

CT3536 Games Programming - Game Dev Project

Daniel Hannon (19484286)

January 2022

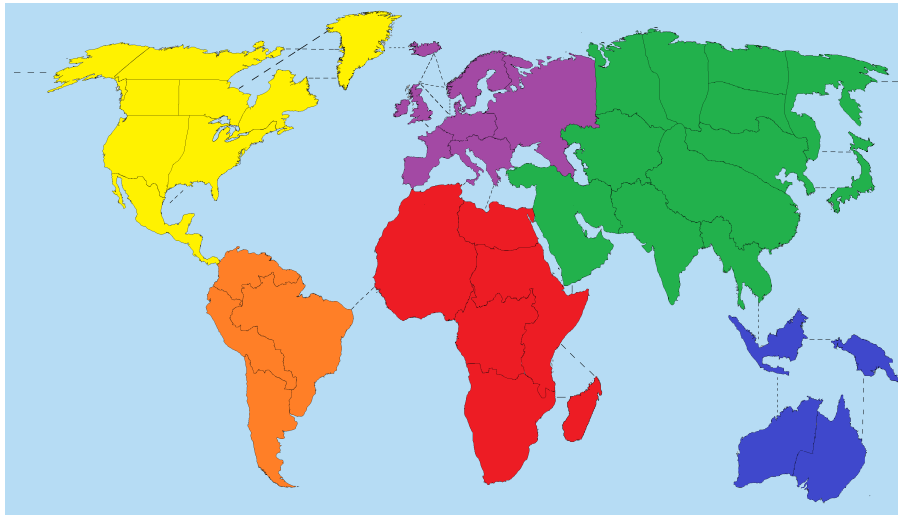
1 Overview

The game I wanted to make was Risk. The reason I wanted to do this was that I do not like the Steam Version of the game, linked [here](#). A goal I made for myself was to make all of the assets (apart from the music, that was taken from Elder Scrolls IV Oblivion) [here is a link to it](#). Overall I got most of what I wanted to done and this will be discussed more in the reflection.

2 Creating the Assets

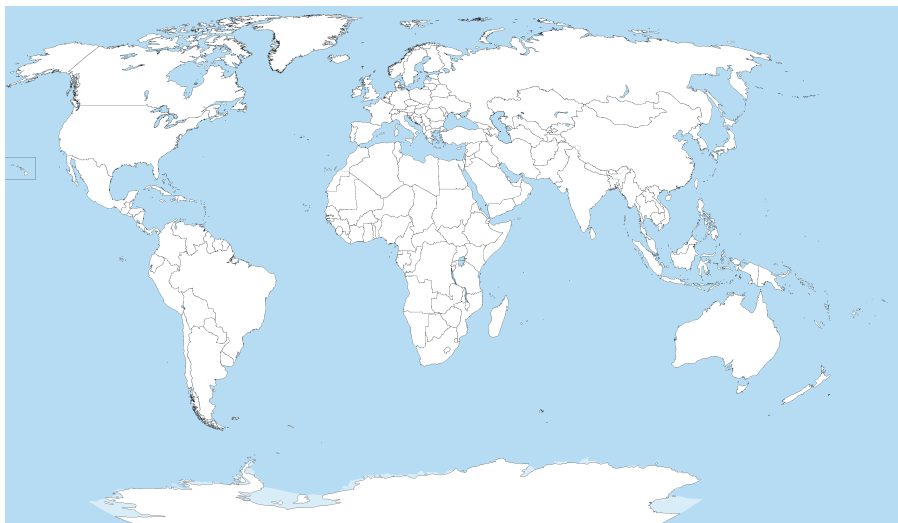
As I am not too experienced with creating assets, this was particularly challenging, I made the conscious decision to make the entire game 2D as I figured it would be easier to make 2D assets than 3D ones and it would contribute greatly to the aesthetic of the game.

2.1 Making The Map



In Order to make a map that was as close to the risk board map as possible I needed something to work as a template, so I obtained a map from wikipedia ([Source](#)) that looked like this:

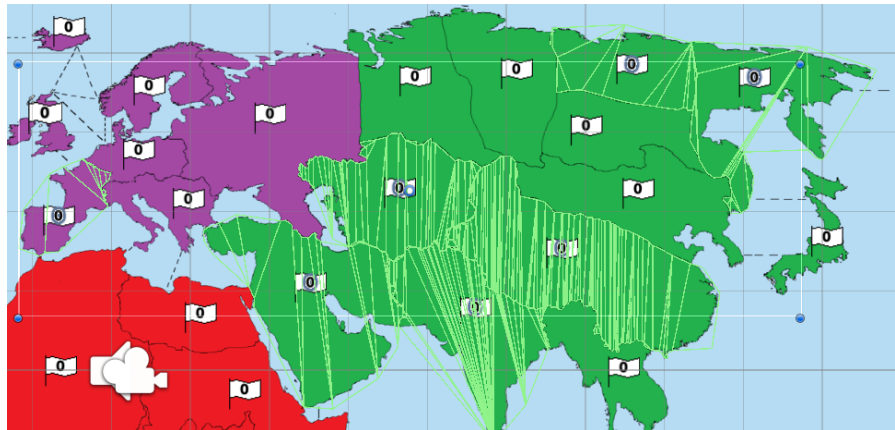
I had to perform numerous tasks to alter the map appropriately to have everything with the correct proportions



that included:

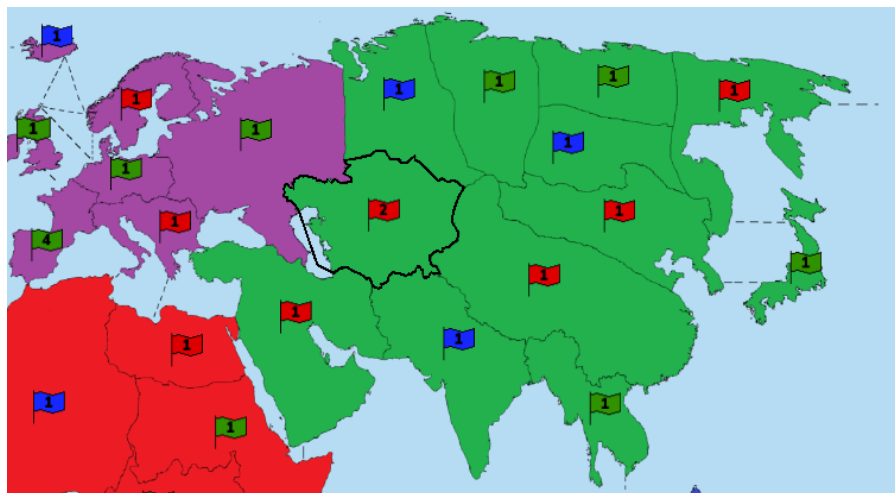
- Resizing Europe, Asia, & Africa
- Erasing Numerous Islands
- resizing greenland
- redrawing most borders to make the appropriate 42 regions,
- recoloring the territories to make the continents prominent

Once I had completed this, I needed to then make it so it was possible to click the region and for the game to know what it was. I achieved this by creating Polygon Colliders for every single Territory. This required me to manually make them follow the borders as closely as possible which was very time consuming. I utilized the special quality of islands and I was a bit more relaxed with how they are laid out and they are just irregular polygons (Unless they have a Land border where that is following the border as closely as possible) Here are some of them highlighted below: Then In order to get the Region The Mouse was over, I used RayCastHit2D to

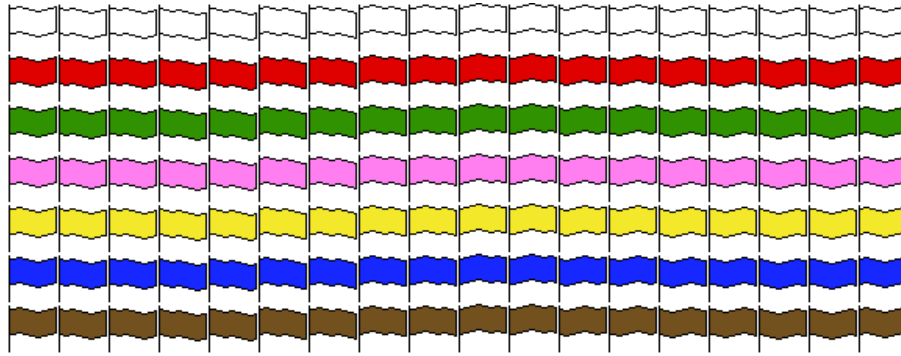


see if I hit anything then performed some checks to see if the collision was valid.

I later realized that it was somewhat difficult to figure out what Territory I had clicked on So I made a Method for the Province Class that each province held that created a line renderer that followed the borders set out by the already existing polygon colliders as a nice visual reference for the player. Here is a demonstration where I click on afghanistan: This screenshot leads to the next section



2.2 Making the flags

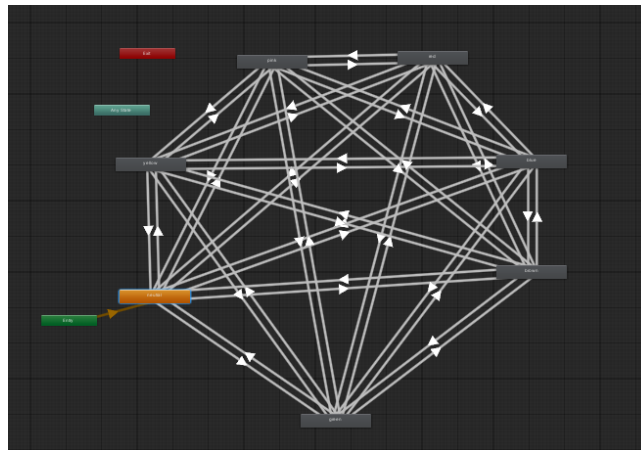


In order to make the flags I used GIMP to make a 32x32 flag, I wanted to make it animated so I created a Flag waving animation to the best of my ability. So Then I duplicated it seven times and added unique colors. I then made seven Unity animations using this spritesheet I created. I was now faced with a new Issue which leads me to the next point.

2.3 Changing the flags color

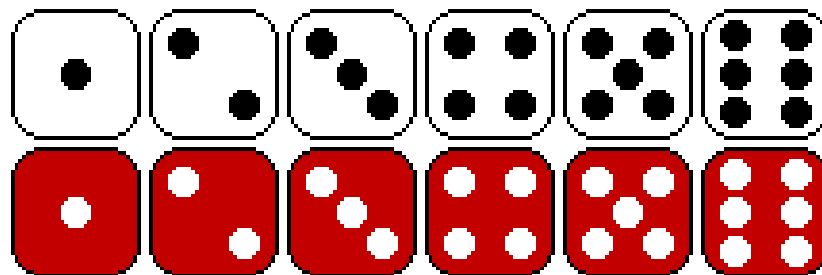
Abandon all hope, ye who enter here

In order to make it transition between the flags when changing teams I had to make an animation tree, sadly



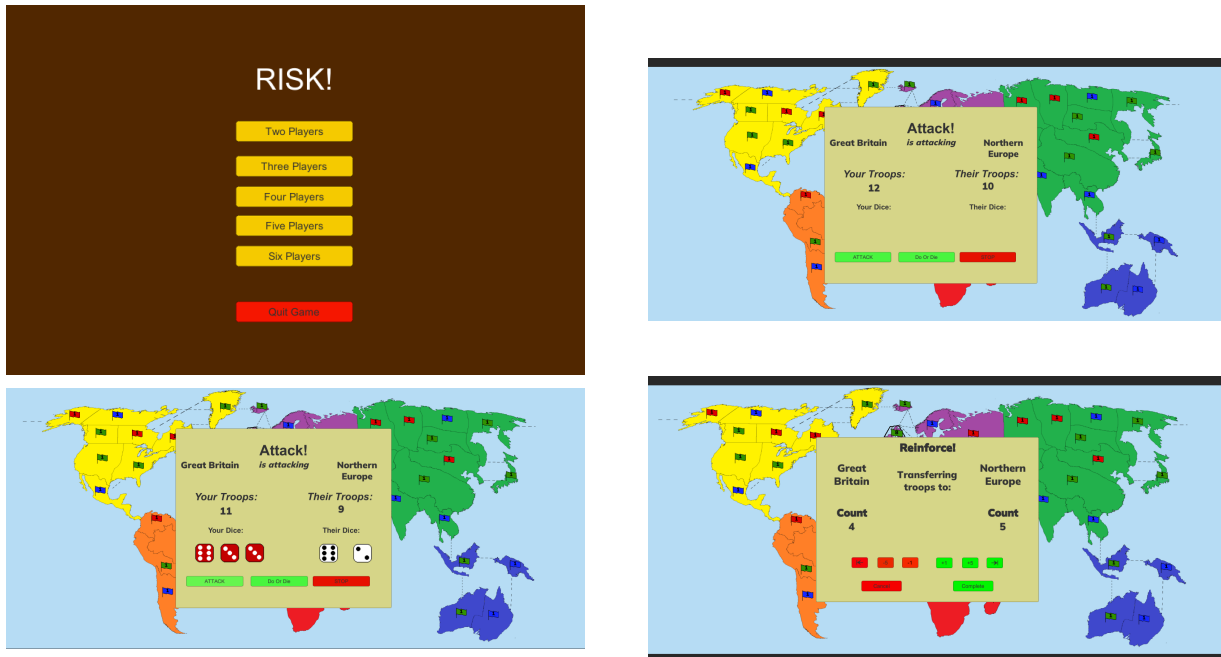
AnyState would not allow the transitions to match the behavior I wanted(They kept transitioning mid animation) I hard coded all 36 conversions. The way I did this was I made an AnimationController with one variable: Team, this was kept as an integer and each value corresponds to a different flag and then if the value changes, it goes to the appropriate flag and so on.

2.4 Making the Dice



I made a 32x32 dice in GIMP and then made all six faces and then an alternative color for the attack dice

3 Making the menus



The Menus were very straight forward for the most part, as they were primarily consisted of buttons, apart from the dice. The Dice were made by making panels with dice faces as images. Each Dice was attached to a Dice Class that was made and when you invoked `Dice.Roll()` it automatically updated the faces of the dice.

4 The Game Itself

Now once all the UI has been shown. I will now go over the turn logic

4.1 Setup

As Risk has two ways of setting up (Either the players pick territories one by one, or they are distributed fairly at random), I Opted for the distribution at random option as I figured that it would be easier to implement. I do however, still have to allow them to populate these regions one by one at the end of the setup phase.

4.2 Your Turn

4.2.1 Replenish

In order to replenish a calculation must be performed to get the number of troops you can distribute. This is done in the same way as the physical variant of risk. That being that the amount of territories you possess are divide by three and rounded down and this value or three whichever is greater, is then added to continental bonuses (These are done by having each continent in a list and I check to see if that list is a subset of all the territories owned by a given player, if it is then the award is granted). You then have to distribute the troops you were awarded here before the beginning of the next phase.

4.3 Attack

The attack form stage is fairly straightforward, you click on a territory if it belongs to you it becomes marked. Then if you selected a territory that is not your own afterwards, an adjacency check is performed and if it is not met then you cannot attack, otherwise the attack interface shows up. If you happen to click on a territory of your own after already selecting a territory of your own, the previously selected territory is replaced with this. If you win the battle a reinforcement interface shows up which allows you to move troops to your newly conquered territory without ending your turn.

4.4 Reinforce

this is the final phase of the Turn. This allows you to move units from one territory to another so long as some connected path exists, and you own both territories. A path is calculated using a simple Depth First Search that I made and then if a path exists then the reinforcement interface shows up, and when this reinforcement ends, your turn concludes. A check is performed to see if anyone owns all 42 territories and if they do you are booted to the menu and the game concludes.

4.5 What is not here/issues

I struggled to calculate an appropriate timeframe for the game and ultimately the UI Design took far longer than I anticipated, so I ended up making this a local hotseat game as I did not have enough time to implement AI. All else that is missing is a territory card system.

Additionally, the game appears somewhat squashed on some monitors as I forgot to account for a 16:9 Aspect Ratio and I ended up forcing a 23:9 ratio to ensure that the entire map was visible.

5 Reflection

Overall this was a very educational experience where I learned a lot about games and project management. I am tempted to keep working on the game and try implement an AI as I reckon it would be pretty fun to do.

6 Code

6.1 Game Manager Script

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5
6
7  /*
8   *   GameManagerScript.cs
9   *   Author: Daniel Hannon (19484286)
10  *   Version: 1
11  */
12  public class GameManagerScript : MonoBehaviour {
13      // Game Control Variables
14      public Camera mainCamera;
15      public bool InDebugMode;
16      private bool attackMenuVisible = false;
17      private bool reinforceMenuVisible = false;
18      private bool mainMenuVisible = true;
19      public Text statusBar;
20      public Button EndTurnButton;
21
22      //Selectors
23      public GameObject selected;
24      public GameObject selected2;
25
26      //Menus
27      public GameObject atkmenu;
28      public GameObject renMenu;
29      public GameObject statusMenu;
30      public GameObject mainMenu;
31
32      //Main Menu Buttons
33      public Text MenuText;
34      public Button twoPlayerButton;
```

```

35 public Button threePlayerButton;
36 public Button fourPlayerButton;
37 public Button fivePlayerButton;
38 public Button sixPlayerButton;
39 public Button quitGame;
40
41 //Attack Menu Specific Variables
42 public Text attacker;
43 public Text defender;
44 public Text attackerTextCount;
45 public Text defenderTextCount;
46 public GameObject[] AttackerDie;
47 public GameObject[] DefenderDie;
48 public Button AttackButton;
49 public Button DoOrDieButton;
50 public Button CancelButton;
51
52 //Reinforce Menu Specific Variables
53 public Text ReinforceFrom;
54 public Text ReinforceTo;
55 public Text ReinforceFromCount;
56 public Text ReinforceToCount;
57 private int FromTerritoryTemp = 0;
58 private int ToTerritoryTemp = 0;
59 public Button reinforceCancel;
60 public Button reinforceConfirm;
61 public Button reinforceAdd1;
62 public Button reinforceDec1;
63 public Button reinforceAdd5;
64 public Button reinforceDec5;
65 public Button reinforceAddAB1;
66 public Button reinforceDecAB1;
67
68 //Turn Specific stuff
69 enum Stages { NOT_STARTED, GAME_SETUP, REPLENISH, ATTACK, REINFORCE };
70 Stages currStage = Stages.NOT_STARTED;
71 int currentPlayer = 0;
72 int numberOfActivePlayers = 2;
73 int numberOfTroopsToPlace = 0;
74 public List<string> playerColors;
75 List<Province>[] TeamTerritories = new List<Province>[6];
76 public bool reinforcementPerformedThisTurn = false;
77
78
79 // Essential Game Variables
80 // I initialize everything at runtime so I don't need to constantly allocate/deallocate memory
81 public List<Province> ClosedList = new List<Province>();
82 public List<Province> OpenList = new List<Province>();
83 public List<string> TeamColors = new List<string>() { "neutral", "red", "blue", "green",
84     ↪ "yellow", "pink", "brown"};
85 public List<Province> AllProvinces = new List<Province>();
86 public List<Province> Europe = new List<Province>();
87 public List<Province> Africa = new List<Province>();
88 public List<Province> Asia = new List<Province>();
89 public List<Province> Oceania = new List<Province>();
90 public List<Province> SouthAmerica = new List<Province>();
91 public List<Province> NorthAmerica = new List<Province>();
92
93 void Start() {
94     mainCamera.aspect = 23f / 9f;

```

```

94 //Attack Menu Setup
95 atkmenu.SetActive(false);
96 AttackButton.GetComponent<Button>().onClick.AddListener(PerformAttack);
97 CancelButton.GetComponent<Button>().onClick.AddListener(CancelButtonListener);
98 DoOrDieButton.GetComponent<Button>().onClick.AddListener(DoOrDieListener);
99
100 //Reinforce Menu Setup
101 renMenu.SetActive(false);
102 reinforceCancel.GetComponent<Button>().onClick.AddListener(CancelReinforce);
103 reinforceAdd1.GetComponent<Button>().onClick.AddListener(ReinforceIncrement1);
104 reinforceAdd5.GetComponent<Button>().onClick.AddListener(ReinforceIncrement5);
105 reinforceAddAB1.GetComponent<Button>().onClick.AddListener(ReinforceIncrementAB1);
106 reinforceDec1.GetComponent<Button>().onClick.AddListener(ReinforceDecrement1);
107 reinforceDec5.GetComponent<Button>().onClick.AddListener(ReinforceDecrement5);
108 reinforceDecAB1.GetComponent<Button>().onClick.AddListener(ReinforceDecrementAB1);
109 reinforceConfirm.GetComponent<Button>().onClick.AddListener(ReinforceConfirm);
110
111 EndTurnButton.GetComponent<Button>().onClick.AddListener(NewTurn);
112 statusMenu.SetActive(false);
113
114 twoPlayerButton.GetComponent<Button>().onClick.AddListener(TwoPlayer);
115 threePlayerButton.GetComponent<Button>().onClick.AddListener(ThreePlayers);
116 fourPlayerButton.GetComponent<Button>().onClick.AddListener(FourPlayers);
117 fivePlayerButton.GetComponent<Button>().onClick.AddListener(FivePlayers);
118 sixPlayerButton.GetComponent<Button>().onClick.AddListener(SixPlayers);
119 quitGame.GetComponent<Button>().onClick.AddListener(QuitGame);
120
121 InDebugMode = false;
122 }
123
124 // Attack Mechanism Listeners
125
126 void AttackSetup() {
127     if (CheckAdjacency(selected.GetComponent<Province>(),
128         ↳ selected2.GetComponent<Province>()) &&
129         ↳ !selected.GetComponent<Province>().Color.Equals(selected2.GetComponent<Province>().Color)
130         ↳ {
131         attackMenuVisible = !attackMenuVisible;
132         atkmenu.SetActive(attackMenuVisible);
133         attacker.text = selected.GetComponent<Province>().ProvinceName;
134         defender.text = selected2.GetComponent<Province>().ProvinceName;
135         attackerTextCount.text = selected.GetComponent<Province>().TroopCount.ToString();
136         defenderTextCount.text = selected2.GetComponent<Province>().TroopCount.ToString();
137     }
138 }
139
140 void CancelButtonListener() {
141     for(int i = 0; i < 3; i++) {
142         AttackerDie[i].GetComponent<DiceRoller>().SetActive(false);
143     }
144
145     for(int j = 0; j < 2; j++) {
146         DefenderDie[j].GetComponent<DiceRoller>().SetActive(false);
147     }
148
149     attackMenuVisible = false;
150     atkmenu.SetActive(false);
151 }
152
153 void DoOrDieListener() {

```

```

151 //Calls Perform Attack until either you conquer the territory, or run out of troops, whatever come
152 while(attackMenuVisible) {
153     PerformAttack();
154 }
155 }
156
157 void PerformAttack() {
158     //Basic Attack Code lol
159     int AttackerDieCount = 0;
160     int DefenderDieCount = 0;
161     List<int> AttackerDiceRolls = new List<int>();
162     List<int> DefenderDiceRolls = new List<int>();
163     //Get Number of Attacker Die
164     switch(selected.GetComponent<Province>().TroopCount) {
165         case 1:
166             CancelButtonListener();
167             break;
168         case 2:
169             //One Die
170             AttackerDieCount = 1;
171             break;
172         case 3:
173             //Two Die
174             AttackerDieCount = 2;
175             break;
176         default:
177             //Three Die
178             AttackerDieCount = 3;
179             break;
180     }
181
182     //Get Number of Defender Die
183     switch(selected2.GetComponent<Province>().TroopCount) {
184         case 0:
185             string temp = selected2.GetComponent<Province>().Color;
186             selected2.GetComponent<Province>().Color =
187                 ↳ selected.GetComponent<Province>().Color;
188             selected2.GetComponent<Province>().TroopCount = 1;
189             selected.GetComponent<Province>().TroopCount--;
190             selected2.GetComponent<Province>().Flag.GetComponent<Animator>().SetInteger("team",
191                 ↳ TeamColors.IndexOf(selected2.GetComponent<Province>().Color));
192             TeamTerritories[playerColors.IndexOf(selected.GetComponent<Province>().Color)].Add(s
193             TeamTerritories[playerColors.IndexOf(temp)].Remove(selected2.GetComponent<Province>()
194             CancelButtonListener();
195             if(selected.GetComponent<Province>().TroopCount > 1) {
196                 ReinforceSetup();
197             }
198             break;
199         case 1:
200             DefenderDieCount = 1;
201             break;
202         default:
203             DefenderDieCount = 2;
204             break;
205     }
206
207     for(int i = AttackerDieCount; i < 3; i++) {
208         AttackerDie[i].GetComponent<DiceRoller>().SetInactive();
209     }

```



```

209     for (int i = DefenderDieCount; i < 2; i++) {
210         DefenderDie[i].GetComponent<DiceRoller>().SetInactive();
211     }
212
213     for(int i = 0; i < AttackerDieCount; i++) {
214         AttackerDiceRolls.Add(AttackerDie[i].GetComponent<DiceRoller>().Roll());
215     }
216
217     for(int i = 0; i < DefenderDieCount; i++) {
218         DefenderDiceRolls.Add(DefenderDie[i].GetComponent<DiceRoller>().Roll());
219     }
220
221     AttackerDiceRolls.Sort();
222     AttackerDiceRolls.Reverse();
223     DefenderDiceRolls.Sort();
224     DefenderDiceRolls.Reverse();
225
226     while(!(AttackerDiceRolls.Count == 0) && !(DefenderDiceRolls.Count == 0)) {
227         if(DefenderDiceRolls[0] >= AttackerDiceRolls[0]) {
228             selected.GetComponent<Province>().TroopCount--;
229             attackerTextCount.text = selected.GetComponent<Province>().TroopCount.ToString();
230         } else {
231             selected2.GetComponent<Province>().TroopCount--;
232             defenderTextCount.text = selected2.GetComponent<Province>().TroopCount.ToString();
233         }
234         AttackerDiceRolls.RemoveAt(0);
235         DefenderDiceRolls.RemoveAt(0);
236     }
237 }
238
239 // Troop Transfer/ Reinforce Mechanism
240
241 public void ReinforceSetup() {
242     if (CheckForPath(selected.GetComponent<Province>(), selected2.GetComponent<Province>())
243         ↳ &&
244         ↳ selected2.GetComponent<Province>().Color.Equals(selected.GetComponent<Province>().Color)
245         ↳ {
246         reinforcementPerformedThisTurn = true;
247         reinforceMenuVisible = true;
248         renMenu.SetActive(true);
249         ReinforceFrom.text = selected.GetComponent<Province>().ProvinceName;
250         ReinforceTo.text = selected2.GetComponent<Province>().ProvinceName;
251         FromTerritoryTemp = selected.GetComponent<Province>().TroopCount - 1;
252         ToTerritoryTemp = 0;
253         ReinforceFromCount.text = (selected.GetComponent<Province>().TroopCount -
254             ↳ ToTerritoryTemp).ToString();
255         ReinforceToCount.text = selected2.GetComponent<Province>().TroopCount.ToString();
256     }
257 }
258
259 public void ReinforceIncrement1() {
260     if (FromTerritoryTemp >= 1) {
261         FromTerritoryTemp -= 1;
262         ToTerritoryTemp++;
263         ReinforceFromCount.text = (selected.GetComponent<Province>().TroopCount -
264             ↳ ToTerritoryTemp).ToString();
265         ReinforceToCount.text = (selected2.GetComponent<Province>().TroopCount +
266             ↳ ToTerritoryTemp).ToString();
267     }
268 }

```

```

263
264 public void ReinforceDecrement1() {
265     if (ToTerritoryTemp >= 1) {
266         ToTerritoryTemp -= 1;
267         FromTerritoryTemp++;
268         ReinforceFromCount.text = (selected.GetComponent<Province>().TroopCount -
269             ↳ ToTerritoryTemp).ToString();
270         ReinforceToCount.text = (selected2.GetComponent<Province>().TroopCount +
271             ↳ ToTerritoryTemp).ToString();
272     }
273 }
274
275 public void ReinforceIncrement5() {
276     if (FromTerritoryTemp >= 1) {
277         if (FromTerritoryTemp >= 5) {
278             FromTerritoryTemp -= 5;
279             ToTerritoryTemp += 5;
280         } else {
281             ToTerritoryTemp += FromTerritoryTemp;
282             FromTerritoryTemp = 0;
283         }
284         ReinforceFromCount.text = (selected.GetComponent<Province>().TroopCount -
285             ↳ ToTerritoryTemp).ToString();
286         ReinforceToCount.text = (selected2.GetComponent<Province>().TroopCount +
287             ↳ ToTerritoryTemp).ToString();
288     }
289 }
290
291 public void ReinforceDecrement5() {
292     if (ToTerritoryTemp >= 1) {
293         if (ToTerritoryTemp >= 5) {
294             ToTerritoryTemp -= 5;
295             FromTerritoryTemp += 5;
296         }
297         else {
298             FromTerritoryTemp += ToTerritoryTemp;
299             ToTerritoryTemp = 0;
300         }
301         ReinforceFromCount.text = (selected.GetComponent<Province>().TroopCount -
302             ↳ ToTerritoryTemp).ToString();
303         ReinforceToCount.text = (selected2.GetComponent<Province>().TroopCount +
304             ↳ ToTerritoryTemp).ToString();
305     }
306 }
307
308 public void ReinforceIncrementAB1() {
309     ToTerritoryTemp += FromTerritoryTemp;
310     FromTerritoryTemp = 0;
311     ReinforceFromCount.text = (selected.GetComponent<Province>().TroopCount -
312         ↳ ToTerritoryTemp).ToString();
313     ReinforceToCount.text = (selected2.GetComponent<Province>().TroopCount +
314         ↳ ToTerritoryTemp).ToString();
315 }
316
317 public void ReinforceDecrementAB1() {
318     FromTerritoryTemp += ToTerritoryTemp;
319     ToTerritoryTemp = 0;
320     ReinforceFromCount.text = (selected.GetComponent<Province>().TroopCount -
321         ↳ ToTerritoryTemp).ToString();

```

```

315 ReinforceToCount.text = (selected2.GetComponent<Province>().TroopCount +
    ↪ ToTerritoryTemp).ToString();
316 }
317
318 public void CancelReinforce() {
319     //Close without making change
320     reinforceMenuVisible = false;
321     renMenu.SetActive(false);
322 }
323
324 public void ReinforceConfirm() {
325     selected.GetComponent<Province>().TroopCount -= ToTerritoryTemp;
326     selected2.GetComponent<Province>().TroopCount += ToTerritoryTemp;
327
328     FromTerritoryTemp = 0;
329     ToTerritoryTemp = 0;
330     reinforceMenuVisible = false;
331     renMenu.SetActive(false);
332 }
333
334 //Main Menu
335 public void TwoPlayer() {
336     numberOfActivePlayers = 2;
337     mainMenu.SetActive(false);
338     mainMenuVisible = false;
339     statusMenu.SetActive(true);
340     GameSetup();
341 }
342
343 public void ThreePlayers() {
344     numberOfActivePlayers = 3;
345     mainMenu.SetActive(false);
346     mainMenuVisible = false;
347     statusMenu.SetActive(true);
348     GameSetup();
349 }
350
351 public void FourPlayers() {
352     numberOfActivePlayers = 4;
353     mainMenu.SetActive(false);
354     mainMenuVisible = false;
355     statusMenu.SetActive(true);
356     GameSetup();
357 }
358
359 public void FivePlayers() {
360     numberOfActivePlayers = 5;
361     mainMenu.SetActive(false);
362     mainMenuVisible = false;
363     statusMenu.SetActive(true);
364     GameSetup();
365 }
366
367 public void SixPlayers() {
368     numberOfActivePlayers = 6;
369     mainMenu.SetActive(false);
370     mainMenuVisible = false;
371     statusMenu.SetActive(true);
372     GameSetup();
373 }

```

```

374
375 public void QuitGame() {
376     Application.Quit();
377 }
378
379 //General Game Mechanics
380
381 public GameObject GetTerritoryClick() {
382     Vector2 worldPoint = Camera.main.ScreenToWorldPoint(Input.mousePosition);
383     RaycastHit2D hit = Physics2D.Raycast(worldPoint, Vector2.zero);
384     if(hit) {
385         if (hit.collider.gameObject.layer == LayerMask.NameToLayer("Provinces")) {
386             return hit.collider.gameObject;
387         }
388     }
389     return null;
390 }
391
392 public bool CheckAdjacency(Province p1, Province p2) {
393     //this is for attacking
394     for(int i = 0; i < p1.neighbors.Length; i++) {
395         if (p1.neighbors[i].GetComponent<Province>().Equals(p2)) {
396             return true;
397         }
398     }
399     return false;
400 }
401
402 public bool CheckForPath(Province p1, Province p2) {
403     OpenList.Add(p1);
404     return CheckForPath(p2);
405 }
406
407 public bool CheckForPath(Province p2) {
408     //this is for reinforcement
409     Province temp;
410     Province temp2;
411     while(OpenList.Count != 0) {
412         temp = OpenList[0];
413         OpenList.Remove(temp);
414         ClosedList.Add(temp);
415         for (int i = 0; i < temp.neighbors.Length; i++) {
416             temp2 = temp.neighbors[i].GetComponent<Province>();
417             if(temp2.Color.Equals(temp.Color)) {
418                 if (temp2.Equals(p2)) {
419                     OpenList.Clear();
420                     ClosedList.Clear();
421                     return true;
422                 } else if (ClosedList.Contains(temp2) != true) {
423                     OpenList.Add(temp2);
424                 }
425             }
426         }
427     }
428     OpenList.Clear();
429     ClosedList.Clear();
430     return false;
431 }
432
433 public void DebugEval() {

```

```

434     if (Input.GetKeyDown(KeyCode.E)) {
435         GameSetup();
436     }
437
438
439     if (Input.GetMouseButtonDown(0) && !attackMenuVisible && !reinforceMenuVisible) {
440         GameObject temp = GetTerritoryClick();
441         if (temp != null) {
442             if (selected != null) {
443                 if (temp != selected) {
444                     selected.GetComponent<Province>().Deselect();
445                 }
446             }
447             if (temp != selected) {
448                 temp.GetComponent<Province>().Select();
449             }
450             selected = temp;
451             selected.GetComponent<Province>().TroopCount += 1;
452         }
453     }
454
455     if (Input.GetMouseButtonDown(1) && !attackMenuVisible) {
456         selected2 = GetTerritoryClick();
457     }
458
459     if (selected != null) {
460         /*
461         * Neutral  0
462         * Red      1
463         * Blue     2
464         * Green    3
465         * Yellow   4
466         * Pink     5
467         * Brown    6
468         */
469         //Flag Color Handling
470         GameObject temp = selected.GetComponent<Province>().Flag;
471         if (temp != null) {
472             if (Input.GetKeyDown(KeyCode.Alpha1)) {
473                 selected.GetComponent<Province>().Color = "red";
474                 temp.GetComponent<Animator>().SetInteger("team", 1);
475             }
476             else if (Input.GetKeyDown(KeyCode.Alpha2)) {
477                 selected.GetComponent<Province>().Color = "blue";
478                 temp.GetComponent<Animator>().SetInteger("team", 2);
479             }
480             else if (Input.GetKeyDown(KeyCode.Alpha3)) {
481                 selected.GetComponent<Province>().Color = "green";
482                 temp.GetComponent<Animator>().SetInteger("team", 3);
483             }
484             else if (Input.GetKeyDown(KeyCode.Alpha4)) {
485                 selected.GetComponent<Province>().Color = "yellow";
486                 temp.GetComponent<Animator>().SetInteger("team", 4);
487             }
488             else if (Input.GetKeyDown(KeyCode.Alpha5)) {
489                 selected.GetComponent<Province>().Color = "pink";
490                 temp.GetComponent<Animator>().SetInteger("team", 5);
491             }
492             else if (Input.GetKeyDown(KeyCode.Alpha6)) {
493                 selected.GetComponent<Province>().Color = "brown";

```

```

494     temp.GetComponent<Animator>().SetInteger("team", 6);
495 }
496 else if (Input.GetKeyDown(KeyCode.Alpha0)) {
497     selected.GetComponent<Province>().Color = "neutral";
498     temp.GetComponent<Animator>().SetInteger("team", 0);
499 }
500 }
501
502 if (selected2 != null) {
503     //Adjacency Checker
504     if (Input.GetKeyDown(KeyCode.A)) {
505         if (CheckAdjacency(selected.GetComponent<Province>(),
506             ↪ selected2.GetComponent<Province>())) {
507             Debug.Log(selected.GetComponent<Province>().ProvinceName + " and " +
508             ↪ selected2.GetComponent<Province>().ProvinceName + " Are Adjacent!");
509         }
510         else {
511             Debug.Log(selected.GetComponent<Province>().ProvinceName + " and " +
512             ↪ selected2.GetComponent<Province>().ProvinceName + " Are Not Adjacent!");
513         }
514     }
515 }
516
517 if (Input.GetKeyDown(KeyCode.D)) {
518     //Attack test
519     AttackSetup();
520 }
521
522 //Check for Path (for troop reinforcements)
523 if (Input.GetKeyDown(KeyCode.S)) {
524     OpenList.Add(selected.GetComponent<Province>());
525     if (CheckForPath(selected2.GetComponent<Province>())) {
526         Debug.Log("A Path From " + selected.GetComponent<Province>() + " to " +
527         ↪ selected2.GetComponent<Province>() + " Exists!");
528     }
529     else {
530         Debug.Log("A Path Does Not Exist!");
531     }
532 }
533
534 if (Input.GetKeyDown(KeyCode.R)) {
535     ReinforceSetup();
536 }
537 }
538 }
539
540 public void GameSetup() {
541     List<Province> provincesCopy = new List<Province>(AllProvinces);
542     for(int i = 0; i < 6; i++) {
543         TeamTerritories[i] = new List<Province>();
544     }
545     int numOfTeams = numberOfActivePlayers;
546     if(numOfTeams == 2) {
547         numOfTeams = 3;
548     }
549     while(provincesCopy.Count != 0) {
550         int temp = Random.Range(0, provincesCopy.Count);
551         provincesCopy[temp].Color = playerColors[currentPlayer];
552         provincesCopy[temp].Flag.GetComponent<Animator>().SetInteger("team",
553         ↪ TeamColors.IndexOf(playerColors[currentPlayer]));

```

```

549     TeamTerritories[currentPlayer].Add(provincesCopy[temp]);
550     provincesCopy[temp].TroopCount = 1;
551     provincesCopy.RemoveAt(temp);
552     if(currentPlayer == (numOfTeams - 1)) {
553         currentPlayer = 0;
554     } else {
555         currentPlayer++;
556     }
557 }
558
559 switch(numberOfActivePlayers) {
560     case 2:
561         numberOfTroopsToPlace = 26;
562         break;
563     case 3:
564         numberOfTroopsToPlace = 21;
565         break;
566     case 4:
567         numberOfTroopsToPlace = 19;
568         break;
569     case 5:
570         numberOfTroopsToPlace = 16;
571         break;
572     case 6:
573         numberOfTroopsToPlace = 13;
574         break;
575 }
576 currStage = Stages.GAME_SETUP;
577 currentPlayer = 0;
578 }
579
580 public void NewTurn() {
581     if(selected) {
582         selected.GetComponent<Province>().Deselect();
583     }
584     if(TeamTerritories[currentPlayer].Count == 42) {
585         MenuText.text = playerColors[currentPlayer].ToUpper() + " Won the Game!";
586         mainMenuVisible = true;
587         mainMenu.SetActive(true);
588         statusMenu.SetActive(false);
589     }
590     else if (!attackMenuVisible && !reinforceMenuVisible && (currStage > Stages.REPLENISH))
591     {
592         if (currentPlayer == (numberOfActivePlayers - 1)) {
593             currentPlayer = 0;
594         }
595         else {
596             currentPlayer++;
597         }
598         while(TeamTerritories[currentPlayer].Count == 0) {
599             if(currentPlayer == (numberOfActivePlayers - 1)) {
600                 currentPlayer = 0;
601             } else {
602                 currentPlayer++;
603             }
604         }
605         numberOfTroopsToPlace = calculateReinforcements();
606         currStage = Stages.REPLENISH;
607     }

```

```

608 }
609
610 public bool FindInList(List<Province> subList, List<Province> superList) {
611     foreach (Province province in subList) {
612         if(!superList.Contains(province)) {
613             return false;
614         }
615     }
616     return true;
617 }
618
619 int calculateReinforcements() {
620     int territoryOwnershipCount = Mathf.Max(TeamTerritories[currentPlayer].Count / 3,3);
621     int continentBonus = 0;
622     //Check Continental Bonuses
623     if(FindInList(Europe, TeamTerritories[currentPlayer])) {
624         continentBonus += 5;
625     }
626     if(FindInList(Asia, TeamTerritories[currentPlayer])) {
627         continentBonus += 7;
628     }
629     if(FindInList(Africa, TeamTerritories[currentPlayer])) {
630         continentBonus += 3;
631     }
632     if(FindInList(Oceania, TeamTerritories[currentPlayer])) {
633         continentBonus += 2;
634     }
635     if(FindInList(SouthAmerica,TeamTerritories[currentPlayer])) {
636         continentBonus += 2;
637     }
638     if(FindInList(NorthAmerica,TeamTerritories[currentPlayer])) {
639         continentBonus += 5;
640     }
641     return territoryOwnershipCount + continentBonus;
642 }
643
644
645 // Update is called once per frame
646 void Update() {
647     /* Basically I want to keep debug mode (in case of emergency) so it's activated/deactivated using
648     if(Input.GetKeyDown(KeyCode.BackQuote)) {
649         //Toggle Debug mode
650         InDebugMode = !InDebugMode;
651         Debug.Log("Debug Mode Toggled!");
652     }
653
654     if (InDebugMode) {
655         DebugEval();
656     } else if (!mainMenuVisible) {
657         switch (currStage) {
658             case Stages.NOT_STARTED:
659                 GameSetup();
660                 break;
661             case Stages.GAME_SETUP:
662                 statusBar.text = playerColors[currentPlayer].ToUpper() + ": Place a Troop!";
663                 if(Input.GetMouseButtonDown(0) && !attackMenuVisible && !reinforceMenuVisible) {
664                     selected = GetTerritoryClick();
665                     if (selected &&
666                         → selected.GetComponent<Province>().Color.Equals(playerColors[currentPlayer]))
667                         → {

```



```

666         selected.GetComponent<Province>().TroopCount++;
667         if(currentPlayer == (numberOfActivePlayers - 1)) {
668             currentPlayer = 0;
669             numberOfTroopsToPlace--;
670             if (numberOfTroopsToPlace == 0) {
671                 currStage = Stages.REPLENISH;
672                 numberOfTroopsToPlace = calculateReinforcements();
673             }
674             } else {
675                 currentPlayer++;
676             }
677         }
678     }
679     break;
680 case Stages.REPLENISH:
681     statusBar.text = playerColors[currentPlayer].ToUpper() + ": Place " +
        ↳ numberOfTroopsToPlace.ToString() + " Troops!";
682     if(Input.GetMouseButtonDown(0) && !attackMenuVisible && !reinforceMenuVisible) {
683         selected = GetTerritoryClick();
684         if (selected &&
        ↳ selected.GetComponent<Province>().Color.Equals(playerColors[currentPlayer]))
        ↳ {
685             selected.GetComponent<Province>().TroopCount++;
686             numberOfTroopsToPlace--;
687             if(numberOfTroopsToPlace == 0) {
688                 currStage = Stages.ATTACK;
689                 selected = null;
690             }
691         }
692     }
693     break;
694 case Stages.ATTACK:
695     if(!selected) {
696         statusBar.text = playerColors[currentPlayer].ToUpper() + ": Click a Territory
        ↳ to attack from";
697     } else {
698         statusBar.text = playerColors[currentPlayer].ToUpper() + ": Select a region to
        ↳ attack\nRight Click To Reinforce";
699     }
700     if(Input.GetMouseButtonDown(0) && !attackMenuVisible && !reinforceMenuVisible) {
701         GameObject temp = GetTerritoryClick();
702         if(temp) {
703             if (temp.GetComponent<Province>().Color.Equals(playerColors[currentPlayer]))
        ↳ {
704                 if(selected) {
705                     selected.GetComponent<Province>().Deselect();
706                 }
707                 selected = temp;
708                 selected.GetComponent<Province>().Select();
709             } else {
710                 selected2 = temp;
711                 AttackSetup();
712             }
713         }
714     }
715     if(Input.GetMouseButtonDown(1)) {
716         currStage = Stages.REINFORCE;
717         reinforcementPerformedThisTurn = false;
718         if (selected) {
719             selected.GetComponent<Province>().Deselect();

```

```

720     }
721     selected = null;
722     selected2 = null;
723 }
724 break;
725 case Stages.REINFORCE:
726     if (!selected) {
727         statusBar.text = playerColors[currentPlayer].ToUpper() + ": Select a Region to
728         ↳ Move Troops From";
729     } else {
730         statusBar.text = playerColors[currentPlayer].ToUpper() + ": Select a Region to
731         ↳ Move Troops To\n(Right Click To change departing region)";
732     }
733     if (Input.GetMouseButtonDown(0) && !attackMenuVisible && !reinforceMenuVisible) {
734         GameObject temp = GetTerritoryClick();
735         if (temp) {
736             if (temp.GetComponent<Province>().Color.Equals(playerColors[currentPlayer]))
737                 ↳ {
738                 if (!selected) {
739                     temp.GetComponent<Province>().Select();
740                     selected = temp;
741                 } else {
742                     selected2 = temp;
743                     ReinforceSetup();
744                 }
745             }
746         }
747     }
748     if (Input.GetMouseButtonDown(1) && !attackMenuVisible && !reinforceMenuVisible) {
749         GameObject temp = GetTerritoryClick();
750         if (temp) {
751             if (temp.GetComponent<Province>().Color.Equals(playerColors[currentPlayer]))
752                 ↳ {
753                 if (selected) {
754                     selected.GetComponent<Province>().Deselect();
755                 }
756                 selected = temp;
757             }
758         }
759     }
760     if (!reinforceMenuVisible && reinforcementPerformedThisTurn) {
761         NewTurn();
762     }
763     break;
764 }
765 if (Input.GetKeyDown(KeyCode.Escape)) {
766     mainMenu.SetActive(true);
767     mainMenuVisible = true;
768     statusMenu.SetActive(true);
769 }
770 }
771 }
772 }

```

6.2 Dice Roller

```

1 using System.Collections;
2 using System.Collections.Generic;

```

```

3  using UnityEngine;
4  using UnityEngine.UI;
5
6  /**
7   * DiceRoller.cs
8   * Author: Daniel Hannon (19484286)
9   * Version: 1
10  */
11
12  public class DiceRoller : MonoBehaviour {
13      // Start is called before the first frame update
14      public Sprite[] AttackDiceFaces;
15      public Sprite[] DefenseDiceFaces;
16      public int dice_type;
17      public int current_value = 0;
18
19      public int Roll() {
20          current_value = Random.Range(1, 7);
21          if (dice_type == 0) {
22              //Attack Dice
23              gameObject.GetComponent<Image>().sprite = AttackDiceFaces[current_value];
24          }
25          else {
26              //Defense Dice
27              gameObject.GetComponent<Image>().sprite = DefenseDiceFaces[current_value];
28          }
29          return current_value;
30      }
31
32      public void SetInactive() {
33          gameObject.GetComponent<Image>().sprite = AttackDiceFaces[0];
34      }
35  }

```

6.3 Province

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  /**
6   * Province.cs
7   * Author: Daniel Hannon (19484286)
8   * Version: 1
9   */
10
11  public class Province : MonoBehaviour {
12      public string ProvinceName;
13      public string Color = "neutral";
14      public int TroopCount = 0;
15      public GameObject Flag;
16      public GameObject TroopField;
17      public GameObject[] neighbors;
18      private LineRenderer highlight = null;
19
20
21      void start() {
22          TroopCount = 0;

```

```

23     Color = "neutral";
24 }
25
26 void Update() {
27
28     if(TroopField!= null) {
29         TroopField.GetComponent<TextMesh>().text = TroopCount.ToString();
30     }
31 }
32
33 public void Select() {
34     //The select method creates a LineRenderer and follows the path set by the PolygonCollider used to
35     //As a means of letting the user know what territory they currently have selected, I realized this
36     //Some UI prompt that has something like *Currently Selected: Region Name* as region names are not
37     highlight = gameObject.AddComponent<LineRenderer>();
38     if (highlight) {
39         Vector2[] path = gameObject.GetComponent<PolygonCollider2D>().points;
40         Color black = new Color(0, 0, 0, 1);
41         highlight.material = new Material(Shader.Find("Legacy Shaders/Particles/Alpha
42         → Blended Premultiply"));
43         highlight.startColor = black;
44         highlight.endColor = black;
45         highlight.startWidth = 0.03f;
46         for (int i = 0; i < path.Length; i++) {
47             path[i] = gameObject.transform.TransformPoint(path[i]);
48         }
49         highlight.positionCount = path.Length + 1;
50         for (int i = 0; i < path.Length; i++) {
51             Vector3 finalLine = path[i];
52             finalLine.z = 30;
53             highlight.SetPosition(i, finalLine);
54
55             if (i == (path.Length - 1)) {
56                 finalLine = path[0];
57                 finalLine.z = 30;
58                 highlight.SetPosition(path.Length, finalLine);
59             }
60         }
61     }
62
63     public void Deselect() {
64         //This destroys the LineRenderer (I was worried it'd delete the points for the PolygonCollider but
65         if (highlight) {
66             Destroy(gameObject.GetComponent<LineRenderer>());
67             highlight = null;
68         }
69     }
70 }

```