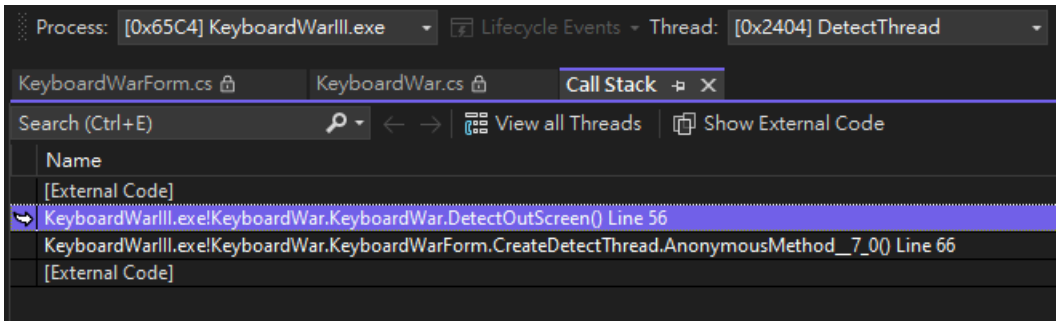
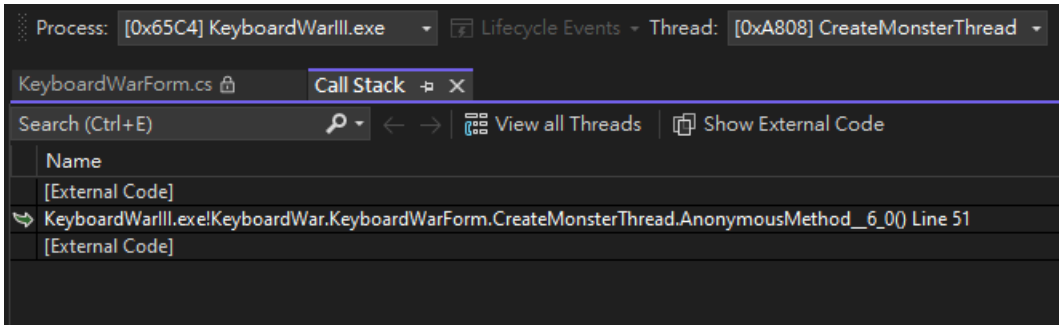
HW3_109403021

1. 各執行緒 Call Stack 截圖

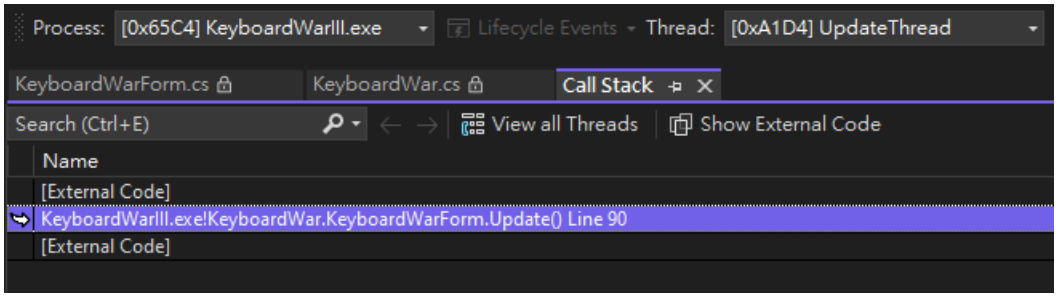
DetectThread



CreateMonsterThread

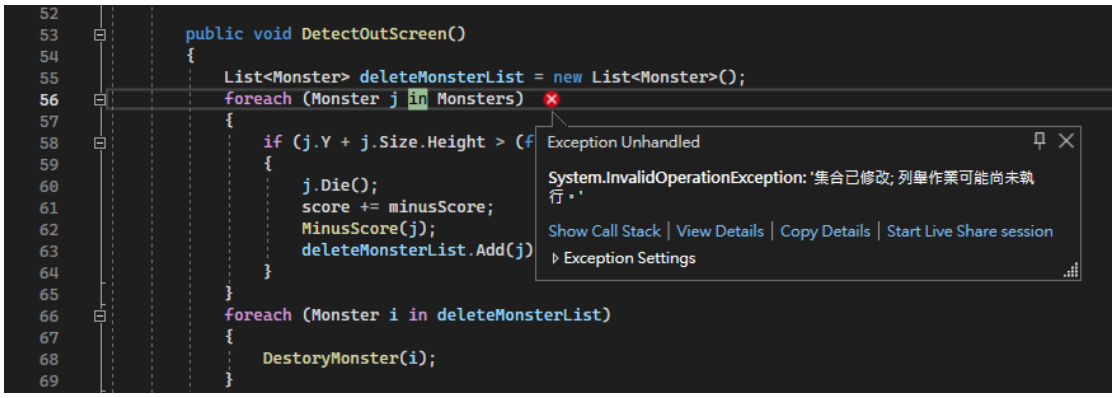


UpdateThread

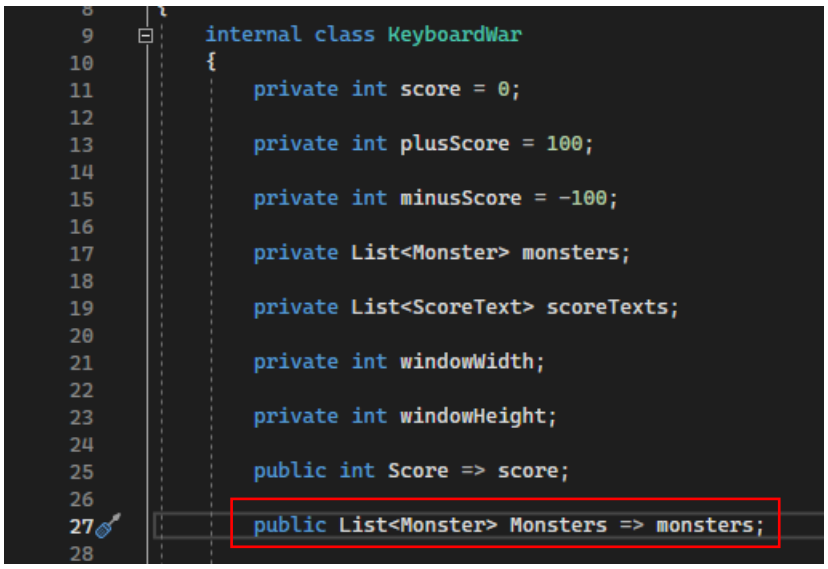


2. 解釋和猜測

首先追蹤 DetectThread 執行緒的 Call Stack，遊戲崩潰是發生在 KeyboardWar.cs 的第 56 行，找到此處程式碼，看到 Exception Unhandled 顯示如下



看來應該是在迴圈遍歷 Monsters 這個集合的過程中，Monsters 被修改了，應該是其他執行緒有動到



Monsters 是 KeyboardWar.cs 中宣告的 List，且為 public 可被其他腳本使用

切換到其他執行緒追蹤其 Call Stack 找到對應的程式碼處，以下程式碼皆位於 KeyboardWarForm.cs

```
58      private void CreateDetectThread()
59      {
60          Thread thread = new Thread((ThreadStart)delegate
61          {
62              while (true)
63              {
64                  keyboardWar.DetectOutScreen();
65                  keyboardWar.DestoryScoreTexts();
66                  Thread.Sleep((int)detectInterval);
67              }
68          });
69          thread.Name = "DetectThread";
70          thread.Start();
71      }
```

```
44      private void CreateMonsterThread()
45      {
46          Thread thread = new Thread((ThreadStart)delegate
47          {
48              while (true)
49              {
50                  keyboardWar.CreateMonster();
51                  Thread.Sleep((int)interval);
52              }
53          });
54          thread.Name = "CreateMonsterThread";
55          thread.Start();
56      }
```

```
73      private new void Update()
74      {
75          while (true)
76          {
77              foreach (Monster monster in keyboardWar.Monsters)
78              {
79                  monster.Update();
80              }
81              foreach (ScoreText scoreText in keyboardWar.ScoreTexts)
82              {
83                  scoreText.Update();
84              }
85              if (interval > 50f)
86              {
87                  interval -= 1f;
88              }
89              pictureBox.Invalidate();
90              Thread.Sleep((int)physicInterval);
91          }
92      }
```

找到了是在 UpdateThread 中上圖紅框處的程式碼，會更動到 KeyKeyboardWar.cs 的公開 List 變數 Monsters。

所以我猜測，正是因為 DetectThread 和 UpdateThread 兩個執行緒都會同時動到 Monsters，之間又沒有同步得當，就會導致 Race Condition 的發生。

而如果要解決這個 bug，就是得處理好同步的問題，應該就是用 lock 之類的方式確保一個時間點只能有一個執行緒訪問 Monsters 變數，但這也可能會讓遊戲無法順暢運行。另一種辦法就是不讓兩邊都訪問 Monsters 這個變數，減少共享，而是重新設計遊戲邏輯將 Monsters 分為兩個不同的變數，各自執行緒不要拜訪到同一變數，但這可能就會對整個架構造成不小的變動十分麻煩。