.6f);
168     private Vector3 boxB7 = new Vector3(0.5f, 6.3f, -2.6f);
169     private Vector3 boxC7 = new Vector3(0.5f, 6.3f, -1.6f);
170     private Vector3 boxD7 = new Vector3(0.5f, 6.3f, -0.6f);
171     private Vector3 boxE7 = new Vector3(0.5f, 6.3f, 0.3f);
172     private Vector3 boxF7 = new Vector3(0.5f, 6.3f, 1.3f);
173     private Vector3 boxG7 = new Vector3(0.5f, 6.3f, 2.3f);
174     private Vector3 boxH7 = new Vector3(0.5f, 6.3f, 3.3f);
175     private Vector3 boxI7 = new Vector3(0.5f, 6.3f, 4.3f);
176     private Vector3 boxJ7 = new Vector3(0.5f, 6.3f, 5.3f);
177
178     private Vector3 boxA8 = new Vector3(0.5f, 7.3f, -3.6f);
179     private Vector3 boxB8 = new Vector3(0.5f, 7.3f, -2.6f);
180     private Vector3 boxC8 = new Vector3(0.5f, 7.3f, -1.6f);
181     private Vector3 boxD8 = new Vector3(0.5f, 7.3f, -0.6f);
182     private Vector3 boxE8 = new Vector3(0.5f, 7.3f, 0.3f);
183     private Vector3 boxF8 = new Vector3(0.5f, 7.3f, 1.3f);
184     private Vector3 boxG8 = new Vector3(0.5f, 7.3f, 2.3f);
185     private Vector3 boxH8 = new Vector3(0.5f, 7.3f, 3.3f);
186     private Vector3 boxI8 = new Vector3(0.5f, 7.3f, 4.3f);
187     private Vector3 boxJ8 = new Vector3(0.5f, 7.3f, 5.3f);
188
189     private Vector3 boxA9 = new Vector3(0.5f, 8.3f, -3.6f);
190     private Vector3 boxB9 = new Vector3(0.5f, 8.3f, -2.6f);
191     private Vector3 boxC9 = new Vector3(0.5f, 8.3f, -1.6f);
192     private Vector3 boxD9 = new Vector3(0.5f, 8.3f, -0.6f);
193     private Vector3 boxE9 = new Vector3(0.5f, 8.3f, 0.3f);
194     private Vector3 boxF9 = new Vector3(0.5f, 8.3f, 1.3f);
195     private Vector3 boxG9 = new Vector3(0.5f, 8.3f, 2.3f);
196     private Vector3 boxH9 = new Vector3(0.5f, 8.3f, 3.3f);
```

```
197     private Vector3 boxI9 = new Vector3(0.5f, 8.3f, 4.3f);
198     private Vector3 boxJ9 = new Vector3(0.5f, 8.3f, 5.3f);
199
200     private Vector3 boxA10 = new Vector3(0.5f, 9.3f, -3.6f);
201     private Vector3 boxB10 = new Vector3(0.5f, 9.3f, -2.6f);
202     private Vector3 boxC10 = new Vector3(0.5f, 9.3f, -1.6f);
203     private Vector3 boxD10 = new Vector3(0.5f, 9.3f, -0.6f);
204     private Vector3 boxE10 = new Vector3(0.5f, 9.3f, 0.3f);
205     private Vector3 boxF10 = new Vector3(0.5f, 9.3f, 1.3f);
206     private Vector3 boxG10 = new Vector3(0.5f, 9.3f, 2.3f);
207     private Vector3 boxH10 = new Vector3(0.5f, 9.3f, 3.3f);
208     private Vector3 boxI10 = new Vector3(0.5f, 9.3f, 4.3f);
209     private Vector3 boxJ10 = new Vector3(0.5f, 9.3f, 5.3f);
210
211     private Vector3 boxA11 = new Vector3(0.5f, 10.3f, -3.6f);
212     private Vector3 boxB11 = new Vector3(0.5f, 10.3f, -2.6f);
213     private Vector3 boxC11 = new Vector3(0.5f, 10.3f, -1.6f);
214     private Vector3 boxD11 = new Vector3(0.5f, 10.3f, -0.6f);
215     private Vector3 boxE11 = new Vector3(0.5f, 10.3f, 0.3f);
216     private Vector3 boxF11 = new Vector3(0.5f, 10.3f, 1.3f);
217     private Vector3 boxG11 = new Vector3(0.5f, 10.3f, 2.3f);
218     private Vector3 boxH11 = new Vector3(0.5f, 10.3f, 3.3f);
219     private Vector3 boxI11 = new Vector3(0.5f, 10.3f, 4.3f);
220     private Vector3 boxJ11 = new Vector3(0.5f, 10.3f, 5.3f);
221
222     private Vector3 boxA12 = new Vector3(0.5f, 11.3f, -3.6f);
223     private Vector3 boxB12 = new Vector3(0.5f, 11.3f, -2.6f);
224     private Vector3 boxC12 = new Vector3(0.5f, 11.3f, -1.6f);
225     private Vector3 boxD12 = new Vector3(0.5f, 11.3f, -0.6f);
226     private Vector3 boxE12 = new Vector3(0.5f, 11.3f, 0.3f);
227     private Vector3 boxF12 = new Vector3(0.5f, 11.3f, 1.3f);
228     private Vector3 boxG12 = new Vector3(0.5f, 11.3f, 2.3f);
229     private Vector3 boxH12 = new Vector3(0.5f, 11.3f, 3.3f);
230     private Vector3 boxI12 = new Vector3(0.5f, 11.3f, 4.3f);
231     private Vector3 boxJ12 = new Vector3(0.5f, 11.3f, 5.3f);
232
233     private Vector3 boxA13 = new Vector3(0.5f, 12.3f, -3.6f);
234     private Vector3 boxB13 = new Vector3(0.5f, 12.3f, -2.6f);
235     private Vector3 boxC13 = new Vector3(0.5f, 12.3f, -1.6f);
236     private Vector3 boxD13 = new Vector3(0.5f, 12.3f, -0.6f);
237     private Vector3 boxE13 = new Vector3(0.5f, 12.3f, 0.3f);
238     private Vector3 boxF13 = new Vector3(0.5f, 12.3f, 1.3f);
239     private Vector3 boxG13 = new Vector3(0.5f, 12.3f, 2.3f);
240     private Vector3 boxH13 = new Vector3(0.5f, 12.3f, 3.3f);
241     private Vector3 boxI13 = new Vector3(0.5f, 12.3f, 4.3f);
242     private Vector3 boxJ13 = new Vector3(0.5f, 12.3f, 5.3f);
243
244     private Vector3 boxA14 = new Vector3(0.5f, 13.3f, -3.6f);
245     private Vector3 boxB14 = new Vector3(0.5f, 13.3f, -2.6f);
```

```
246     private Vector3 boxC14 = new Vector3(0.5f, 13.3f, -1.6f);
247     private Vector3 boxD14 = new Vector3(0.5f, 13.3f, -0.6f);
248     private Vector3 boxE14 = new Vector3(0.5f, 13.3f, 0.3f);
249     private Vector3 boxF14 = new Vector3(0.5f, 13.3f, 1.3f);
250     private Vector3 boxG14 = new Vector3(0.5f, 13.3f, 2.3f);
251     private Vector3 boxH14 = new Vector3(0.5f, 13.3f, 3.3f);
252     private Vector3 boxI14 = new Vector3(0.5f, 13.3f, 4.3f);
253     private Vector3 boxJ14 = new Vector3(0.5f, 13.3f, 5.3f);
254
255     private Vector3 boxA15 = new Vector3(0.5f, 14.3f, -3.6f);
256     private Vector3 boxB15 = new Vector3(0.5f, 14.3f, -2.6f);
257     private Vector3 boxC15 = new Vector3(0.5f, 14.3f, -1.6f);
258     private Vector3 boxD15 = new Vector3(0.5f, 14.3f, -0.6f);
259     private Vector3 boxE15 = new Vector3(0.5f, 14.3f, 0.3f);
260     private Vector3 boxF15 = new Vector3(0.5f, 14.3f, 1.3f);
261     private Vector3 boxG15 = new Vector3(0.5f, 14.3f, 2.3f);
262     private Vector3 boxH15 = new Vector3(0.5f, 14.3f, 3.3f);
263     private Vector3 boxI15 = new Vector3(0.5f, 14.3f, 4.3f);
264     private Vector3 boxJ15 = new Vector3(0.5f, 14.3f, 5.3f);
265
266     private Vector3 boxA16 = new Vector3(0.5f, 15.3f, -3.6f);
267     private Vector3 boxB16 = new Vector3(0.5f, 15.3f, -2.6f);
268     private Vector3 boxC16 = new Vector3(0.5f, 15.3f, -1.6f);
269     private Vector3 boxD16 = new Vector3(0.5f, 15.3f, -0.6f);
270     private Vector3 boxE16 = new Vector3(0.5f, 15.3f, 0.3f);
271     private Vector3 boxF16 = new Vector3(0.5f, 15.3f, 1.3f);
272     private Vector3 boxG16 = new Vector3(0.5f, 15.3f, 2.3f);
273     private Vector3 boxH16 = new Vector3(0.5f, 15.3f, 3.3f);
274     private Vector3 boxI16 = new Vector3(0.5f, 15.3f, 4.3f);
275     private Vector3 boxJ16 = new Vector3(0.5f, 15.3f, 5.3f);
276
277     private Vector3 boxA17 = new Vector3(0.5f, 16.3f, -3.6f);
278     private Vector3 boxB17 = new Vector3(0.5f, 16.3f, -2.6f);
279     private Vector3 boxC17 = new Vector3(0.5f, 16.3f, -1.6f);
280     private Vector3 boxD17 = new Vector3(0.5f, 16.3f, -0.6f);
281     private Vector3 boxE17 = new Vector3(0.5f, 16.3f, 0.3f);
282     private Vector3 boxF17 = new Vector3(0.5f, 16.3f, 1.3f);
283     private Vector3 boxG17 = new Vector3(0.5f, 16.3f, 2.3f);
284     private Vector3 boxH17 = new Vector3(0.5f, 16.3f, 3.3f);
285     private Vector3 boxI17 = new Vector3(0.5f, 16.3f, 4.3f);
286     private Vector3 boxJ17 = new Vector3(0.5f, 16.3f, 5.3f);
287
288     private Vector3 boxA18 = new Vector3(0.5f, 17.3f, -3.6f);
289     private Vector3 boxB18 = new Vector3(0.5f, 17.3f, -2.6f);
290     private Vector3 boxC18 = new Vector3(0.5f, 17.3f, -1.6f);
291     private Vector3 boxD18 = new Vector3(0.5f, 17.3f, -0.6f);
292     private Vector3 boxE18 = new Vector3(0.5f, 17.3f, 0.3f);
293     private Vector3 boxF18 = new Vector3(0.5f, 17.3f, 1.3f);
294     private Vector3 boxG18 = new Vector3(0.5f, 17.3f, 2.3f);
```

```
295     private Vector3 boxH18 = new Vector3(0.5f, 17.3f, 3.3f);
296     private Vector3 boxI18 = new Vector3(0.5f, 17.3f, 4.3f);
297     private Vector3 boxJ18 = new Vector3(0.5f, 17.3f, 5.3f);
298
299     private Vector3 boxA19 = new Vector3(0.5f, 18.3f, -3.6f);
300     private Vector3 boxB19 = new Vector3(0.5f, 18.3f, -2.6f);
301     private Vector3 boxC19 = new Vector3(0.5f, 18.3f, -1.6f);
302     private Vector3 boxD19 = new Vector3(0.5f, 18.3f, -0.6f);
303     private Vector3 boxE19 = new Vector3(0.5f, 18.3f, 0.3f);
304     private Vector3 boxF19 = new Vector3(0.5f, 18.3f, 1.3f);
305     private Vector3 boxG19 = new Vector3(0.5f, 18.3f, 2.3f);
306     private Vector3 boxH19 = new Vector3(0.5f, 18.3f, 3.3f);
307     private Vector3 boxI19 = new Vector3(0.5f, 18.3f, 4.3f);
308     private Vector3 boxJ19 = new Vector3(0.5f, 18.3f, 5.3f);
309
310     private Vector3 boxA20 = new Vector3(0.5f, 19.3f, -3.6f);
311     private Vector3 boxB20 = new Vector3(0.5f, 19.3f, -2.6f);
312     private Vector3 boxC20 = new Vector3(0.5f, 19.3f, -1.6f);
313     private Vector3 boxD20 = new Vector3(0.5f, 19.3f, -0.6f);
314     private Vector3 boxE20 = new Vector3(0.5f, 19.3f, 0.3f);
315     private Vector3 boxF20 = new Vector3(0.5f, 19.3f, 1.3f);
316     private Vector3 boxG20 = new Vector3(0.5f, 19.3f, 2.3f);
317     private Vector3 boxH20 = new Vector3(0.5f, 19.3f, 3.3f);
318     private Vector3 boxI20 = new Vector3(0.5f, 19.3f, 4.3f);
319     private Vector3 boxJ20 = new Vector3(0.5f, 19.3f, 5.3f);
320
321     //VacancyInts
322     //-4, -3, -2, -1, 0, 1, 2, 3, 4, 5
323
324     void Start()
325     {
326         timerTime = Random.Range(0.2f, 0.6f);
327         myRot = GameController.blockRotation;
328         AddPositions();
329         CheckRows();
330         PrefSpots();
331         ChooseSpot();
332     }
333     void Update()
334     {
335         timerTime -= Time.deltaTime;
336         myRot = GameController.blockRotation;
337         if (timerTime <= 0.0f)
338         {
339             GoToSpot();
340         }
341         if (chosenRot >= 4)
342         {
343             chosenRot = 0;
```

```
344         }
345     }
346     //Add the positions to the Lists
347     void AddPositions()
348     {
349         row1.Add(boxA1);
350         row1.Add(boxB1);
351         row1.Add(boxC1);
352         row1.Add(boxD1);
353         row1.Add(boxE1);
354         row1.Add(boxF1);
355         row1.Add(boxG1);
356         row1.Add(boxH1);
357         row1.Add(boxI1);
358         row1.Add(boxJ1);
359 
360         row2.Add(boxA2);
361         row2.Add(boxB2);
362         row2.Add(boxC2);
363         row2.Add(boxD2);
364         row2.Add(boxE2);
365         row2.Add(boxF2);
366         row2.Add(boxG2);
367         row2.Add(boxH2);
368         row2.Add(boxI2);
369         row2.Add(boxJ2);
370 
371         row3.Add(boxA3);
372         row3.Add(boxB3);
373         row3.Add(boxC3);
374         row3.Add(boxD3);
375         row3.Add(boxE3);
376         row3.Add(boxF3);
377         row3.Add(boxG3);
378         row3.Add(boxH3);
379         row3.Add(boxI3);
380         row3.Add(boxJ3);
381 
382         row4.Add(boxA4);
383         row4.Add(boxB4);
384         row4.Add(boxC4);
385         row4.Add(boxD4);
386         row4.Add(boxE4);
387         row4.Add(boxF4);
388         row4.Add(boxG4);
389         row4.Add(boxH4);
390         row4.Add(boxI4);
391         row4.Add(boxJ4);
392 
```

```
393             row5.Add(boxA5);
394             row5.Add(boxB5);
395             row5.Add(boxC5);
396             row5.Add(boxD5);
397             row5.Add(boxE5);
398             row5.Add(boxF5);
399             row5.Add(boxG5);
400             row5.Add(boxH5);
401             row5.Add(boxI5);
402             row5.Add(boxJ5);
403
404             row6.Add(boxA6);
405             row6.Add(boxB6);
406             row6.Add(boxC6);
407             row6.Add(boxD6);
408             row6.Add(boxE6);
409             row6.Add(boxF6);
410             row6.Add(boxG6);
411             row6.Add(boxH6);
412             row6.Add(boxI6);
413             row6.Add(boxJ6);
414
415             row7.Add(boxA7);
416             row7.Add(boxB7);
417             row7.Add(boxC7);
418             row7.Add(boxD7);
419             row7.Add(boxE7);
420             row7.Add(boxF7);
421             row7.Add(boxG7);
422             row7.Add(boxH7);
423             row7.Add(boxI7);
424             row7.Add(boxJ7);
425
426             row8.Add(boxA8);
427             row8.Add(boxB8);
428             row8.Add(boxC8);
429             row8.Add(boxD8);
430             row8.Add(boxE8);
431             row8.Add(boxF8);
432             row8.Add(boxG8);
433             row8.Add(boxH8);
434             row8.Add(boxI8);
435             row8.Add(boxJ8);
436
437             row9.Add(boxA9);
438             row9.Add(boxB9);
439             row9.Add(boxC9);
440             row9.Add(boxD9);
441             row9.Add(boxE9);
```

```
442         row9.Add(boxF9);
443         row9.Add(boxG9);
444         row9.Add(boxH9);
445         row9.Add(boxI9);
446         row9.Add(boxJ9);
447
448         row10.Add(boxA10);
449         row10.Add(boxB10);
450         row10.Add(boxC10);
451         row10.Add(boxD10);
452         row10.Add(boxE10);
453         row10.Add(boxF10);
454         row10.Add(boxG10);
455         row10.Add(boxH10);
456         row10.Add(boxI10);
457         row10.Add(boxJ10);
458
459         row11.Add(boxA11);
460         row11.Add(boxB11);
461         row11.Add(boxC11);
462         row11.Add(boxD11);
463         row11.Add(boxE11);
464         row11.Add(boxF11);
465         row11.Add(boxG11);
466         row11.Add(boxH11);
467         row11.Add(boxI11);
468         row11.Add(boxJ11);
469
470         row12.Add(boxA12);
471         row12.Add(boxB12);
472         row12.Add(boxC12);
473         row12.Add(boxD12);
474         row12.Add(boxE12);
475         row12.Add(boxF12);
476         row12.Add(boxG12);
477         row12.Add(boxH12);
478         row12.Add(boxI12);
479         row12.Add(boxJ12);
480
481         row13.Add(boxA13);
482         row13.Add(boxB13);
483         row13.Add(boxC13);
484         row13.Add(boxD13);
485         row13.Add(boxE13);
486         row13.Add(boxF13);
487         row13.Add(boxG13);
488         row13.Add(boxH13);
489         row13.Add(boxI13);
490         row13.Add(boxJ13);
```

```
491
492         row14.Add(boxA14);
493         row14.Add(boxB14);
494         row14.Add(boxC14);
495         row14.Add(boxD14);
496         row14.Add(boxE14);
497         row14.Add(boxF14);
498         row14.Add(boxG14);
499         row14.Add(boxH14);
500         row14.Add(boxI14);
501         row14.Add(boxJ14);
502
503         row15.Add(boxA15);
504         row15.Add(boxB15);
505         row15.Add(boxC15);
506         row15.Add(boxD15);
507         row15.Add(boxE15);
508         row15.Add(boxF15);
509         row15.Add(boxG15);
510         row15.Add(boxH15);
511         row15.Add(boxI15);
512         row15.Add(boxJ15);
513
514         row16.Add(boxA16);
515         row16.Add(boxB16);
516         row16.Add(boxC16);
517         row16.Add(boxD16);
518         row16.Add(boxE16);
519         row16.Add(boxF16);
520         row16.Add(boxG16);
521         row16.Add(boxH16);
522         row16.Add(boxI16);
523         row16.Add(boxJ16);
524
525         row17.Add(boxA17);
526         row17.Add(boxB17);
527         row17.Add(boxC17);
528         row17.Add(boxD17);
529         row17.Add(boxE17);
530         row17.Add(boxF17);
531         row17.Add(boxG17);
532         row17.Add(boxH17);
533         row17.Add(boxI17);
534         row17.Add(boxJ17);
535
536         row18.Add(boxA18);
537         row18.Add(boxB18);
538         row18.Add(boxC18);
539         row18.Add(boxD18);
```

```
540             row18.Add(boxE18);
541             row18.Add(boxF18);
542             row18.Add(boxG18);
543             row18.Add(boxH18);
544             row18.Add(boxI18);
545             row18.Add(boxJ18);
546 
547             row19.Add(boxA19);
548             row19.Add(boxB19);
549             row19.Add(boxC19);
550             row19.Add(boxD19);
551             row19.Add(boxE19);
552             row19.Add(boxF19);
553             row19.Add(boxG19);
554             row19.Add(boxH19);
555             row19.Add(boxI19);
556             row19.Add(boxJ19);
557 
558             row20.Add(boxA20);
559             row20.Add(boxB20);
560             row20.Add(boxC20);
561             row20.Add(boxD20);
562             row20.Add(boxE20);
563             row20.Add(boxF20);
564             row20.Add(boxG20);
565             row20.Add(boxH20);
566             row20.Add(boxI20);
567             row20.Add(boxJ20);
568         }
569         //Row by row, checks which rows have been checked.
570         void CheckRows()
571         {
572             if (openSpots.Count == 0)
573             {
574                 if (!checkedRow1)
575                 {
576                     checkedRow1 = true;
577                     CheckRow1();
578                 }
579                 else if (!checkedRow2)
580                 {
581                     checkedRow2 = true;
582                     CheckRow2();
583                 }
584                 else if (!checkedRow3)
585                 {
586                     checkedRow3 = true;
587                     CheckRow3();
588                 }
```

```
589             else if (!checkedRow4)
590             {
591                 checkedRow4 = true;
592                 CheckRow4();
593             }
594             else if (!checkedRow5)
595             {
596                 checkedRow5 = true;
597                 CheckRow5();
598             }
599             else if (!checkedRow6)
600             {
601                 checkedRow6 = true;
602                 CheckRow6();
603             }
604             else if (!checkedRow7)
605             {
606                 checkedRow7 = true;
607                 CheckRow7();
608             }
609             else if (!checkedRow8)
610             {
611                 checkedRow8 = true;
612                 CheckRow8();
613             }
614             else if (!checkedRow9)
615             {
616                 checkedRow9 = true;
617                 CheckRow9();
618             }
619             else if (!checkedRow10)
620             {
621                 checkedRow10 = true;
622                 CheckRow10();
623             }
624             else if (!checkedRow11)
625             {
626                 checkedRow11 = true;
627                 CheckRow11();
628             }
629             else if (!checkedRow12)
630             {
631                 checkedRow12 = true;
632                 CheckRow12();
633             }
634             else if (!checkedRow13)
635             {
636                 checkedRow13 = true;
637                 CheckRow13();
```

```
638             }
639             else if (!checkedRow14)
640             {
641                 checkedRow14 = true;
642                 CheckRow14();
643             }
644             else if (!checkedRow15)
645             {
646                 checkedRow15 = true;
647                 CheckRow15();
648             }
649             else if (!checkedRow16)
650             {
651                 checkedRow16 = true;
652                 CheckRow16();
653             }
654             else if (!checkedRow17)
655             {
656                 checkedRow17 = true;
657                 CheckRow17();
658             }
659             else if (!checkedRow18)
660             {
661                 checkedRow18 = true;
662                 CheckRow18();
663             }
664             else if (!checkedRow19)
665             {
666                 checkedRow19 = true;
667                 CheckRow19();
668             }
669             else if (!checkedRow20)
670             {
671                 checkedRow20 = true;
672                 CheckRow20();
673             }
674         }
675     }
676     void CheckRow1()
677     {
678         //Check all spaces to see if there's anything there
679         foreach (Vector3 vec in row1)
680         {
681             //If something isn't there, check the 6 spaces above that spot. If nothing's there,
682             //add the position to the row's Vacancy list and that index to the row's list of indeces.
683             if (!Physics.CheckBox(vec, cubeQuart))
684             {
```

```
685                 Vector3 vecPlus1 = new Vector3(vec.x, vec.y + 1, vec.z);
686                 Vector3 vecPlus2 = new Vector3(vec.x, vec.y + 2, vec.z);
687                 Vector3 vecPlus3 = new Vector3(vec.x, vec.y + 3, vec.z);
688                 Vector3 vecPlus4 = new Vector3(vec.x, vec.y + 4, vec.z);
689                 Vector3 vecPlus5 = new Vector3(vec.x, vec.y + 5, vec.z);
690                 Vector3 vecPlus6 = new Vector3(vec.x, vec.y + 6, vec.z);
691                 Vector3 vecPlus7 = new Vector3(vec.x, vec.y + 7, vec.z);
692                 Vector3 vecPlus8 = new Vector3(vec.x, vec.y + 8, vec.z);
693                 Vector3 vecPlus9 = new Vector3(vec.x, vec.y + 9, vec.z);
694                 Vector3 vecPlus10 = new Vector3(vec.x, vec.y + 10,
                      vec.z);
695
696                 if (!Physics.CheckBox(vecPlus1, cubeQuart) && !
                      Physics.CheckBox(vecPlus2, cubeQuart) && !
                      Physics.CheckBox(vecPlus3, cubeQuart)
697                       && !Physics.CheckBox(vecPlus4, cubeQuart) && !
                       Physics.CheckBox(vecPlus5, cubeQuart) && !
                       Physics.CheckBox(vecPlus6, cubeQuart)
698                       && !Physics.CheckBox(vecPlus7, cubeQuart) && !
                       Physics.CheckBox(vecPlus8, cubeQuart) && !
                       Physics.CheckBox(vecPlus9, cubeQuart)
699                       && !Physics.CheckBox(vecPlus10, cubeQuart))
700                 {
701                     row1Vacancies.Add(vec);
702                 }
703             }
704         }
705         foreach (Vector3 vec in row1Vacancies)
706         {
707             row1VacancyInts.Add(Mathf.RoundToInt(vec.z));
708         }
709         row1VacancyInts.Sort();
710         //Checks each of the verified empty spaces that work for the
              block's rotation.
711         //Checks if all spaces that work for that rotation are empty
              spaces.
712         //Adds possible places to list openspots.
713         //Checks each rotation based on prescribed preference.
714         foreach (int i in row1VacancyInts)
715         {
716             if (i <= 3)
717             {
718                 if (row1VacancyInts.Contains(i + 1) &&
                      row1VacancyInts.Contains(i + 2))
719                 {
720                     openSpots.Add(i);
721                 }
722             }
723         }
```

```
724         if (openSpots.Count > 0)
725         {
726             chosenRot = 1;
727         }
728         else
729         {
730             foreach (int i in row1VacancyInts)
731             {
732                 if (i <= 4)
733                 {
734                     if (row1VacancyInts.Contains(i + 1))
735                     {
736                         openSpots.Add(i);
737                     }
738                 }
739             }
740             if (openSpots.Count > 0)
741             {
742                 chosenRot = 0;
743             }
744             else
745             {
746                 foreach (int i in row1VacancyInts)
747                 {
748                     if (i >= -2)
749                     {
750                         Vector3 rot3ZMinus2YPlus1 = new Vector3(0.5f,
                      1.3f, ((float)i - 1.35f));
751                         Vector3 rot3ZMinus1YPlus1 = new Vector3(0.5f,
                      1.3f, ((float)i - 2.35f));
752                         if (!Physics.CheckBox(rot3ZMinus2YPlus1,
                      cubeQuart) && !(Physics.CheckBox(rot3ZMinus1YPlus1,
                      cubeQuart)))
753                         {
754                             openSpots.Add(i);
755                         }
756                     }
757                 }
758                 if (openSpots.Count > 0)
759                 {
760                     chosenRot = 3;
761                 }
762                 else
763                 {
764                     foreach (int i in row1VacancyInts)
765                     {
766                         if (i <= 4)
767                         {
768                             openSpots.Add(i);
```

```
769                         }
770                     }
771                     if (openSpots.Count > 0)
772                     {
773                         chosenRot = 2;
774                     }
775                 }
776             }
777         }
778         if (openSpots.Count == 0)
779         {
780             CheckRows();
781         }
782     }
783     void CheckRow2()
784     {
785         //Check all spaces to see if there's anything there
786         foreach (Vector3 vec in row2)
787         {
788             //If something isn't there, check the 6 spaces above that spot. If nothing's there,
789             //add the position to the row's Vacancy list and that index to the row's list of indeces.
790             if (!Physics.CheckBox(vec, cubeQuart))
791             {
792                 Vector3 vecPlus1 = new Vector3(vec.x, vec.y + 1, vec.z);
793                 Vector3 vecPlus2 = new Vector3(vec.x, vec.y + 2, vec.z);
794                 Vector3 vecPlus3 = new Vector3(vec.x, vec.y + 3, vec.z);
795                 Vector3 vecPlus4 = new Vector3(vec.x, vec.y + 4, vec.z);
796                 Vector3 vecPlus5 = new Vector3(vec.x, vec.y + 5, vec.z);
797                 Vector3 vecPlus6 = new Vector3(vec.x, vec.y + 6, vec.z);
798                 Vector3 vecPlus7 = new Vector3(vec.x, vec.y + 7, vec.z);
799                 Vector3 vecPlus8 = new Vector3(vec.x, vec.y + 8, vec.z);
800                 Vector3 vecPlus9 = new Vector3(vec.x, vec.y + 9, vec.z);
801                 Vector3 vecPlus10 = new Vector3(vec.x, vec.y + 10, vec.z);
802 
803                 if (!Physics.CheckBox(vecPlus1, cubeQuart) && !Physics.CheckBox(vecPlus2, cubeQuart) && !Physics.CheckBox(vecPlus3, cubeQuart)
804                    && !Physics.CheckBox(vecPlus4, cubeQuart) && !Physics.CheckBox(vecPlus5, cubeQuart) && !Physics.CheckBox(vecPlus6, cubeQuart)
805                     && !Physics.CheckBox(vecPlus7, cubeQuart) && !Physics.CheckBox(vecPlus8, cubeQuart) && !Physics.CheckBox(vecPlus9, cubeQuart)
806                         && !Physics.CheckBox(vecPlus10, cubeQuart))
807                 {
808                     row2Vacancies.Add(vec);
```

```
809                 }
810             }
811         }
812         foreach (Vector3 vec in row2Vacancies)
813         {
814             row2VacancyInts.Add(Mathf.RoundToInt(vec.z));
815         }
816         row2VacancyInts.Sort();
817         //Checks each of the verified empty spaces that work for the
              block's rotation.
818         //Checks if all spaces that work for that rotation are empty
              spaces.
819         //Adds possible places to list openspots.
820         //Checks each rotation based on prescribed preference.
821         foreach (int i in row2VacancyInts)
822         {
823             if (i <= 3)
824             {
825                 if (row2VacancyInts.Contains(i + 1) &&
                      row2VacancyInts.Contains(i + 2))
826                 {
827                     openSpots.Add(i);
828                 }
829             }
830         }
831         if (openSpots.Count > 0)
832         {
833             chosenRot = 1;
834         }
835         else
836         {
837             foreach (int i in row2VacancyInts)
838             {
839                 if (i <= 4)
840                 {
841                     if (row2VacancyInts.Contains(i + 1))
842                     {
843                         openSpots.Add(i);
844                     }
845                 }
846             }
847             if (openSpots.Count > 0)
848             {
849                 chosenRot = 0;
850             }
851             else
852             {
853                 foreach (int i in row2VacancyInts)
854                 {
```

```
855                     if (i >= -2)
856                     {
857                         Vector3 rot3ZMinus2YPlus1 = new Vector3(0.5f,
                    2.3f, ((float)i - 1.35f));
858                         Vector3 rot3ZMinus1YPlus1 = new Vector3(0.5f,
                    2.3f, ((float)i - 2.35f));
859                         if (!Physics.CheckBox(rot3ZMinus2YPlus1,
                    cubeQuart) && !(Physics.CheckBox(rot3ZMinus1YPlus1,
                    cubeQuart)))
860                         {
861                             openSpots.Add(i);
862                         }
863                     }
864                 }
865                 if (openSpots.Count > 0)
866                 {
867                     chosenRot = 3;
868                 }
869                 else
870                 {
871                     foreach (int i in row2VacancyInts)
872                     {
873                         if (i <= 4)
874                         {
875                             openSpots.Add(i);
876                         }
877                     }
878                     if (openSpots.Count > 0)
879                     {
880                         chosenRot = 2;
881                     }
882                 }
883             }
884         }
885         if (openSpots.Count == 0)
886         {
887             CheckRows();
888         }
889     }
890     void CheckRow3()
891     {
892         //Check all spaces to see if there's anything there
893         foreach (Vector3 vec in row3)
894         {
895             //If something isn't there, check the 6 spaces above that
                spot. If nothing's there,
896             //add the position to the row's Vacancy list and that index
                to the row's list of indeces.
897             if (!Physics.CheckBox(vec, cubeQuart))
```

```
898             {
899                 Vector3 vecPlus1 = new Vector3(vec.x, vec.y + 1, vec.z);
900                 Vector3 vecPlus2 = new Vector3(vec.x, vec.y + 2, vec.z);
901                 Vector3 vecPlus3 = new Vector3(vec.x, vec.y + 3, vec.z);
902                 Vector3 vecPlus4 = new Vector3(vec.x, vec.y + 4, vec.z);
903                 Vector3 vecPlus5 = new Vector3(vec.x, vec.y + 5, vec.z);
904                 Vector3 vecPlus6 = new Vector3(vec.x, vec.y + 6, vec.z);
905                 Vector3 vecPlus7 = new Vector3(vec.x, vec.y + 7, vec.z);
906                 Vector3 vecPlus8 = new Vector3(vec.x, vec.y + 8, vec.z);
907                 Vector3 vecPlus9 = new Vector3(vec.x, vec.y + 9, vec.z);
908                 Vector3 vecPlus10 = new Vector3(vec.x, vec.y + 10,
                      vec.z);
909 
910                 if (!Physics.CheckBox(vecPlus1, cubeQuart) && !
                      Physics.CheckBox(vecPlus2, cubeQuart) && !
                      Physics.CheckBox(vecPlus3, cubeQuart)
911                     && !Physics.CheckBox(vecPlus4, cubeQuart) && !
                     Physics.CheckBox(vecPlus5, cubeQuart) && !
                     Physics.CheckBox(vecPlus6, cubeQuart)
912                         && !Physics.CheckBox(vecPlus7, cubeQuart) && !
                     Physics.CheckBox(vecPlus8, cubeQuart) && !
                     Physics.CheckBox(vecPlus9, cubeQuart)
913                             && !Physics.CheckBox(vecPlus10, cubeQuart))
914                 {
915                     row3Vacancies.Add(vec);
916                 }
917             }
918         }
919         foreach (Vector3 vec in row3Vacancies)
920         {
921             row3VacancyInts.Add(Mathf.RoundToInt(vec.z));
922         }
923         row3VacancyInts.Sort();
924         //Checks each of the verified empty spaces that work for the
              block's rotation.
925         //Checks if all spaces that work for that rotation are empty
              spaces.
926         //Adds possible places to list openspots.
927         //Checks each rotation based on prescribed preference.
928         foreach (int i in row3VacancyInts)
929         {
930             if (i <= 3)
931             {
932                 if (row3VacancyInts.Contains(i + 1) &&
                      row3VacancyInts.Contains(i + 2))
933                 {
934                     openSpots.Add(i);
935                 }
936             }
```

```
937         }
938         if (openSpots.Count > 0)
939         {
940             chosenRot = 1;
941         }
942         else
943         {
944             foreach (int i in row3VacancyInts)
945             {
946                 if (i <= 4)
947                 {
948                     if (row3VacancyInts.Contains(i + 1))
949                     {
950                         openSpots.Add(i);
951                     }
952                 }
953             }
954             if (openSpots.Count > 0)
955             {
956                 chosenRot = 0;
957             }
958             else
959             {
960                 foreach (int i in row3VacancyInts)
961                 {
962                     if (i >= -2)
963                     {
964                         Vector3 rot3ZMinus2YPlus1 = new Vector3(0.5f,
                        3.3f, ((float)i - 1.35f));
965                         Vector3 rot3ZMinus1YPlus1 = new Vector3(0.5f,
                        3.3f, ((float)i - 2.35f));
966                         if (!Physics.CheckBox(rot3ZMinus2YPlus1,
                        cubeQuart) && !(Physics.CheckBox(rot3ZMinus1YPlus1,
                        cubeQuart)))
967                         {
968                             openSpots.Add(i);
969                         }
970                     }
971                 }
972                 if (openSpots.Count > 0)
973                 {
974                     chosenRot = 3;
975                 }
976                 else
977                 {
978                     foreach (int i in row3VacancyInts)
979                     {
980                         if (i <= 4)
981                         {
```

```
982                             openSpots.Add(i);
983                         }
984                     }
985                     if (openSpots.Count > 0)
986                     {
987                         chosenRot = 2;
988                     }
989                 }
990             }
991         }
992         if (openSpots.Count == 0)
993         {
994             CheckRows();
995         }
996     }
997     void CheckRow4()
998     {
999         //Check all spaces to see if there's anything there
1000        foreach (Vector3 vec in row4)
1001        {
1002            //If something isn't there, check the 6 spaces above that spot. If nothing's there,
1003            //add the position to the row's Vacancy list and that index to the row's list of indeces.
1004            if (!Physics.CheckBox(vec, cubeQuart))
1005            {
1006                Vector3 vecPlus1 = new Vector3(vec.x, vec.y + 1, vec.z);
1007                Vector3 vecPlus2 = new Vector3(vec.x, vec.y + 2, vec.z);
1008                Vector3 vecPlus3 = new Vector3(vec.x, vec.y + 3, vec.z);
1009                Vector3 vecPlus4 = new Vector3(vec.x, vec.y + 4, vec.z);
1010                Vector3 vecPlus5 = new Vector3(vec.x, vec.y + 5, vec.z);
1011                Vector3 vecPlus6 = new Vector3(vec.x, vec.y + 6, vec.z);
1012                Vector3 vecPlus7 = new Vector3(vec.x, vec.y + 7, vec.z);
1013                Vector3 vecPlus8 = new Vector3(vec.x, vec.y + 8, vec.z);
1014                Vector3 vecPlus9 = new Vector3(vec.x, vec.y + 9, vec.z);
1015                Vector3 vecPlus10 = new Vector3(vec.x, vec.y + 10, vec.z);
1016
1017                if (!Physics.CheckBox(vecPlus1, cubeQuart) && !Physics.CheckBox(vecPlus2, cubeQuart) && !Physics.CheckBox(vecPlus3, cubeQuart)
1018                    && !Physics.CheckBox(vecPlus4, cubeQuart) && !Physics.CheckBox(vecPlus5, cubeQuart) && !Physics.CheckBox(vecPlus6, cubeQuart)
1019                    && !Physics.CheckBox(vecPlus7, cubeQuart) && !Physics.CheckBox(vecPlus8, cubeQuart) && !Physics.CheckBox(vecPlus9, cubeQuart)
1020                        && !Physics.CheckBox(vecPlus10, cubeQuart))
1021                {
```

```
1022                        row4Vacancies.Add(vec);
1023                    }
1024                }
1025            }
1026            foreach (Vector3 vec in row4Vacancies)
1027            {
1028                row4VacancyInts.Add(Mathf.RoundToInt(vec.z));
1029            }
1030            row4VacancyInts.Sort();
1031            //Checks each of the verified empty spaces that work for the
                  block's rotation.
1032            //Checks if all spaces that work for that rotation are empty
                  spaces.
1033            //Adds possible places to list openspots.
1034            //Checks each rotation based on prescribed preference.
1035            foreach (int i in row4VacancyInts)
1036            {
1037                if (i <= 3)
1038                {
1039                    if (row4VacancyInts.Contains(i + 1) &&
                          row4VacancyInts.Contains(i + 2))
1040                    {
1041                        openSpots.Add(i);
1042                    }
1043                }
1044            }
1045            if (openSpots.Count > 0)
1046            {
1047                chosenRot = 1;
1048            }
1049            else
1050            {
1051                foreach (int i in row4VacancyInts)
1052                {
1053                    if (i <= 4)
1054                    {
1055                        if (row4VacancyInts.Contains(i + 1))
1056                        {
1057                            openSpots.Add(i);
1058                        }
1059                    }
1060                }
1061                if (openSpots.Count > 0)
1062                {
1063                    chosenRot = 0;
1064                }
1065                else
1066                {
1067                    foreach (int i in row4VacancyInts)
```

```
1068                    {
1069                        if (i >= -2)
1070                        {
1071                            Vector3 rot3ZMinus2YPlus1 = new Vector3(0.5f,
                          4.3f, ((float)i - 1.35f));
1072                            Vector3 rot3ZMinus1YPlus1 = new Vector3(0.5f,
                          4.3f, ((float)i - 2.35f));
1073                            if (!Physics.CheckBox(rot3ZMinus2YPlus1,
                          cubeQuart) && !(Physics.CheckBox(rot3ZMinus1YPlus1,
                          cubeQuart)))
1074                            {
1075                                openSpots.Add(i);
1076                            }
1077                        }
1078                    }
1079                    if (openSpots.Count > 0)
1080                    {
1081                        chosenRot = 3;
1082                    }
1083                    else
1084                    {
1085                        foreach (int i in row4VacancyInts)
1086                        {
1087                            if (i <= 4)
1088                            {
1089                                openSpots.Add(i);
1090                            }
1091                        }
1092                        if (openSpots.Count > 0)
1093                        {
1094                            chosenRot = 2;
1095                        }
1096                    }
1097                }
1098            }
1099            if (openSpots.Count == 0)
1100            {
1101                CheckRows();
1102            }
1103        }
1104        void CheckRow5()
1105        {
1106            //Check all spaces to see if there's anything there
1107            foreach (Vector3 vec in row5)
1108            {
1109                //If something isn't there, check the 6 spaces above that
                      spot. If nothing's there,
1110                //add the position to the row's Vacancy list and that index
                      to the row's list of indeces.
```

```
1111            if (!Physics.CheckBox(vec, cubeQuart))
1112            {
1113                Vector3 vecPlus1 = new Vector3(vec.x, vec.y + 1, vec.z);
1114                Vector3 vecPlus2 = new Vector3(vec.x, vec.y + 2, vec.z);
1115                Vector3 vecPlus3 = new Vector3(vec.x, vec.y + 3, vec.z);
1116                Vector3 vecPlus4 = new Vector3(vec.x, vec.y + 4, vec.z);
1117                Vector3 vecPlus5 = new Vector3(vec.x, vec.y + 5, vec.z);
1118                Vector3 vecPlus6 = new Vector3(vec.x, vec.y + 6, vec.z);
1119                Vector3 vecPlus7 = new Vector3(vec.x, vec.y + 7, vec.z);
1120                Vector3 vecPlus8 = new Vector3(vec.x, vec.y + 8, vec.z);
1121                Vector3 vecPlus9 = new Vector3(vec.x, vec.y + 9, vec.z);
1122                Vector3 vecPlus10 = new Vector3(vec.x, vec.y + 10,
                      vec.z);
1123
1124                if (!Physics.CheckBox(vecPlus1, cubeQuart) && !
                      Physics.CheckBox(vecPlus2, cubeQuart) && !
                      Physics.CheckBox(vecPlus3, cubeQuart)
1125                       && !Physics.CheckBox(vecPlus4, cubeQuart) && !
                       Physics.CheckBox(vecPlus5, cubeQuart) && !
                       Physics.CheckBox(vecPlus6, cubeQuart)
1126                           && !Physics.CheckBox(vecPlus7, cubeQuart) && !
                       Physics.CheckBox(vecPlus8, cubeQuart) && !
                       Physics.CheckBox(vecPlus9, cubeQuart)
1127                            && !Physics.CheckBox(vecPlus10, cubeQuart))
1128                {
1129                    row5Vacancies.Add(vec);
1130                }
1131            }
1132        }
1133        foreach (Vector3 vec in row5Vacancies)
1134        {
1135            row5VacancyInts.Add(Mathf.RoundToInt(vec.z));
1136        }
1137        row5VacancyInts.Sort();
1138        //Checks each of the verified empty spaces that work for the
              block's rotation.
1139        //Checks if all spaces that work for that rotation are empty
              spaces.
1140        //Adds possible places to list openspots.
1141        //Checks each rotation based on prescribed preference.
1142        foreach (int i in row5VacancyInts)
1143        {
1144            if (i <= 3)
1145            {
1146                if (row5VacancyInts.Contains(i + 1) &&
                      row5VacancyInts.Contains(i + 2))
1147                {
1148                    openSpots.Add(i);
1149                }
```

```
1150                }
1151            }
1152            if (openSpots.Count > 0)
1153            {
1154                chosenRot = 1;
1155            }
1156            else
1157            {
1158                foreach (int i in row5VacancyInts)
1159                {
1160                    if (i <= 4)
1161                    {
1162                        if (row5VacancyInts.Contains(i + 1))
1163                        {
1164                            openSpots.Add(i);
1165                        }
1166                    }
1167                }
1168                if (openSpots.Count > 0)
1169                {
1170                    chosenRot = 0;
1171                }
1172                else
1173                {
1174                    foreach (int i in row5VacancyInts)
1175                    {
1176                        if (i >= -2)
1177                        {
1178                            Vector3 rot3ZMinus2YPlus1 = new Vector3(0.5f,
                      5.3f, ((float)i - 1.35f));
1179                            Vector3 rot3ZMinus1YPlus1 = new Vector3(0.5f,
                      5.3f, ((float)i - 2.35f));
1180                            if (!Physics.CheckBox(rot3ZMinus2YPlus1,
                      cubeQuart) && !(Physics.CheckBox(rot3ZMinus1YPlus1,
                      cubeQuart)))
1181                            {
1182                                openSpots.Add(i);
1183                            }
1184                        }
1185                    }
1186                    if (openSpots.Count > 0)
1187                    {
1188                        chosenRot = 3;
1189                    }
1190                    else
1191                    {
1192                        foreach (int i in row5VacancyInts)
1193                        {
1194                            if (i <= 4)
```

```
1195                            {
1196                                openSpots.Add(i);
1197                            }
1198                        }
1199                        if (openSpots.Count > 0)
1200                        {
1201                            chosenRot = 2;
1202                        }
1203                    }
1204                }
1205            }
1206            if (openSpots.Count == 0)
1207            {
1208                CheckRows();
1209            }
1210        }
1211        void CheckRow6()
1212        {
1213            //Check all spaces to see if there's anything there
1214            foreach (Vector3 vec in row6)
1215            {
1216                //If something isn't there, check the 6 spaces above that spot. If nothing's there,
1217                //add the position to the row's Vacancy list and that index to the row's list of indeces.
1218                if (!Physics.CheckBox(vec, cubeQuart))
1219                {
1220                    Vector3 vecPlus1 = new Vector3(vec.x, vec.y + 1, vec.z);
1221                    Vector3 vecPlus2 = new Vector3(vec.x, vec.y + 2, vec.z);
1222                    Vector3 vecPlus3 = new Vector3(vec.x, vec.y + 3, vec.z);
1223                    Vector3 vecPlus4 = new Vector3(vec.x, vec.y + 4, vec.z);
1224                    Vector3 vecPlus5 = new Vector3(vec.x, vec.y + 5, vec.z);
1225                    Vector3 vecPlus6 = new Vector3(vec.x, vec.y + 6, vec.z);
1226                    Vector3 vecPlus7 = new Vector3(vec.x, vec.y + 7, vec.z);
1227                    Vector3 vecPlus8 = new Vector3(vec.x, vec.y + 8, vec.z);
1228                    Vector3 vecPlus9 = new Vector3(vec.x, vec.y + 9, vec.z);
1229                    Vector3 vecPlus10 = new Vector3(vec.x, vec.y + 10, vec.z);
1230
1231                    if (!Physics.CheckBox(vecPlus1, cubeQuart) && !Physics.CheckBox(vecPlus2, cubeQuart) && !Physics.CheckBox(vecPlus3, cubeQuart)
1232                       && !Physics.CheckBox(vecPlus4, cubeQuart) && !Physics.CheckBox(vecPlus5, cubeQuart) && !Physics.CheckBox(vecPlus6, cubeQuart)
1233                          && !Physics.CheckBox(vecPlus7, cubeQuart) && !Physics.CheckBox(vecPlus8, cubeQuart) && !Physics.CheckBox(vecPlus9, cubeQuart)
1234                             && !Physics.CheckBox(vecPlus10, cubeQuart))
```

```
1235                    {
1236                        row6Vacancies.Add(vec);
1237                    }
1238                }
1239            }
1240            foreach (Vector3 vec in row6Vacancies)
1241            {
1242                row6VacancyInts.Add(Mathf.RoundToInt(vec.z));
1243            }
1244            row6VacancyInts.Sort();
1245            //Checks each of the verified empty spaces that work for the block's rotation.
1246            //Checks if all spaces that work for that rotation are empty spaces.
1247            //Adds possible places to list openspots.
1248            //Checks each rotation based on prescribed preference.
1249            foreach (int i in row6VacancyInts)
1250            {
1251                if (i <= 3)
1252                {
1253                    if (row6VacancyInts.Contains(i + 1) && row6VacancyInts.Contains(i + 2))
1254                    {
1255                        openSpots.Add(i);
1256                    }
1257                }
1258            }
1259            if (openSpots.Count > 0)
1260            {
1261                chosenRot = 1;
1262            }
1263            else
1264            {
1265                foreach (int i in row6VacancyInts)
1266                {
1267                    if (i <= 4)
1268                    {
1269                        if (row6VacancyInts.Contains(i + 1))
1270                        {
1271                            openSpots.Add(i);
1272                        }
1273                    }
1274                }
1275                if (openSpots.Count > 0)
1276                {
1277                    chosenRot = 0;
1278                }
1279                else
1280                {
```

```
1281                foreach (int i in row6VacancyInts)
1282                {
1283                    if (i >= -2)
1284                    {
1285                        Vector3 rot3ZMinus2YPlus1 = new Vector3(0.5f,
                      6.3f, ((float)i - 1.35f));
1286                        Vector3 rot3ZMinus1YPlus1 = new Vector3(0.5f,
                      6.3f, ((float)i - 2.35f));
1287                        if (!Physics.CheckBox(rot3ZMinus2YPlus1,
                      cubeQuart) && !(Physics.CheckBox(rot3ZMinus1YPlus1,
                      cubeQuart)))
1288                        {
1289                            openSpots.Add(i);
1290                        }
1291                    }
1292                }
1293                if (openSpots.Count > 0)
1294                {
1295                    chosenRot = 3;
1296                }
1297                else
1298                {
1299                    foreach (int i in row6VacancyInts)
1300                    {
1301                        if (i <= 4)
1302                        {
1303                            openSpots.Add(i);
1304                        }
1305                    }
1306                    if (openSpots.Count > 0)
1307                    {
1308                        chosenRot = 2;
1309                    }
1310                }
1311            }
1312        }
1313        if (openSpots.Count == 0)
1314        {
1315            CheckRows();
1316        }
1317    }
1318    void CheckRow7()
1319    {
1320        //Check all spaces to see if there's anything there
1321        foreach (Vector3 vec in row7)
1322        {
1323            //If something isn't there, check the 6 spaces above that
                  spot. If nothing's there,
1324            //add the position to the row's Vacancy list and that index
```

```
                to the row's list of indeces.
            if (!Physics.CheckBox(vec, cubeQuart))
            {
                Vector3 vecPlus1 = new Vector3(vec.x, vec.y + 1, vec.z);
                Vector3 vecPlus2 = new Vector3(vec.x, vec.y + 2, vec.z);
                Vector3 vecPlus3 = new Vector3(vec.x, vec.y + 3, vec.z);
                Vector3 vecPlus4 = new Vector3(vec.x, vec.y + 4, vec.z);
                Vector3 vecPlus5 = new Vector3(vec.x, vec.y + 5, vec.z);
                Vector3 vecPlus6 = new Vector3(vec.x, vec.y + 6, vec.z);
                Vector3 vecPlus7 = new Vector3(vec.x, vec.y + 7, vec.z);
                Vector3 vecPlus8 = new Vector3(vec.x, vec.y + 8, vec.z);
                Vector3 vecPlus9 = new Vector3(vec.x, vec.y + 9, vec.z);
                Vector3 vecPlus10 = new Vector3(vec.x, vec.y + 10,
                  vec.z);

                if (!Physics.CheckBox(vecPlus1, cubeQuart) && !
                  Physics.CheckBox(vecPlus2, cubeQuart) && !
                  Physics.CheckBox(vecPlus3, cubeQuart)
                     && !Physics.CheckBox(vecPlus4, cubeQuart) && !
                   Physics.CheckBox(vecPlus5, cubeQuart) && !
                   Physics.CheckBox(vecPlus6, cubeQuart)
                      && !Physics.CheckBox(vecPlus7, cubeQuart) && !
                   Physics.CheckBox(vecPlus8, cubeQuart) && !
                   Physics.CheckBox(vecPlus9, cubeQuart)
                        && !Physics.CheckBox(vecPlus10, cubeQuart))
                {
                    row7Vacancies.Add(vec);
                }
            }
        }
        foreach (Vector3 vec in row7Vacancies)
        {
            row7VacancyInts.Add(Mathf.RoundToInt(vec.z));
        }
        row7VacancyInts.Sort();
        //Checks each of the verified empty spaces that work for the
          block's rotation.
        //Checks if all spaces that work for that rotation are empty
          spaces.
        //Adds possible places to list openspots.
        //Checks each rotation based on prescribed preference.
        foreach (int i in row7VacancyInts)
        {
            if (i <= 3)
            {
                if (row7VacancyInts.Contains(i + 1) &&
                  row7VacancyInts.Contains(i + 2))
                {
                    openSpots.Add(i);
```

```
1363                    }
1364                }
1365            }
1366            if (openSpots.Count > 0)
1367            {
1368                chosenRot = 1;
1369            }
1370            else
1371            {
1372                foreach (int i in row7VacancyInts)
1373                {
1374                    if (i <= 4)
1375                    {
1376                        if (row7VacancyInts.Contains(i + 1))
1377                        {
1378                            openSpots.Add(i);
1379                        }
1380                    }
1381                }
1382                if (openSpots.Count > 0)
1383                {
1384                    chosenRot = 0;
1385                }
1386                else
1387                {
1388                    foreach (int i in row7VacancyInts)
1389                    {
1390                        if (i >= -2)
1391                        {
1392                            Vector3 rot3ZMinus2YPlus1 = new Vector3(0.5f,
                      7.3f, ((float)i - 1.35f));
1393                            Vector3 rot3ZMinus1YPlus1 = new Vector3(0.5f,
                      7.3f, ((float)i - 2.35f));
1394                            if (!Physics.CheckBox(rot3ZMinus2YPlus1,
                      cubeQuart) && !(Physics.CheckBox(rot3ZMinus1YPlus1,
                      cubeQuart)))
1395                            {
1396                                openSpots.Add(i);
1397                            }
1398                        }
1399                    }
1400                    if (openSpots.Count > 0)
1401                    {
1402                        chosenRot = 3;
1403                    }
1404                    else
1405                    {
1406                        foreach (int i in row7VacancyInts)
1407                        {
```

```
1408                            if (i <= 4)
1409                            {
1410                                openSpots.Add(i);
1411                            }
1412                        }
1413                        if (openSpots.Count > 0)
1414                        {
1415                            chosenRot = 2;
1416                        }
1417                    }
1418                }
1419            }
1420            if (openSpots.Count == 0)
1421            {
1422                CheckRows();
1423            }
1424        }
1425        void CheckRow8()
1426        {
1427            //Check all spaces to see if there's anything there
1428            foreach (Vector3 vec in row8)
1429            {
1430                //If something isn't there, check the 6 spaces above that
                      spot. If nothing's there,
1431                //add the position to the row's Vacancy list and that index
                      to the row's list of indeces.
1432                if (!Physics.CheckBox(vec, cubeQuart))
1433                {
1434                    Vector3 vecPlus1 = new Vector3(vec.x, vec.y + 1, vec.z);
1435                    Vector3 vecPlus2 = new Vector3(vec.x, vec.y + 2, vec.z);
1436                    Vector3 vecPlus3 = new Vector3(vec.x, vec.y + 3, vec.z);
1437                    Vector3 vecPlus4 = new Vector3(vec.x, vec.y + 4, vec.z);
1438                    Vector3 vecPlus5 = new Vector3(vec.x, vec.y + 5, vec.z);
1439                    Vector3 vecPlus6 = new Vector3(vec.x, vec.y + 6, vec.z);
1440                    Vector3 vecPlus7 = new Vector3(vec.x, vec.y + 7, vec.z);
1441                    Vector3 vecPlus8 = new Vector3(vec.x, vec.y + 8, vec.z);
1442                    Vector3 vecPlus9 = new Vector3(vec.x, vec.y + 9, vec.z);
1443                    Vector3 vecPlus10 = new Vector3(vec.x, vec.y + 10,
                          vec.z);
1444
1445                    if (!Physics.CheckBox(vecPlus1, cubeQuart) && !
                          Physics.CheckBox(vecPlus2, cubeQuart) && !
                          Physics.CheckBox(vecPlus3, cubeQuart)
1446                           && !Physics.CheckBox(vecPlus4, cubeQuart) && !
                           Physics.CheckBox(vecPlus5, cubeQuart) && !
                           Physics.CheckBox(vecPlus6, cubeQuart)
1447                            && !Physics.CheckBox(vecPlus7, cubeQuart) && !
                           Physics.CheckBox(vecPlus8, cubeQuart) && !
                           Physics.CheckBox(vecPlus9, cubeQuart)
```

```
1448                            && !Physics.CheckBox(vecPlus10, cubeQuart))
1449                {
1450                    row8Vacancies.Add(vec);
1451                }
1452            }
1453        }
1454        foreach (Vector3 vec in row8Vacancies)
1455        {
1456            row8VacancyInts.Add(Mathf.RoundToInt(vec.z));
1457        }
1458        row8VacancyInts.Sort();
1459        //Checks each of the verified empty spaces that work for the block's rotation.
1460        //Checks if all spaces that work for that rotation are empty spaces.
1461        //Adds possible places to list openspots.
1462        //Checks each rotation based on prescribed preference.
1463        foreach (int i in row8VacancyInts)
1464        {
1465            if (i <= 3)
1466            {
1467                if (row8VacancyInts.Contains(i + 1) && row8VacancyInts.Contains(i + 2))
1468                {
1469                    openSpots.Add(i);
1470                }
1471            }
1472        }
1473        if (openSpots.Count > 0)
1474        {
1475            chosenRot = 1;
1476        }
1477        else
1478        {
1479            foreach (int i in row8VacancyInts)
1480            {
1481                if (i <= 4)
1482                {
1483                    if (row8VacancyInts.Contains(i + 1))
1484                    {
1485                        openSpots.Add(i);
1486                    }
1487                }
1488            }
1489            if (openSpots.Count > 0)
1490            {
1491                chosenRot = 0;
1492            }
1493            else
```

```
1494            {
1495                foreach (int i in row8VacancyInts)
1496                {
1497                    if (i >= -2)
1498                    {
1499                        Vector3 rot3ZMinus2YPlus1 = new Vector3(0.5f,
                      8.3f, ((float)i - 1.35f));
1500                        Vector3 rot3ZMinus1YPlus1 = new Vector3(0.5f,
                      8.3f, ((float)i - 2.35f));
1501                        if (!Physics.CheckBox(rot3ZMinus2YPlus1,
                      cubeQuart) && !(Physics.CheckBox(rot3ZMinus1YPlus1,
                      cubeQuart)))
1502                        {
1503                            openSpots.Add(i);
1504                        }
1505                    }
1506                }
1507                if (openSpots.Count > 0)
1508                {
1509                    chosenRot = 3;
1510                }
1511                else
1512                {
1513                    foreach (int i in row8VacancyInts)
1514                    {
1515                        if (i <= 4)
1516                        {
1517                            openSpots.Add(i);
1518                        }
1519                    }
1520                    if (openSpots.Count > 0)
1521                    {
1522                        chosenRot = 2;
1523                    }
1524                }
1525            }
1526        }
1527        if (openSpots.Count == 0)
1528        {
1529            CheckRows();
1530        }
1531    }
1532    void CheckRow9()
1533    {
1534        //Check all spaces to see if there's anything there
1535        foreach (Vector3 vec in row9)
1536        {
1537            //If something isn't there, check the 6 spaces above that
                  spot. If nothing's there,
```

```
1538            //add the position to the row's Vacancy list and that index
                  to the row's list of indeces.
1539            if (!Physics.CheckBox(vec, cubeQuart))
1540            {
1541                Vector3 vecPlus1 = new Vector3(vec.x, vec.y + 1, vec.z);
1542                Vector3 vecPlus2 = new Vector3(vec.x, vec.y + 2, vec.z);
1543                Vector3 vecPlus3 = new Vector3(vec.x, vec.y + 3, vec.z);
1544                Vector3 vecPlus4 = new Vector3(vec.x, vec.y + 4, vec.z);
1545                Vector3 vecPlus5 = new Vector3(vec.x, vec.y + 5, vec.z);
1546                Vector3 vecPlus6 = new Vector3(vec.x, vec.y + 6, vec.z);
1547                Vector3 vecPlus7 = new Vector3(vec.x, vec.y + 7, vec.z);
1548                Vector3 vecPlus8 = new Vector3(vec.x, vec.y + 8, vec.z);
1549                Vector3 vecPlus9 = new Vector3(vec.x, vec.y + 9, vec.z);
1550                Vector3 vecPlus10 = new Vector3(vec.x, vec.y + 10,
                      vec.z);
1551
1552                if (!Physics.CheckBox(vecPlus1, cubeQuart) && !
                      Physics.CheckBox(vecPlus2, cubeQuart) && !
                      Physics.CheckBox(vecPlus3, cubeQuart)
1553                    && !Physics.CheckBox(vecPlus4, cubeQuart) && !
                     Physics.CheckBox(vecPlus5, cubeQuart) && !
                     Physics.CheckBox(vecPlus6, cubeQuart)
1554                     && !Physics.CheckBox(vecPlus7, cubeQuart) && !
                     Physics.CheckBox(vecPlus8, cubeQuart) && !
                     Physics.CheckBox(vecPlus9, cubeQuart)
1555                          && !Physics.CheckBox(vecPlus10, cubeQuart))
1556                {
1557                    row9Vacancies.Add(vec);
1558                }
1559            }
1560        }
1561        foreach (Vector3 vec in row9Vacancies)
1562        {
1563            row9VacancyInts.Add(Mathf.RoundToInt(vec.z));
1564        }
1565        row9VacancyInts.Sort();
1566        //Checks each of the verified empty spaces that work for the
              block's rotation.
1567        //Checks if all spaces that work for that rotation are empty
              spaces.
1568        //Adds possible places to list openspots.
1569        //Checks each rotation based on prescribed preference.
1570        foreach (int i in row9VacancyInts)
1571        {
1572            if (i <= 3)
1573            {
1574                if (row9VacancyInts.Contains(i + 1) &&
                      row9VacancyInts.Contains(i + 2))
1575                {
```

```
1576                        openSpots.Add(i);
1577                    }
1578                }
1579            }
1580            if (openSpots.Count > 0)
1581            {
1582                chosenRot = 1;
1583            }
1584            else
1585            {
1586                foreach (int i in row9VacancyInts)
1587                {
1588                    if (i <= 4)
1589                    {
1590                        if (row9VacancyInts.Contains(i + 1))
1591                        {
1592                            openSpots.Add(i);
1593                        }
1594                    }
1595                }
1596                if (openSpots.Count > 0)
1597                {
1598                    chosenRot = 0;
1599                }
1600                else
1601                {
1602                    foreach (int i in row9VacancyInts)
1603                    {
1604                        if (i >= -2)
1605                        {
1606                            Vector3 rot3ZMinus2YPlus1 = new Vector3(0.5f,
                      9.3f, ((float)i - 1.35f));
1607                            Vector3 rot3ZMinus1YPlus1 = new Vector3(0.5f,
                      9.3f, ((float)i - 2.35f));
1608                            if (!Physics.CheckBox(rot3ZMinus2YPlus1,
                      cubeQuart) && !(Physics.CheckBox(rot3ZMinus1YPlus1,
                      cubeQuart)))
1609                            {
1610                                openSpots.Add(i);
1611                            }
1612                        }
1613                    }
1614                    if (openSpots.Count > 0)
1615                    {
1616                        chosenRot = 3;
1617                    }
1618                    else
1619                    {
1620                        foreach (int i in row9VacancyInts)
```

```
1621                    {
1622                        if (i <= 4)
1623                        {
1624                            openSpots.Add(i);
1625                        }
1626                    }
1627                    if (openSpots.Count > 0)
1628                    {
1629                        chosenRot = 2;
1630                    }
1631                }
1632            }
1633        }
1634        if (openSpots.Count == 0)
1635        {
1636            CheckRows();
1637        }
1638    }
1639    void CheckRow10()
1640    {
1641        //Check all spaces to see if there's anything there
1642        foreach (Vector3 vec in row10)
1643        {
1644            //If something isn't there, check the 6 spaces above that spot. If nothing's there,
1645            //add the position to the row's Vacancy list and that index to the row's list of indeces.
1646            if (!Physics.CheckBox(vec, cubeQuart))
1647            {
1648                Vector3 vecPlus1 = new Vector3(vec.x, vec.y + 1, vec.z);
1649                Vector3 vecPlus2 = new Vector3(vec.x, vec.y + 2, vec.z);
1650                Vector3 vecPlus3 = new Vector3(vec.x, vec.y + 3, vec.z);
1651                Vector3 vecPlus4 = new Vector3(vec.x, vec.y + 4, vec.z);
1652                Vector3 vecPlus5 = new Vector3(vec.x, vec.y + 5, vec.z);
1653                Vector3 vecPlus6 = new Vector3(vec.x, vec.y + 6, vec.z);
1654                Vector3 vecPlus7 = new Vector3(vec.x, vec.y + 7, vec.z);
1655                Vector3 vecPlus8 = new Vector3(vec.x, vec.y + 8, vec.z);
1656                Vector3 vecPlus9 = new Vector3(vec.x, vec.y + 9, vec.z);
1657                Vector3 vecPlus10 = new Vector3(vec.x, vec.y + 10, vec.z);
1658
1659                if (!Physics.CheckBox(vecPlus1, cubeQuart) && !Physics.CheckBox(vecPlus2, cubeQuart) && !Physics.CheckBox(vecPlus3, cubeQuart)
1660                    && !Physics.CheckBox(vecPlus4, cubeQuart) && !Physics.CheckBox(vecPlus5, cubeQuart) && !Physics.CheckBox(vecPlus6, cubeQuart)
1661                    && !Physics.CheckBox(vecPlus7, cubeQuart) && !Physics.CheckBox(vecPlus8, cubeQuart) && !
```

```
                    Physics.CheckBox(vecPlus9, cubeQuart)
1662                     && !Physics.CheckBox(vecPlus10, cubeQuart))
1663                {
1664                    row10Vacancies.Add(vec);
1665                }
1666            }
1667        }
1668        foreach (Vector3 vec in row10Vacancies)
1669        {
1670            row10VacancyInts.Add(Mathf.RoundToInt(vec.z));
1671        }
1672        row10VacancyInts.Sort();
1673        //Checks each of the verified empty spaces that work for the block's rotation.
1674        //Checks if all spaces that work for that rotation are empty spaces.
1675        //Adds possible places to list openspots.
1676        //Checks each rotation based on prescribed preference.
1677        foreach (int i in row10VacancyInts)
1678        {
1679            if (i <= 3)
1680            {
1681                if (row10VacancyInts.Contains(i + 1) && row10VacancyInts.Contains(i + 2))
1682                {
1683                    openSpots.Add(i);
1684                }
1685            }
1686        }
1687        if (openSpots.Count > 0)
1688        {
1689            chosenRot = 1;
1690        }
1691        else
1692        {
1693            foreach (int i in row10VacancyInts)
1694            {
1695                if (i <= 4)
1696                {
1697                    if (row10VacancyInts.Contains(i + 1))
1698                    {
1699                        openSpots.Add(i);
1700                    }
1701                }
1702            }
1703            if (openSpots.Count > 0)
1704            {
1705                chosenRot = 0;
1706            }
```

```
1707                else
1708                {
1709                    foreach (int i in row10VacancyInts)
1710                    {
1711                        if (i >= -2)
1712                        {
1713                            Vector3 rot3ZMinus2YPlus1 = new Vector3(0.5f,
                        10.3f, ((float)i - 1.35f));
1714                            Vector3 rot3ZMinus1YPlus1 = new Vector3(0.5f,
                        10.3f, ((float)i - 2.35f));
1715                            if (!Physics.CheckBox(rot3ZMinus2YPlus1,
                        cubeQuart) && !(Physics.CheckBox(rot3ZMinus1YPlus1,
                        cubeQuart)))
1716                            {
1717                                openSpots.Add(i);
1718                            }
1719                        }
1720                    }
1721                    if (openSpots.Count > 0)
1722                    {
1723                        chosenRot = 3;
1724                    }
1725                    else
1726                    {
1727                        foreach (int i in row10VacancyInts)
1728                        {
1729                            if (i <= 4)
1730                            {
1731                                openSpots.Add(i);
1732                            }
1733                        }
1734                        if (openSpots.Count > 0)
1735                        {
1736                            chosenRot = 2;
1737                        }
1738                    }
1739                }
1740            }
1741            if (openSpots.Count == 0)
1742            {
1743                CheckRows();
1744            }
1745        }
1746        void CheckRow11()
1747        {
1748            //Check all spaces to see if there's anything there
1749            foreach (Vector3 vec in row11)
1750            {
1751                //If something isn't there, check the 6 spaces above that
```

```
                spot. If nothing's there,
1752            //add the position to the row's Vacancy list and that index
                to the row's list of indeces.
1753            if (!Physics.CheckBox(vec, cubeQuart))
1754            {
1755                Vector3 vecPlus1 = new Vector3(vec.x, vec.y + 1, vec.z);
1756                Vector3 vecPlus2 = new Vector3(vec.x, vec.y + 2, vec.z);
1757                Vector3 vecPlus3 = new Vector3(vec.x, vec.y + 3, vec.z);
1758                Vector3 vecPlus4 = new Vector3(vec.x, vec.y + 4, vec.z);
1759                Vector3 vecPlus5 = new Vector3(vec.x, vec.y + 5, vec.z);
1760                Vector3 vecPlus6 = new Vector3(vec.x, vec.y + 6, vec.z);
1761                Vector3 vecPlus7 = new Vector3(vec.x, vec.y + 7, vec.z);
1762                Vector3 vecPlus8 = new Vector3(vec.x, vec.y + 8, vec.z);
1763                Vector3 vecPlus9 = new Vector3(vec.x, vec.y + 9, vec.z);
1764                Vector3 vecPlus10 = new Vector3(vec.x, vec.y + 10,
                  vec.z);
1765
1766                if (!Physics.CheckBox(vecPlus1, cubeQuart) && !
                  Physics.CheckBox(vecPlus2, cubeQuart) && !
                  Physics.CheckBox(vecPlus3, cubeQuart)
1767                   && !Physics.CheckBox(vecPlus4, cubeQuart) && !
                   Physics.CheckBox(vecPlus5, cubeQuart) && !
                   Physics.CheckBox(vecPlus6, cubeQuart)
1768                    && !Physics.CheckBox(vecPlus7, cubeQuart) && !
                   Physics.CheckBox(vecPlus8, cubeQuart) && !
                   Physics.CheckBox(vecPlus9, cubeQuart)
1769                        && !Physics.CheckBox(vecPlus10, cubeQuart))
1770                {
1771                    row11Vacancies.Add(vec);
1772                }
1773            }
1774        }
1775        foreach (Vector3 vec in row11Vacancies)
1776        {
1777            row1VacancyInts.Add(Mathf.RoundToInt(vec.z));
1778        }
1779        row11VacancyInts.Sort();
1780        //Checks each of the verified empty spaces that work for the
            block's rotation.
1781        //Checks if all spaces that work for that rotation are empty
            spaces.
1782        //Adds possible places to list openspots.
1783        //Checks each rotation based on prescribed preference.
1784        foreach (int i in row11VacancyInts)
1785        {
1786            if (i <= 3)
1787            {
1788                if (row11VacancyInts.Contains(i + 1) &&
                  row11VacancyInts.Contains(i + 2))
```

```
1789                    {
1790                        openSpots.Add(i);
1791                    }
1792                }
1793            }
1794            if (openSpots.Count > 0)
1795            {
1796                chosenRot = 1;
1797            }
1798            else
1799            {
1800                foreach (int i in row11VacancyInts)
1801                {
1802                    if (i <= 4)
1803                    {
1804                        if (row11VacancyInts.Contains(i + 1))
1805                        {
1806                            openSpots.Add(i);
1807                        }
1808                    }
1809                }
1810                if (openSpots.Count > 0)
1811                {
1812                    chosenRot = 0;
1813                }
1814                else
1815                {
1816                    foreach (int i in row11VacancyInts)
1817                    {
1818                        if (i >= -2)
1819                        {
1820                            Vector3 rot3ZMinus2YPlus1 = new Vector3(0.5f,
                      11.3f, ((float)i - 1.35f));
1821                            Vector3 rot3ZMinus1YPlus1 = new Vector3(0.5f,
                      11.3f, ((float)i - 2.35f));
1822                            if (!Physics.CheckBox(rot3ZMinus2YPlus1,
                      cubeQuart) && !(Physics.CheckBox(rot3ZMinus1YPlus1,
                      cubeQuart)))
1823                            {
1824                                openSpots.Add(i);
1825                            }
1826                        }
1827                    }
1828                    if (openSpots.Count > 0)
1829                    {
1830                        chosenRot = 3;
1831                    }
1832                    else
1833                    {
```

```
1834                    foreach (int i in row11VacancyInts)
1835                    {
1836                        if (i <= 4)
1837                        {
1838                            openSpots.Add(i);
1839                        }
1840                    }
1841                    if (openSpots.Count > 0)
1842                    {
1843                        chosenRot = 2;
1844                    }
1845                }
1846            }
1847        }
1848        if (openSpots.Count == 0)
1849        {
1850            CheckRows();
1851        }
1852    }
1853    void CheckRow12()
1854    {
1855        //Check all spaces to see if there's anything there
1856        foreach (Vector3 vec in row12)
1857        {
1858            //If something isn't there, check the 6 spaces above that
                  spot. If nothing's there,
1859            //add the position to the row's Vacancy list and that index
                  to the row's list of indeces.
1860            if (!Physics.CheckBox(vec, cubeQuart))
1861            {
1862                Vector3 vecPlus1 = new Vector3(vec.x, vec.y + 1, vec.z);
1863                Vector3 vecPlus2 = new Vector3(vec.x, vec.y + 2, vec.z);
1864                Vector3 vecPlus3 = new Vector3(vec.x, vec.y + 3, vec.z);
1865                Vector3 vecPlus4 = new Vector3(vec.x, vec.y + 4, vec.z);
1866                Vector3 vecPlus5 = new Vector3(vec.x, vec.y + 5, vec.z);
1867                Vector3 vecPlus6 = new Vector3(vec.x, vec.y + 6, vec.z);
1868                Vector3 vecPlus7 = new Vector3(vec.x, vec.y + 7, vec.z);
1869                Vector3 vecPlus8 = new Vector3(vec.x, vec.y + 8, vec.z);
1870                Vector3 vecPlus9 = new Vector3(vec.x, vec.y + 9, vec.z);
1871                Vector3 vecPlus10 = new Vector3(vec.x, vec.y + 10,
                      vec.z);
1872
1873                if (!Physics.CheckBox(vecPlus1, cubeQuart) && !
                      Physics.CheckBox(vecPlus2, cubeQuart) && !
                      Physics.CheckBox(vecPlus3, cubeQuart)
1874                       && !Physics.CheckBox(vecPlus4, cubeQuart) && !
                       Physics.CheckBox(vecPlus5, cubeQuart) && !
                       Physics.CheckBox(vecPlus6, cubeQuart)
1875                        && !Physics.CheckBox(vecPlus7, cubeQuart) && !
```

```
                    Physics.CheckBox(vecPlus8, cubeQuart) && !
                    Physics.CheckBox(vecPlus9, cubeQuart)
1876                     && !Physics.CheckBox(vecPlus10, cubeQuart))
1877                {
1878                    row12Vacancies.Add(vec);
1879                }
1880            }
1881        }
1882        foreach (Vector3 vec in row12Vacancies)
1883        {
1884            row12VacancyInts.Add(Mathf.RoundToInt(vec.z));
1885        }
1886        row12VacancyInts.Sort();
1887        //Checks each of the verified empty spaces that work for the
              block's rotation.
1888        //Checks if all spaces that work for that rotation are empty
              spaces.
1889        //Adds possible places to list openspots.
1890        //Checks each rotation based on prescribed preference.
1891        foreach (int i in row12VacancyInts)
1892        {
1893            if (i <= 3)
1894            {
1895                if (row12VacancyInts.Contains(i + 1) &&
                      row12VacancyInts.Contains(i + 2))
1896                {
1897                    openSpots.Add(i);
1898                }
1899            }
1900        }
1901        if (openSpots.Count > 0)
1902        {
1903            chosenRot = 1;
1904        }
1905        else
1906        {
1907            foreach (int i in row12VacancyInts)
1908            {
1909                if (i <= 4)
1910                {
1911                    if (row12VacancyInts.Contains(i + 1))
1912                    {
1913                        openSpots.Add(i);
1914                    }
1915                }
1916            }
1917            if (openSpots.Count > 0)
1918            {
1919                chosenRot = 0;
```

```
1920                }
1921                else
1922                {
1923                    foreach (int i in row12VacancyInts)
1924                    {
1925                        if (i >= -2)
1926                        {
1927                            Vector3 rot3ZMinus2YPlus1 = new Vector3(0.5f,
                      12.3f, ((float)i - 1.35f));
1928                            Vector3 rot3ZMinus1YPlus1 = new Vector3(0.5f,
                      12.3f, ((float)i - 2.35f));
1929                            if (!Physics.CheckBox(rot3ZMinus2YPlus1,
                      cubeQuart) && !(Physics.CheckBox(rot3ZMinus1YPlus1,
                      cubeQuart)))
1930                            {
1931                                openSpots.Add(i);
1932                            }
1933                        }
1934                    }
1935                    if (openSpots.Count > 0)
1936                    {
1937                        chosenRot = 3;
1938                    }
1939                    else
1940                    {
1941                        foreach (int i in row12VacancyInts)
1942                        {
1943                            if (i <= 4)
1944                            {
1945                                openSpots.Add(i);
1946                            }
1947                        }
1948                        if (openSpots.Count > 0)
1949                        {
1950                            chosenRot = 2;
1951                        }
1952                    }
1953                }
1954            }
1955            if (openSpots.Count == 0)
1956            {
1957                CheckRows();
1958            }
1959        }
1960        void CheckRow13()
1961        {
1962            //Check all spaces to see if there's anything there
1963            foreach (Vector3 vec in row13)
1964            {
```

```
1965            //If something isn't there, check the 6 spaces above that spot. If nothing's there,
1966            //add the position to the row's Vacancy list and that index to the row's list of indeces.
1967            if (!Physics.CheckBox(vec, cubeQuart))
1968            {
1969                Vector3 vecPlus1 = new Vector3(vec.x, vec.y + 1, vec.z);
1970                Vector3 vecPlus2 = new Vector3(vec.x, vec.y + 2, vec.z);
1971                Vector3 vecPlus3 = new Vector3(vec.x, vec.y + 3, vec.z);
1972                Vector3 vecPlus4 = new Vector3(vec.x, vec.y + 4, vec.z);
1973                Vector3 vecPlus5 = new Vector3(vec.x, vec.y + 5, vec.z);
1974                Vector3 vecPlus6 = new Vector3(vec.x, vec.y + 6, vec.z);
1975                Vector3 vecPlus7 = new Vector3(vec.x, vec.y + 7, vec.z);
1976                Vector3 vecPlus8 = new Vector3(vec.x, vec.y + 8, vec.z);
1977                Vector3 vecPlus9 = new Vector3(vec.x, vec.y + 9, vec.z);
1978                Vector3 vecPlus10 = new Vector3(vec.x, vec.y + 10, vec.z);
1979
1980                if (!Physics.CheckBox(vecPlus1, cubeQuart) && !Physics.CheckBox(vecPlus2, cubeQuart) && !Physics.CheckBox(vecPlus3, cubeQuart)
1981                   && !Physics.CheckBox(vecPlus4, cubeQuart) && !Physics.CheckBox(vecPlus5, cubeQuart) && !Physics.CheckBox(vecPlus6, cubeQuart)
1982                    && !Physics.CheckBox(vecPlus7, cubeQuart) && !Physics.CheckBox(vecPlus8, cubeQuart) && !Physics.CheckBox(vecPlus9, cubeQuart)
1983                         && !Physics.CheckBox(vecPlus10, cubeQuart))
1984                {
1985                    row13Vacancies.Add(vec);
1986                }
1987            }
1988        }
1989        foreach (Vector3 vec in row13Vacancies)
1990        {
1991            row13VacancyInts.Add(Mathf.RoundToInt(vec.z));
1992        }
1993        row13VacancyInts.Sort();
1994        //Checks each of the verified empty spaces that work for the block's rotation.
1995        //Checks if all spaces that work for that rotation are empty spaces.
1996        //Adds possible places to list openspots.
1997        //Checks each rotation based on prescribed preference.
1998        foreach (int i in row13VacancyInts)
1999        {
2000            if (i <= 3)
2001            {
2002                if (row13VacancyInts.Contains(i + 1) &&
```

```
                    row13VacancyInts.Contains(i + 2))
                {
                    openSpots.Add(i);
                }
            }
        }
        if (openSpots.Count > 0)
        {
            chosenRot = 1;
        }
        else
        {
            foreach (int i in row13VacancyInts)
            {
                if (i <= 4)
                {
                    if (row13VacancyInts.Contains(i + 1))
                    {
                        openSpots.Add(i);
                    }
                }
            }
            if (openSpots.Count > 0)
            {
                chosenRot = 0;
            }
            else
            {
                foreach (int i in row13VacancyInts)
                {
                    if (i >= -2)
                    {
                        Vector3 rot3ZMinus2YPlus1 = new Vector3(0.5f,
                    13.3f, ((float)i - 1.35f));
                        Vector3 rot3ZMinus1YPlus1 = new Vector3(0.5f,
                    13.3f, ((float)i - 2.35f));
                        if (!Physics.CheckBox(rot3ZMinus2YPlus1,
                    cubeQuart) && !(Physics.CheckBox(rot3ZMinus1YPlus1,
                    cubeQuart)))
                        {
                            openSpots.Add(i);
                        }
                    }
                }
                if (openSpots.Count > 0)
                {
                    chosenRot = 3;
                }
                else
```

```
2047                {
2048                    foreach (int i in row13VacancyInts)
2049                    {
2050                        if (i <= 4)
2051                        {
2052                            openSpots.Add(i);
2053                        }
2054                    }
2055                    if (openSpots.Count > 0)
2056                    {
2057                        chosenRot = 2;
2058                    }
2059                }
2060            }
2061        }
2062        if (openSpots.Count == 0)
2063        {
2064            CheckRows();
2065        }
2066    }
2067    void CheckRow14()
2068    {
2069        //Check all spaces to see if there's anything there
2070        foreach (Vector3 vec in row14)
2071        {
2072            //If something isn't there, check the 6 spaces above that
                  spot. If nothing's there,
2073            //add the position to the row's Vacancy list and that index
                  to the row's list of indeces.
2074            if (!Physics.CheckBox(vec, cubeQuart))
2075            {
2076                Vector3 vecPlus1 = new Vector3(vec.x, vec.y + 1, vec.z);
2077                Vector3 vecPlus2 = new Vector3(vec.x, vec.y + 2, vec.z);
2078                Vector3 vecPlus3 = new Vector3(vec.x, vec.y + 3, vec.z);
2079                Vector3 vecPlus4 = new Vector3(vec.x, vec.y + 4, vec.z);
2080                Vector3 vecPlus5 = new Vector3(vec.x, vec.y + 5, vec.z);
2081                Vector3 vecPlus6 = new Vector3(vec.x, vec.y + 6, vec.z);
2082                Vector3 vecPlus7 = new Vector3(vec.x, vec.y + 7, vec.z);
2083                Vector3 vecPlus8 = new Vector3(vec.x, vec.y + 8, vec.z);
2084                Vector3 vecPlus9 = new Vector3(vec.x, vec.y + 9, vec.z);
2085                Vector3 vecPlus10 = new Vector3(vec.x, vec.y + 10,
                      vec.z);
2086
2087                if (!Physics.CheckBox(vecPlus1, cubeQuart) && !
                      Physics.CheckBox(vecPlus2, cubeQuart) && !
                      Physics.CheckBox(vecPlus3, cubeQuart)
2088                       && !Physics.CheckBox(vecPlus4, cubeQuart) && !
                       Physics.CheckBox(vecPlus5, cubeQuart) && !
                       Physics.CheckBox(vecPlus6, cubeQuart)
```

```
2089                        && !Physics.CheckBox(vecPlus7, cubeQuart) && !
                        Physics.CheckBox(vecPlus8, cubeQuart) && !
                        Physics.CheckBox(vecPlus9, cubeQuart)
2090                        && !Physics.CheckBox(vecPlus10, cubeQuart))
2091                {
2092                    row14Vacancies.Add(vec);
2093                }
2094            }
2095        }
2096        foreach (Vector3 vec in row14Vacancies)
2097        {
2098            row4VacancyInts.Add(Mathf.RoundToInt(vec.z));
2099        }
2100        row14VacancyInts.Sort();
2101        //Checks each of the verified empty spaces that work for the
            block's rotation.
2102        //Checks if all spaces that work for that rotation are empty
            spaces.
2103        //Adds possible places to list openspots.
2104        //Checks each rotation based on prescribed preference.
2105        foreach (int i in row14VacancyInts)
2106        {
2107            if (i <= 3)
2108            {
2109                if (row14VacancyInts.Contains(i + 1) &&
                    row14VacancyInts.Contains(i + 2))
2110                {
2111                    openSpots.Add(i);
2112                }
2113            }
2114        }
2115        if (openSpots.Count > 0)
2116        {
2117            chosenRot = 1;
2118        }
2119        else
2120        {
2121            foreach (int i in row14VacancyInts)
2122            {
2123                if (i <= 4)
2124                {
2125                    if (row14VacancyInts.Contains(i + 1))
2126                    {
2127                        openSpots.Add(i);
2128                    }
2129                }
2130            }
2131            if (openSpots.Count > 0)
2132            {
```

```
2133                    chosenRot = 0;
2134                }
2135                else
2136                {
2137                    foreach (int i in row14VacancyInts)
2138                    {
2139                        if (i >= -2)
2140                        {
2141                            Vector3 rot3ZMinus2YPlus1 = new Vector3(0.5f,
                      14.3f, ((float)i - 1.35f));
2142                            Vector3 rot3ZMinus1YPlus1 = new Vector3(0.5f,
                      14.3f, ((float)i - 2.35f));
2143                            if (!Physics.CheckBox(rot3ZMinus2YPlus1,
                      cubeQuart) && !(Physics.CheckBox(rot3ZMinus1YPlus1,
                      cubeQuart)))
2144                            {
2145                                openSpots.Add(i);
2146                            }
2147                        }
2148                    }
2149                    if (openSpots.Count > 0)
2150                    {
2151                        chosenRot = 3;
2152                    }
2153                    else
2154                    {
2155                        foreach (int i in row14VacancyInts)
2156                        {
2157                            if (i <= 4)
2158                            {
2159                                openSpots.Add(i);
2160                            }
2161                        }
2162                        if (openSpots.Count > 0)
2163                        {
2164                            chosenRot = 2;
2165                        }
2166                    }
2167                }
2168            }
2169            if (openSpots.Count == 0)
2170            {
2171                CheckRows();
2172            }
2173        }
2174        void CheckRow15()
2175        {
2176            //Check all spaces to see if there's anything there
2177            foreach (Vector3 vec in row15)
```

```
2178        {
2179            //If something isn't there, check the 6 spaces above that spot. If nothing's there,
2180            //add the position to the row's Vacancy list and that index to the row's list of indeces.
2181            if (!Physics.CheckBox(vec, cubeQuart))
2182            {
2183                Vector3 vecPlus1 = new Vector3(vec.x, vec.y + 1, vec.z);
2184                Vector3 vecPlus2 = new Vector3(vec.x, vec.y + 2, vec.z);
2185                Vector3 vecPlus3 = new Vector3(vec.x, vec.y + 3, vec.z);
2186                Vector3 vecPlus4 = new Vector3(vec.x, vec.y + 4, vec.z);
2187                Vector3 vecPlus5 = new Vector3(vec.x, vec.y + 5, vec.z);
2188                Vector3 vecPlus6 = new Vector3(vec.x, vec.y + 6, vec.z);
2189                Vector3 vecPlus7 = new Vector3(vec.x, vec.y + 7, vec.z);
2190                Vector3 vecPlus8 = new Vector3(vec.x, vec.y + 8, vec.z);
2191                Vector3 vecPlus9 = new Vector3(vec.x, vec.y + 9, vec.z);
2192                Vector3 vecPlus10 = new Vector3(vec.x, vec.y + 10, vec.z);
2193
2194                if (!Physics.CheckBox(vecPlus1, cubeQuart) && !Physics.CheckBox(vecPlus2, cubeQuart) && !Physics.CheckBox(vecPlus3, cubeQuart)
2195                   && !Physics.CheckBox(vecPlus4, cubeQuart) && !Physics.CheckBox(vecPlus5, cubeQuart) && !Physics.CheckBox(vecPlus6, cubeQuart)
2196                    && !Physics.CheckBox(vecPlus7, cubeQuart) && !Physics.CheckBox(vecPlus8, cubeQuart) && !Physics.CheckBox(vecPlus9, cubeQuart)
2197                        && !Physics.CheckBox(vecPlus10, cubeQuart))
2198                {
2199                    row15Vacancies.Add(vec);
2200                }
2201            }
2202        }
2203        foreach (Vector3 vec in row15Vacancies)
2204        {
2205            row15VacancyInts.Add(Mathf.RoundToInt(vec.z));
2206        }
2207        row15VacancyInts.Sort();
2208        //Checks each of the verified empty spaces that work for the block's rotation.
2209        //Checks if all spaces that work for that rotation are empty spaces.
2210        //Adds possible places to list openspots.
2211        //Checks each rotation based on prescribed preference.
2212        foreach (int i in row15VacancyInts)
2213        {
2214            if (i <= 3)
2215            {
```

```
2216                    if (row15VacancyInts.Contains(i + 1) &&
                          row15VacancyInts.Contains(i + 2))
2217                    {
2218                        openSpots.Add(i);
2219                    }
2220                }
2221            }
2222            if (openSpots.Count > 0)
2223            {
2224                chosenRot = 1;
2225            }
2226            else
2227            {
2228                foreach (int i in row15VacancyInts)
2229                {
2230                    if (i <= 4)
2231                    {
2232                        if (row15VacancyInts.Contains(i + 1))
2233                        {
2234                            openSpots.Add(i);
2235                        }
2236                    }
2237                }
2238                if (openSpots.Count > 0)
2239                {
2240                    chosenRot = 0;
2241                }
2242                else
2243                {
2244                    foreach (int i in row15VacancyInts)
2245                    {
2246                        if (i >= -2)
2247                        {
2248                            Vector3 rot3ZMinus2YPlus1 = new Vector3(0.5f,
                              15.3f, ((float)i - 1.35f));
2249                            Vector3 rot3ZMinus1YPlus1 = new Vector3(0.5f,
                              15.3f, ((float)i - 2.35f));
2250                            if (!Physics.CheckBox(rot3ZMinus2YPlus1,
                              cubeQuart) && !(Physics.CheckBox(rot3ZMinus1YPlus1,
                              cubeQuart)))
2251                            {
2252                                openSpots.Add(i);
2253                            }
2254                        }
2255                    }
2256                    if (openSpots.Count > 0)
2257                    {
2258                        chosenRot = 3;
2259                    }
```

```
2260                 else
2261                 {
2262                     foreach (int i in row15VacancyInts)
2263                     {
2264                         if (i <= 4)
2265                         {
2266                             openSpots.Add(i);
2267                         }
2268                     }
2269                     if (openSpots.Count > 0)
2270                     {
2271                         chosenRot = 2;
2272                     }
2273                 }
2274             }
2275         }
2276         if (openSpots.Count == 0)
2277         {
2278             CheckRows();
2279         }
2280     }
2281     void CheckRow16()
2282     {
2283         //Check all spaces to see if there's anything there
2284         foreach (Vector3 vec in row16)
2285         {
2286             //If something isn't there, check the 6 spaces above that
                    spot. If nothing's there,
2287             //add the position to the row's Vacancy list and that index
                    to the row's list of indeces.
2288             if (!Physics.CheckBox(vec, cubeQuart))
2289             {
2290                 Vector3 vecPlus1 = new Vector3(vec.x, vec.y + 1, vec.z);
2291                 Vector3 vecPlus2 = new Vector3(vec.x, vec.y + 2, vec.z);
2292                 Vector3 vecPlus3 = new Vector3(vec.x, vec.y + 3, vec.z);
2293                 Vector3 vecPlus4 = new Vector3(vec.x, vec.y + 4, vec.z);
2294                 Vector3 vecPlus5 = new Vector3(vec.x, vec.y + 5, vec.z);
2295                 Vector3 vecPlus6 = new Vector3(vec.x, vec.y + 6, vec.z);
2296                 Vector3 vecPlus7 = new Vector3(vec.x, vec.y + 7, vec.z);
2297                 Vector3 vecPlus8 = new Vector3(vec.x, vec.y + 8, vec.z);
2298                 Vector3 vecPlus9 = new Vector3(vec.x, vec.y + 9, vec.z);
2299                 Vector3 vecPlus10 = new Vector3(vec.x, vec.y + 10,
                      vec.z);
2300
2301                 if (!Physics.CheckBox(vecPlus1, cubeQuart) && !
                      Physics.CheckBox(vecPlus2, cubeQuart) && !
                      Physics.CheckBox(vecPlus3, cubeQuart)
2302                        && !Physics.CheckBox(vecPlus4, cubeQuart) && !
                       Physics.CheckBox(vecPlus5, cubeQuart) && !
```

```
                    Physics.CheckBox(vecPlus6, cubeQuart)
2303                    && !Physics.CheckBox(vecPlus7, cubeQuart) && !
                    Physics.CheckBox(vecPlus8, cubeQuart) && !
                    Physics.CheckBox(vecPlus9, cubeQuart)
2304                    && !Physics.CheckBox(vecPlus10, cubeQuart))
2305                {
2306                    row16Vacancies.Add(vec);
2307                }
2308            }
2309        }
2310        foreach (Vector3 vec in row16Vacancies)
2311        {
2312            row16VacancyInts.Add(Mathf.RoundToInt(vec.z));
2313        }
2314        row16VacancyInts.Sort();
2315        //Checks each of the verified empty spaces that work for the
              block's rotation.
2316        //Checks if all spaces that work for that rotation are empty
              spaces.
2317        //Adds possible places to list openspots.
2318        //Checks each rotation based on prescribed preference.
2319        foreach (int i in row16VacancyInts)
2320        {
2321            if (i <= 3)
2322            {
2323                if (row16VacancyInts.Contains(i + 1) &&
                      row16VacancyInts.Contains(i + 2))
2324                {
2325                    openSpots.Add(i);
2326                }
2327            }
2328        }
2329        if (openSpots.Count > 0)
2330        {
2331            chosenRot = 1;
2332        }
2333        else
2334        {
2335            foreach (int i in row16VacancyInts)
2336            {
2337                if (i <= 4)
2338                {
2339                    if (row16VacancyInts.Contains(i + 1))
2340                    {
2341                        openSpots.Add(i);
2342                    }
2343                }
2344            }
2345            if (openSpots.Count > 0)
```

```
2346                {
2347                    chosenRot = 0;
2348                }
2349                else
2350                {
2351                    foreach (int i in row16VacancyInts)
2352                    {
2353                        if (i >= -2)
2354                        {
2355                            Vector3 rot3ZMinus2YPlus1 = new Vector3(0.5f,
                        16.3f, ((float)i - 1.35f));
2356                            Vector3 rot3ZMinus1YPlus1 = new Vector3(0.5f,
                        16.3f, ((float)i - 2.35f));
2357                            if (!Physics.CheckBox(rot3ZMinus2YPlus1,
                        cubeQuart) && !(Physics.CheckBox(rot3ZMinus1YPlus1,
                        cubeQuart)))
2358                            {
2359                                openSpots.Add(i);
2360                            }
2361                        }
2362                    }
2363                    if (openSpots.Count > 0)
2364                    {
2365                        chosenRot = 3;
2366                    }
2367                    else
2368                    {
2369                        foreach (int i in row16VacancyInts)
2370                        {
2371                            if (i <= 4)
2372                            {
2373                                openSpots.Add(i);
2374                            }
2375                        }
2376                        if (openSpots.Count > 0)
2377                        {
2378                            chosenRot = 2;
2379                        }
2380                    }
2381                }
2382            }
2383            if (openSpots.Count == 0)
2384            {
2385                CheckRows();
2386            }
2387        }
2388        void CheckRow17()
2389        {
2390            //Check all spaces to see if there's anything there
```

```
2391            foreach (Vector3 vec in row17)
2392            {
2393                //If something isn't there, check the 6 spaces above that
                      spot. If nothing's there,
2394                //add the position to the row's Vacancy list and that index
                      to the row's list of indeces.
2395                if (!Physics.CheckBox(vec, cubeQuart))
2396                {
2397                    Vector3 vecPlus1 = new Vector3(vec.x, vec.y + 1, vec.z);
2398                    Vector3 vecPlus2 = new Vector3(vec.x, vec.y + 2, vec.z);
2399                    Vector3 vecPlus3 = new Vector3(vec.x, vec.y + 3, vec.z);
2400                    Vector3 vecPlus4 = new Vector3(vec.x, vec.y + 4, vec.z);
2401                    Vector3 vecPlus5 = new Vector3(vec.x, vec.y + 5, vec.z);
2402                    Vector3 vecPlus6 = new Vector3(vec.x, vec.y + 6, vec.z);
2403                    Vector3 vecPlus7 = new Vector3(vec.x, vec.y + 7, vec.z);
2404                    Vector3 vecPlus8 = new Vector3(vec.x, vec.y + 8, vec.z);
2405                    Vector3 vecPlus9 = new Vector3(vec.x, vec.y + 9, vec.z);
2406                    Vector3 vecPlus10 = new Vector3(vec.x, vec.y + 10,
                          vec.z);
2407
2408                    if (!Physics.CheckBox(vecPlus1, cubeQuart) && !
                          Physics.CheckBox(vecPlus2, cubeQuart) && !
                          Physics.CheckBox(vecPlus3, cubeQuart)
2409                           && !Physics.CheckBox(vecPlus4, cubeQuart) && !
                           Physics.CheckBox(vecPlus5, cubeQuart) && !
                           Physics.CheckBox(vecPlus6, cubeQuart)
2410                            && !Physics.CheckBox(vecPlus7, cubeQuart) && !
                           Physics.CheckBox(vecPlus8, cubeQuart) && !
                           Physics.CheckBox(vecPlus9, cubeQuart)
2411                                 && !Physics.CheckBox(vecPlus10, cubeQuart))
2412                    {
2413                        row17Vacancies.Add(vec);
2414                    }
2415                }
2416            }
2417            foreach (Vector3 vec in row17Vacancies)
2418            {
2419                row17VacancyInts.Add(Mathf.RoundToInt(vec.z));
2420            }
2421            row17VacancyInts.Sort();
2422            //Checks each of the verified empty spaces that work for the
                  block's rotation.
2423            //Checks if all spaces that work for that rotation are empty
                  spaces.
2424            //Adds possible places to list openspots.
2425            //Checks each rotation based on prescribed preference.
2426            foreach (int i in row17VacancyInts)
2427            {
2428                if (i <= 3)
```

```
2429                {
2430                    if (row17VacancyInts.Contains(i + 1) &&
                          row17VacancyInts.Contains(i + 2))
2431                    {
2432                        openSpots.Add(i);
2433                    }
2434                }
2435            }
2436            if (openSpots.Count > 0)
2437            {
2438                chosenRot = 1;
2439            }
2440            else
2441            {
2442                foreach (int i in row17VacancyInts)
2443                {
2444                    if (i <= 4)
2445                    {
2446                        if (row17VacancyInts.Contains(i + 1))
2447                        {
2448                            openSpots.Add(i);
2449                        }
2450                    }
2451                }
2452                if (openSpots.Count > 0)
2453                {
2454                    chosenRot = 0;
2455                }
2456                else
2457                {
2458                    foreach (int i in row17VacancyInts)
2459                    {
2460                        if (i >= -2)
2461                        {
2462                            Vector3 rot3ZMinus2YPlus1 = new Vector3(0.5f,
                              17.3f, ((float)i - 1.35f));
2463                            Vector3 rot3ZMinus1YPlus1 = new Vector3(0.5f,
                              17.3f, ((float)i - 2.35f));
2464                            if (!Physics.CheckBox(rot3ZMinus2YPlus1,
                              cubeQuart) && !(Physics.CheckBox(rot3ZMinus1YPlus1,
                              cubeQuart)))
2465                            {
2466                                openSpots.Add(i);
2467                            }
2468                        }
2469                    }
2470                    if (openSpots.Count > 0)
2471                    {
2472                        chosenRot = 3;
```

```
2473                }
2474                else
2475                {
2476                    foreach (int i in row17VacancyInts)
2477                    {
2478                        if (i <= 4)
2479                        {
2480                            openSpots.Add(i);
2481                        }
2482                    }
2483                    if (openSpots.Count > 0)
2484                    {
2485                        chosenRot = 2;
2486                    }
2487                }
2488            }
2489        }
2490        if (openSpots.Count == 0)
2491        {
2492            CheckRows();
2493        }
2494    }
2495    void CheckRow18()
2496    {
2497        //Check all spaces to see if there's anything there
2498        foreach (Vector3 vec in row18)
2499        {
2500            //If something isn't there, check the 6 spaces above that
                  spot. If nothing's there,
2501            //add the position to the row's Vacancy list and that index
                  to the row's list of indeces.
2502            if (!Physics.CheckBox(vec, cubeQuart))
2503            {
2504                Vector3 vecPlus1 = new Vector3(vec.x, vec.y + 1, vec.z);
2505                Vector3 vecPlus2 = new Vector3(vec.x, vec.y + 2, vec.z);
2506                Vector3 vecPlus3 = new Vector3(vec.x, vec.y + 3, vec.z);
2507                Vector3 vecPlus4 = new Vector3(vec.x, vec.y + 4, vec.z);
2508                Vector3 vecPlus5 = new Vector3(vec.x, vec.y + 5, vec.z);
2509                Vector3 vecPlus6 = new Vector3(vec.x, vec.y + 6, vec.z);
2510                Vector3 vecPlus7 = new Vector3(vec.x, vec.y + 7, vec.z);
2511                Vector3 vecPlus8 = new Vector3(vec.x, vec.y + 8, vec.z);
2512                Vector3 vecPlus9 = new Vector3(vec.x, vec.y + 9, vec.z);
2513                Vector3 vecPlus10 = new Vector3(vec.x, vec.y + 10,
                      vec.z);
2514
2515                if (!Physics.CheckBox(vecPlus1, cubeQuart) && !
                      Physics.CheckBox(vecPlus2, cubeQuart) && !
                      Physics.CheckBox(vecPlus3, cubeQuart)
2516                       && !Physics.CheckBox(vecPlus4, cubeQuart) && !
```

```
                    Physics.CheckBox(vecPlus5, cubeQuart) && !
                    Physics.CheckBox(vecPlus6, cubeQuart)
2517                    && !Physics.CheckBox(vecPlus7, cubeQuart) && !
                    Physics.CheckBox(vecPlus8, cubeQuart) && !
                    Physics.CheckBox(vecPlus9, cubeQuart)
2518                    && !Physics.CheckBox(vecPlus10, cubeQuart))
2519                {
2520                    row18Vacancies.Add(vec);
2521                }
2522            }
2523        }
2524        foreach (Vector3 vec in row18Vacancies)
2525        {
2526            row18VacancyInts.Add(Mathf.RoundToInt(vec.z));
2527        }
2528        row18VacancyInts.Sort();
2529        //Checks each of the verified empty spaces that work for the
              block's rotation.
2530        //Checks if all spaces that work for that rotation are empty
              spaces.
2531        //Adds possible places to list openspots.
2532        //Checks each rotation based on prescribed preference.
2533        foreach (int i in row18VacancyInts)
2534        {
2535            if (i <= 3)
2536            {
2537                if (row18VacancyInts.Contains(i + 1) &&
                      row18VacancyInts.Contains(i + 2))
2538                {
2539                    openSpots.Add(i);
2540                }
2541            }
2542        }
2543        if (openSpots.Count > 0)
2544        {
2545            chosenRot = 1;
2546        }
2547        else
2548        {
2549            foreach (int i in row18VacancyInts)
2550            {
2551                if (i <= 4)
2552                {
2553                    if (row18VacancyInts.Contains(i + 1))
2554                    {
2555                        openSpots.Add(i);
2556                    }
2557                }
2558            }
```

```
2559                if (openSpots.Count > 0)
2560                {
2561                    chosenRot = 0;
2562                }
2563                else
2564                {
2565                    foreach (int i in row18VacancyInts)
2566                    {
2567                        if (i >= -2)
2568                        {
2569                            Vector3 rot3ZMinus2YPlus1 = new Vector3(0.5f,
                      18.3f, ((float)i - 1.35f));
2570                            Vector3 rot3ZMinus1YPlus1 = new Vector3(0.5f,
                      18.3f, ((float)i - 2.35f));
2571                            if (!Physics.CheckBox(rot3ZMinus2YPlus1,
                      cubeQuart) && !(Physics.CheckBox(rot3ZMinus1YPlus1,
                      cubeQuart)))
2572                            {
2573                                openSpots.Add(i);
2574                            }
2575                        }
2576                    }
2577                    if (openSpots.Count > 0)
2578                    {
2579                        chosenRot = 3;
2580                    }
2581                    else
2582                    {
2583                        foreach (int i in row18VacancyInts)
2584                        {
2585                            if (i <= 4)
2586                            {
2587                                openSpots.Add(i);
2588                            }
2589                        }
2590                        if (openSpots.Count > 0)
2591                        {
2592                            chosenRot = 2;
2593                        }
2594                    }
2595                }
2596            }
2597            if (openSpots.Count == 0)
2598            {
2599                CheckRows();
2600            }
2601        }
2602        void CheckRow19()
2603        {
```

```
2604          //Check all spaces to see if there's anything there
2605          foreach (Vector3 vec in row19)
2606          {
2607              //If something isn't there, check the 6 spaces above that
                    spot. If nothing's there,
2608              //add the position to the row's Vacancy list and that index
                    to the row's list of indeces.
2609              if (!Physics.CheckBox(vec, cubeQuart))
2610              {
2611                  Vector3 vecPlus1 = new Vector3(vec.x, vec.y + 1, vec.z);
2612                  Vector3 vecPlus2 = new Vector3(vec.x, vec.y + 2, vec.z);
2613                  Vector3 vecPlus3 = new Vector3(vec.x, vec.y + 3, vec.z);
2614                  Vector3 vecPlus4 = new Vector3(vec.x, vec.y + 4, vec.z);
2615                  Vector3 vecPlus5 = new Vector3(vec.x, vec.y + 5, vec.z);
2616                  Vector3 vecPlus6 = new Vector3(vec.x, vec.y + 6, vec.z);
2617                  Vector3 vecPlus7 = new Vector3(vec.x, vec.y + 7, vec.z);
2618                  Vector3 vecPlus8 = new Vector3(vec.x, vec.y + 8, vec.z);
2619                  Vector3 vecPlus9 = new Vector3(vec.x, vec.y + 9, vec.z);
2620                  Vector3 vecPlus10 = new Vector3(vec.x, vec.y + 10,
                        vec.z);
2621
2622                  if (!Physics.CheckBox(vecPlus1, cubeQuart) && !
                        Physics.CheckBox(vecPlus2, cubeQuart) && !
                        Physics.CheckBox(vecPlus3, cubeQuart)
2623                         && !Physics.CheckBox(vecPlus4, cubeQuart) && !
                        Physics.CheckBox(vecPlus5, cubeQuart) && !
                        Physics.CheckBox(vecPlus6, cubeQuart)
2624                             && !Physics.CheckBox(vecPlus7, cubeQuart) && !
                        Physics.CheckBox(vecPlus8, cubeQuart) && !
                        Physics.CheckBox(vecPlus9, cubeQuart)
2625                                 && !Physics.CheckBox(vecPlus10, cubeQuart))
2626                  {
2627                      row19Vacancies.Add(vec);
2628                  }
2629              }
2630          }
2631          foreach (Vector3 vec in row19Vacancies)
2632          {
2633              row19VacancyInts.Add(Mathf.RoundToInt(vec.z));
2634          }
2635          row19VacancyInts.Sort();
2636          //Checks each of the verified empty spaces that work for the
                block's rotation.
2637          //Checks if all spaces that work for that rotation are empty
                spaces.
2638          //Adds possible places to list openspots.
2639          //Checks each rotation based on prescribed preference.
2640          foreach (int i in row19VacancyInts)
2641          {
```

```
2642                if (i <= 3)
2643                {
2644                    if (row19VacancyInts.Contains(i + 1) &&
                          row19VacancyInts.Contains(i + 2))
2645                    {
2646                        openSpots.Add(i);
2647                    }
2648                }
2649            }
2650            if (openSpots.Count > 0)
2651            {
2652                chosenRot = 1;
2653            }
2654            else
2655            {
2656                foreach (int i in row19VacancyInts)
2657                {
2658                    if (i <= 4)
2659                    {
2660                        if (row19VacancyInts.Contains(i + 1))
2661                        {
2662                            openSpots.Add(i);
2663                        }
2664                    }
2665                }
2666                if (openSpots.Count > 0)
2667                {
2668                    chosenRot = 0;
2669                }
2670                else
2671                {
2672                    foreach (int i in row19VacancyInts)
2673                    {
2674                        if (i >= -2)
2675                        {
2676                            Vector3 rot3ZMinus2YPlus1 = new Vector3(0.5f,
                              19.3f, ((float)i - 1.35f));
2677                            Vector3 rot3ZMinus1YPlus1 = new Vector3(0.5f,
                              19.3f, ((float)i - 2.35f));
2678                            if (!Physics.CheckBox(rot3ZMinus2YPlus1,
                              cubeQuart) && !(Physics.CheckBox(rot3ZMinus1YPlus1,
                              cubeQuart)))
2679                            {
2680                                openSpots.Add(i);
2681                            }
2682                        }
2683                    }
2684                    if (openSpots.Count > 0)
2685                    {
```

```
2686                        chosenRot = 3;
2687                    }
2688                    else
2689                    {
2690                        foreach (int i in row19VacancyInts)
2691                        {
2692                            if (i <= 4)
2693                            {
2694                                openSpots.Add(i);
2695                            }
2696                        }
2697                        if (openSpots.Count > 0)
2698                        {
2699                            chosenRot = 2;
2700                        }
2701                    }
2702                }
2703            }
2704            if (openSpots.Count == 0)
2705            {
2706                CheckRows();
2707            }
2708        }
2709        void CheckRow20()
2710        {
2711            //Check all spaces to see if there's anything there
2712            foreach (Vector3 vec in row20)
2713            {
2714                //If something isn't there, check the 6 spaces above that
                      spot. If nothing's there,
2715                //add the position to the row's Vacancy list and that index
                      to the row's list of indeces.
2716                if (!Physics.CheckBox(vec, cubeQuart))
2717                {
2718                    Vector3 vecPlus1 = new Vector3(vec.x, vec.y + 1, vec.z);
2719                    Vector3 vecPlus2 = new Vector3(vec.x, vec.y + 2, vec.z);
2720                    Vector3 vecPlus3 = new Vector3(vec.x, vec.y + 3, vec.z);
2721                    Vector3 vecPlus4 = new Vector3(vec.x, vec.y + 4, vec.z);
2722                    Vector3 vecPlus5 = new Vector3(vec.x, vec.y + 5, vec.z);
2723                    Vector3 vecPlus6 = new Vector3(vec.x, vec.y + 6, vec.z);
2724                    Vector3 vecPlus7 = new Vector3(vec.x, vec.y + 7, vec.z);
2725                    Vector3 vecPlus8 = new Vector3(vec.x, vec.y + 8, vec.z);
2726                    Vector3 vecPlus9 = new Vector3(vec.x, vec.y + 9, vec.z);
2727                    Vector3 vecPlus10 = new Vector3(vec.x, vec.y + 10,
                          vec.z);
2728
2729                    if (!Physics.CheckBox(vecPlus1, cubeQuart) && !
                          Physics.CheckBox(vecPlus2, cubeQuart) && !
                          Physics.CheckBox(vecPlus3, cubeQuart)
```

```
2730                        && !Physics.CheckBox(vecPlus4, cubeQuart) && !
                        Physics.CheckBox(vecPlus5, cubeQuart) && !
                        Physics.CheckBox(vecPlus6, cubeQuart)
2731                        && !Physics.CheckBox(vecPlus7, cubeQuart) && !
                        Physics.CheckBox(vecPlus8, cubeQuart) && !
                        Physics.CheckBox(vecPlus9, cubeQuart)
2732                        && !Physics.CheckBox(vecPlus10, cubeQuart))
2733                {
2734                    row20Vacancies.Add(vec);
2735                }
2736            }
2737        }
2738        foreach (Vector3 vec in row20Vacancies)
2739        {
2740            row20VacancyInts.Add(Mathf.RoundToInt(vec.z));
2741        }
2742        row20VacancyInts.Sort();
2743        //Checks each of the verified empty spaces that work for the
            block's rotation.
2744        //Checks if all spaces that work for that rotation are empty
            spaces.
2745        //Adds possible places to list openspots.
2746        //Checks each rotation based on prescribed preference.
2747        foreach (int i in row20VacancyInts)
2748        {
2749            if (i <= 3)
2750            {
2751                if (row20VacancyInts.Contains(i + 1) &&
                  row20VacancyInts.Contains(i + 2))
2752                {
2753                    openSpots.Add(i);
2754                }
2755            }
2756        }
2757        if (openSpots.Count > 0)
2758        {
2759            chosenRot = 1;
2760        }
2761        else
2762        {
2763            foreach (int i in row20VacancyInts)
2764            {
2765                if (i <= 4)
2766                {
2767                    if (row20VacancyInts.Contains(i + 1))
2768                    {
2769                        openSpots.Add(i);
2770                    }
2771                }
```

```
2772                }
2773                if (openSpots.Count > 0)
2774                {
2775                    chosenRot = 0;
2776                }
2777                else
2778                {
2779                    foreach (int i in row20VacancyInts)
2780                    {
2781                        if (i >= -2)
2782                        {
2783                            Vector3 rot3ZMinus2YPlus1 = new Vector3(0.5f,
                        20.3f, ((float)i - 1.35f));
2784                            Vector3 rot3ZMinus1YPlus1 = new Vector3(0.5f,
                        20.3f, ((float)i - 2.35f));
2785                            if (!Physics.CheckBox(rot3ZMinus2YPlus1,
                        cubeQuart) && !(Physics.CheckBox(rot3ZMinus1YPlus1,
                        cubeQuart)))
2786                            {
2787                                openSpots.Add(i);
2788                            }
2789                        }
2790                    }
2791                    if (openSpots.Count > 0)
2792                    {
2793                        chosenRot = 3;
2794                    }
2795                    else
2796                    {
2797                        foreach (int i in row20VacancyInts)
2798                        {
2799                            if (i <= 4)
2800                            {
2801                                openSpots.Add(i);
2802                            }
2803                        }
2804                        if (openSpots.Count > 0)
2805                        {
2806                            chosenRot = 2;
2807                        }
2808                    }
2809                }
2810            }
2811            if (openSpots.Count == 0)
2812            {
2813                CheckRows();
2814            }
2815        }
2816        //Chooses preferred spots based on the block's rotation
```

```
2817      void PrefSpots()
2818      {
2819          foreach (int i in openSpots)
2820          {
2821              switch (chosenRot)
2822              {
2823                  case 0:
2824                      if (i % 2 == 0)
2825                      {
2826                          preferredSpots.Add(i);
2827                      }
2828                      break;
2829                  case 1:
2830                      if (i % 2 != 0)
2831                      {
2832                          preferredSpots.Add(i);
2833                      }
2834                      if (i == -4)
2835                      {
2836                          preferredSpots.Add(i);
2837                      }
2838                      break;
2839                  default:
2840                      break;
2841              }
2842          }
2843      }
2844      //Chooses the spot to move to.
2845      void ChooseSpot()
2846      {
2847          if (preferredSpots.Count > 0)
2848          {
2849              spot = preferredSpots[Random.Range(0, preferredSpots.Count)];
2850          }
2851          else if (openSpots.Count > 0)
2852          {
2853              spot = openSpots[Random.Range(0, openSpots.Count)];
2854              if (chosenRot == 3)
2855              {
2856                  spot -= 2;
2857              }
2858          }
2859      }
2860      //Rotates to the chosen rotation.
2861      //Moves to the chosen spot
2862      //Moves down quickly when the spot is chosen.
2863      //Each action is given a small timer break.
2864      void GoToSpot()
2865      {
```

```
2866          if (chosenRot != myRot)
2867          {
2868              Rotate();
2869              timerTime = Random.Range(0.1f, 0.3f);
2870          }
2871          else {
2872              if (mySpot > spot)
2873              {
2874                  MoveLeft();
2875                  mySpot--;
2876                  timerTime = Random.Range(0.2f, 0.6f);
2877              }
2878
2879              else if (mySpot < spot)
2880              {
2881              MoveRight();
2882              mySpot++;
2883              timerTime = Random.Range(0.2f, 0.6f);
2884              }
2885              else
2886              {
2887                  if (downed == false)
2888                  {
2889                      downed = true;
2890                      MoveDown();
2891                      Destroy(this);
2892                  }
2893              }
2894          }
2895      }
2896      void MoveRight()
2897      {
2898          gameObject.GetComponent<Ell1_Movement>().MoveRight();
2899      }
2900
2901      void MoveLeft()
2902      {
2903          gameObject.GetComponent<Ell1_Movement>().MoveLeft();
2904      }
2905
2906      void MoveDown()
2907      {
2908          gameObject.GetComponent<Ell1_Movement>().MoveDown();
2909      }
2910
2911      void Rotate()
2912      {
2913          gameObject.GetComponent<Ell1_Movement>().Rotate();
2914      }
```

```
2915  }
```