1. SpawnTest：test board size

```
[TestCase(1)]
[TestCase(2)]
[TestCase(3)]
public void SpawnTest(int mode)
{

//test object
AltObject boardManager = altDriver.FindObject(By.NAME, "BoardManager");
Assert.NotNull(boardManager, "BoardManager should be present in the scene");

//test script
const string componentName = "BoardManager";
const string methodName = "SpawnBoard";
const string propertyName = "BoardList.Count";
const string assemblyName = "Assembly-CSharp";
object[] parameters = new object[] {mode};
int AnsExpected = 0;
if(mode == 1) AnsExpected = 81;
else if(mode == 2) AnsExpected = 256;
else if(mode == 3) AnsExpected = 480;
//var propertyValue = boardManager.GetComponentProperty<int>(componentName, propertyName, assemblyName);
var data = boardManager.CallComponentMethod<int>(componentName, methodName, assemblyName, parameters);
var propertyValue = boardManager.GetComponentProperty<int>(componentName, propertyName, assemblyName);

Assert.AreEqual(AnsExpected, propertyValue);
}
```

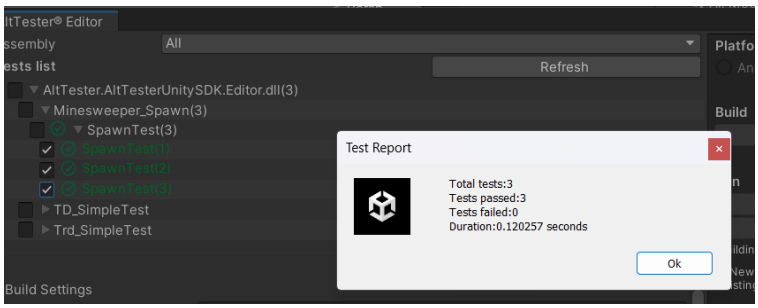
```
[01:14:00] 2025-06-05 01:14:00.1087|Tester|DEBUG|command received: {
  "component": "BoardManager",
[01:14:00] spawn 1
[01:14:00] 2025-06-05 01:14:00.1252|Tester|DEBUG|response sent: {"messageId":"638846540401021294","driverId":"e31c24ecf00642aeb47f5f4d43b4bd35",
[01:14:00] 2025-06-05 01:14:00.1252|Tester|DEBUG|command received: {
  "component": "BoardManager",
[01:14:00] 2025-06-05 01:14:00.1393|Tester|DEBUG|response sent: {"messageId":"638846
[01:14:00] 2025-06-05 01:14:00.1393|Editor|INFO|==============> TEST Minesweeper_S
[01:14:00] 2025-06-05 01:14:00.1393|Editor|ERROR|  Expected: 81
  But was:  0
[01:14:00] 2025-06-05 01:14:00.1393|Editor|INFO|==============================
[01:14:00] 2025-06-05 01:14:00.1580|Tester|DEBUG|command received: {"messageId": null,"driverId": "","commandName": "DriverDisconnectedNotification","isNo
```

Test Report

Total tests:1
Tests passed:0
Tests failed:1
Duration:0.1427942 seconds

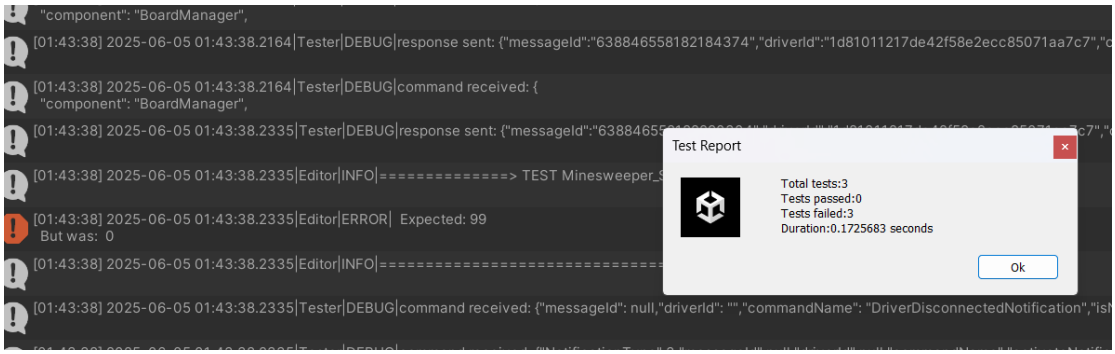
Ok

```
public void SpawnBoard(int mode = 1) //1:9*9, 10, 2:16*16, 40, 3:30*16, 99
{
    Debug.Log($"spawn {mode}");
    switch (mode)
    {
        case 1:
            width = 9;
            height = 9;
            mineCount = 10;
            break;
        case 2:
            width = 16;
            height = 16;
            mineCount = 40;
            break;
        case 3:
            width = 30;
            height = 16;
            mineCount = 99;
            break;
        default:
            width = 9;
            height = 9;
            mineCount = 10;
            break;
    }

    int totalCells = width * height;
    BoardList = new List<int>(new int[totalCells]);
```

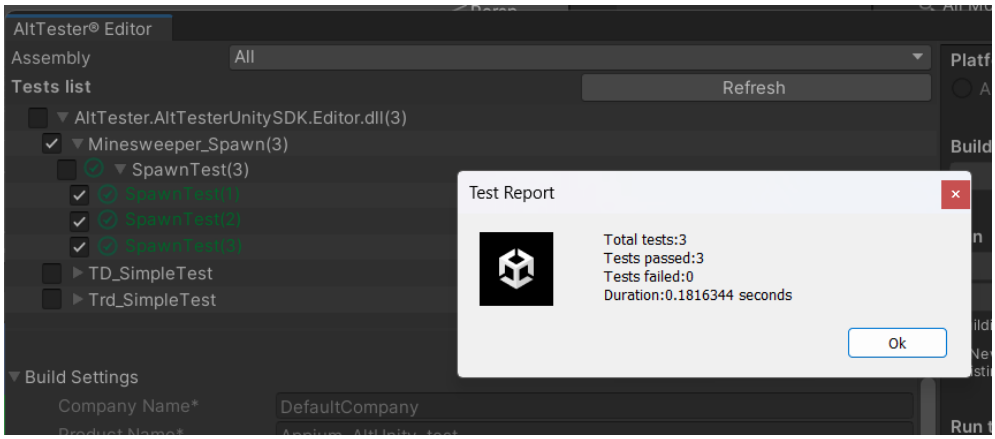


2. SpawnTest：test bomb number

```
//test bomb number
propertyName = "placedMines";
if(mode == 1) AnsExpected = 10;
else if(mode == 2) AnsExpected = 40;
else if(mode == 3) AnsExpected = 99;
propertyValue = boardManager.GetComponentProperty<int>(componentName, propertyName, assemblyName);

Assert.AreEqual(AnsExpected, propertyValue);
```

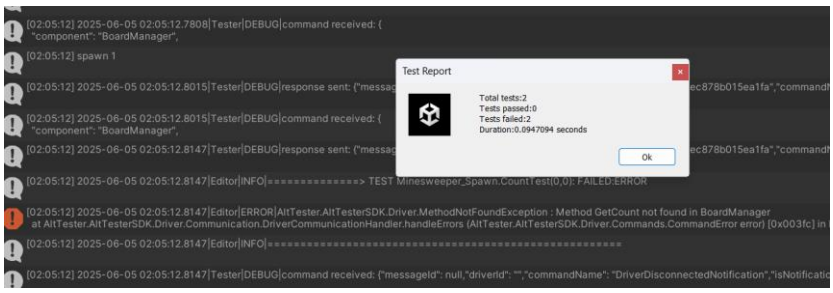



```
// 放置地雷：-1 為地雷
placedMines = 0;
System.Random rand = new System.Random();
while (placedMines < mineCount)
{
    int index = rand.Next(0, totalCells);
    if (BoardList[index] != -1)
    {
        BoardList[index] = -1;
        placedMines++;
    }
}
```



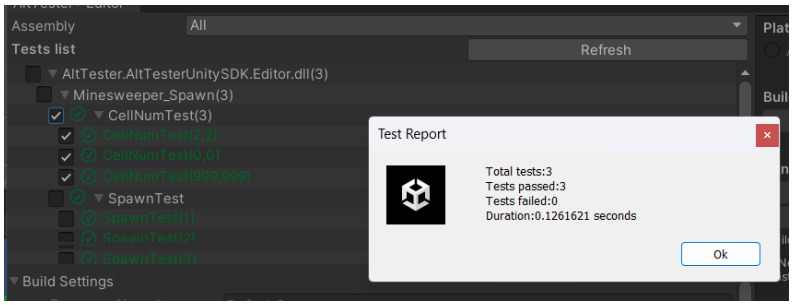
3. SpawnTest：test cell number

```
[TestCase(2, 2)]
[TestCase(0, 0)] //edge case
[TestCase(999, 999)] //edge case
public void CellNumTest(int x, int y)
{
    //test object
    AltObject boardManager = altDriver.FindObject(By.NAME, "BoardManager");

    //spawn board
    string componentName = "BoardManager";
    string methodName = "SpawnBoard";
    string assemblyName = "Assembly-CSharp";
    object[] parameters = new object[] {1};
    var data = boardManager.CallComponentMethod<int>(componentName, methodName, assemblyName, parameters);

    //test cell represent number
    string methodToVerifyName = "GetCount";
    object[] pos = new object[] {(x, y)};
    var cellCount =  boardManager.CallComponentMethod<int>(componentName, methodToVerifyName, assemblyName, pos);
    int A = 0, B = 8;
    Assert.IsTrue(cellCount >= A && cellCount <= B);
}
```



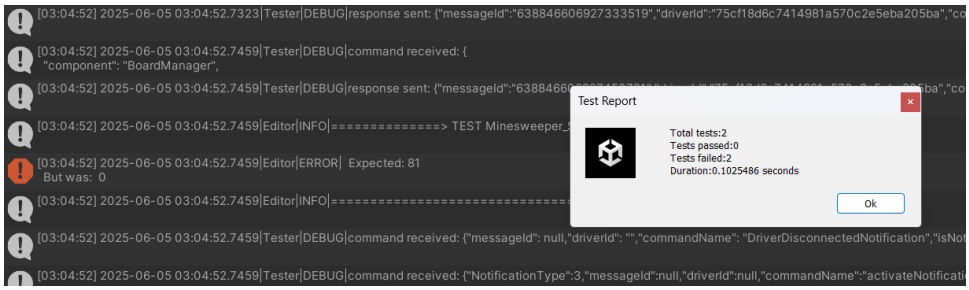

```
// 計算非地雷格周圍地雷數量
for (int y = 0; y < height; y++)
{
    for (int x = 0; x < width; x++)
    {
        int index = GetIndex(x, y);
        if (BoardList[index] == -1)
            continue;

        int count = GetCount(x, y);
        BoardList[index] = count;
    }
}
```

```
int GetIndex(int x, int y)
{
    return y * width + x;
}

bool IsInside(int x, int y)
{
    return x >= 0 && y >= 0 && x < width && y < height;
}

public int GetCount(int x, int y)
{
    int count = 0;
    for (int dx = -1; dx <= 1; dx++)
    {
        for (int dy = -1; dy <= 1; dy++)
        {
            if (dx == 0 && dy == 0)
                continue;
            int nx = x + dx;
            int ny = y + dy;
            if (IsInside(nx, ny) && BoardList[GetIndex(nx, ny)] == -1)
            {
                count++;
            }
        }
    }
    return count;
}
```



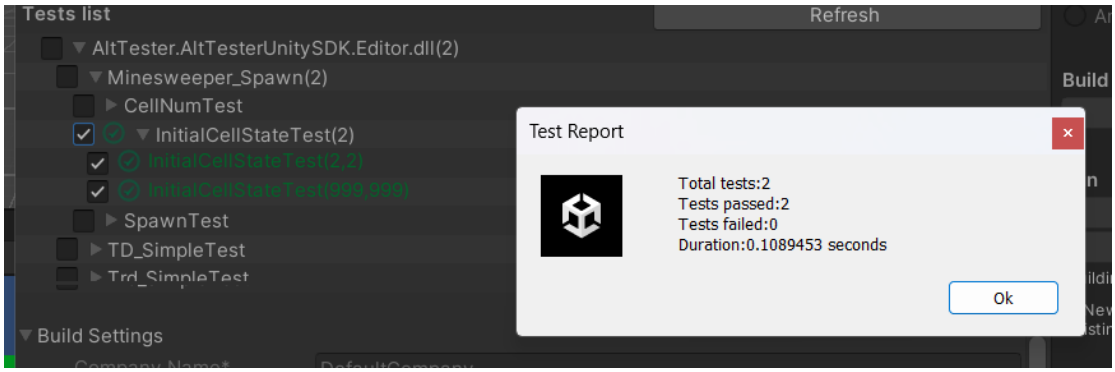
4. SpawnTest：test cell state at init

```
[TestCase(2, 2)]
[TestCase(999, 999)] //edge case
public void InitialCellStateTest(int x, int y)
{
    //test object
    AltObject boardManager = altDriver.FindObject(By.NAME, "BoardManager");

    //spawn board
    string componentName = "BoardManager";
    string methodName = "SpawnBoard";
    string assemblyName = "Assembly-CSharp";
    object[] parameters = new object[] {1};
    var data = boardManager.CallComponentMethod<int>(componentName, methodName, assemblyName, parameters);

    //cell record count
    string propertyName = "CellList.Count";
    var propertyValue = boardManager.GetComponentProperty<int>(componentName, propertyName, assemblyName);
    Assert.AreEqual(81, propertyValue);

    //check target cell state
    string methodToVerifyName = "GetState";
    object[] pos = new object[] {x, y};
    var cellCount =  boardManager.CallComponentMethod<int>(componentName, methodToVerifyName, assemblyName, pos);
    Assert.AreEqual(0, cellCount);
}
```




```
// initial cell with unopen state
CellList = new List<int>(new int[totalCells]);
for(int i = 0;i<totalCells;i++) CellList[i] = 0;
```

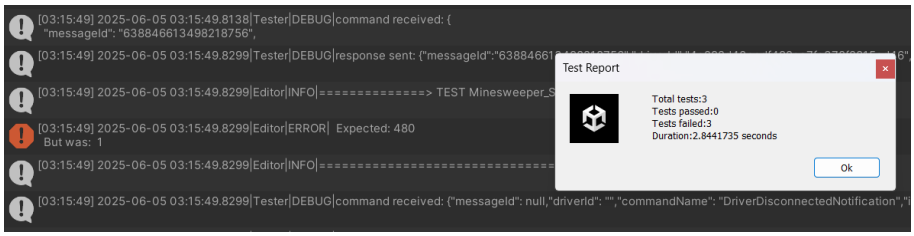
```
public int GetState(int x, int y)
{
    if(x >= width && y >= height) return 0;
    return CellList[GetIndex(x, y)];
}
```



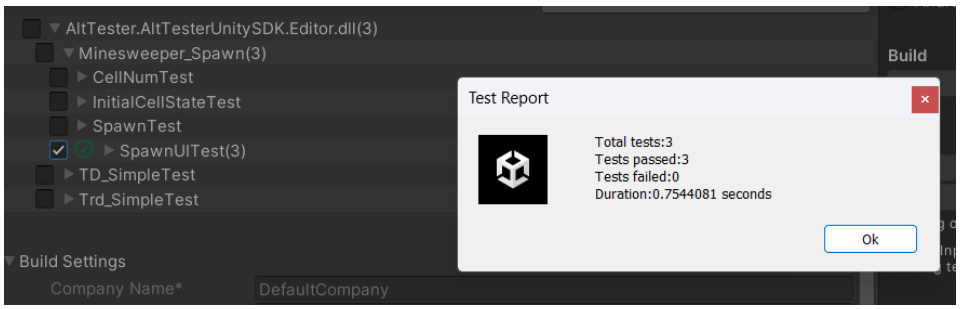
5. SpawnTest：test cell UI spawn

```
//test spawn UI cells
var cells = altDriver.FindObjectsWhichContain(By.NAME, "Cell");
Assert.AreEqual(AnsExpected, cells.Count);
```

```
//test UI cells position
string propertyName = "width";
var propertyValue = boardManager.GetComponentProperty<int>(componentName, propertyName, assemblyName);
var cellPos = cells[0].GetComponentProperty<dynamic>("UnityEngine.RectTransform", "position", "UnityEngine.CoreModule");
var cellPosW = cells[1].GetComponentProperty<dynamic>("UnityEngine.RectTransform", "position", "UnityEngine.CoreModule");
var cellPosH = cells[propertyValue].GetComponentProperty<dynamic>("UnityEngine.RectTransform", "position", "UnityEngine.Co

//test cellWidth
propertyName = "cellWidth";
var cellWidth = boardManager.GetComponentProperty<float>(componentName, propertyName, assemblyName);
Assert.AreEqual(Math.Round((double)cellWidth, 2), Math.Round((double)(cellPosW["x"] - cellPos["x"]), 2));

//test cellHeight
propertyName = "cellHeight";
var cellHeight = boardManager.GetComponentProperty<float>(componentName, propertyName, assemblyName);
Assert.AreEqual(Math.Round((double)cellHeight, 2), Math.Round((double)(cellPosH["y"] - cellPos["y"]), 2));
```




```
// spawn cells UI
foreach(Transform obj in Board) Destroy(obj.gameObject);
for(int y = 0;y<height;y++)
{
    for(int x = 0;x<width;x++)
    {
        GameObject cell = Instantiate(CellPrefab);
        cell.transform.SetParent(Board, false);
        cell.name = "Cell_" + GetIndex(x, y).ToString();
        cell.transform.localPosition = new Vector3(x*cellWidth, y*cellHeight, 0);
        if(mode == 1) cell.transform.localScale *= 1;
        else if(mode == 2 || mode == 4) cell.transform.localScale *= 0.7f;
        else if(mode == 3) cell.transform.localScale *= 0.5f;
    }
}
```



6. GameTest：init -> gaming

```
[Test]
public void GameStateTest()
{
    //get init game state
    AltObject gameManager = altDriver.FindObject(By.NAME, "GameManager");
    string componentName = "GameManager";
    string propertyName = "GameState";
    string assemblyName = "Assembly-CSharp";
    var propertyValue = gameManager.GetComponentProperty<string>(componentName, propertyName, assemblyName);

    Assert.AreEqual("0", propertyValue);

    //spawn board
    AltObject boardManager = altDriver.FindObject(By.NAME, "BoardManager");
    string methodName = "SpawnBoard";
    componentName = "BoardManager";
    assemblyName = "Assembly-CSharp";
    object[] parameters = new object[] {1};
    var data = boardManager.CallComponentMethod<int>(componentName, methodName, assemblyName, parameters);

    //get gaming state
    AltObject gameManagerAfter = altDriver.FindObject(By.NAME, "GameManager");
    componentName = "GameManager";
    propertyName = "GameState";
    var propertyAfter = gameManagerAfter.GetComponentProperty<string>(componentName, propertyName, assemblyName);

    Assert.AreEqual("1", propertyValue);

}
```

```
[05:00:02] 2025-06-05 05:00:02.5819|Editor|ERROR| String lengths are both 1. Strings differ at index 0.
  Expected: "1"
[05:00:02] 2025-06-05 05:00:02.5819|Editor|INFO|==========================================
[05:00:02] 2025-06-05 05:00:02.5819|Tester|DEBUG|command received: {"messageId": null,"driverId": "","comma
[05:00:02] 2025-06-05 05:00:02.5819|Teste                                                  d":null,"
[05:00:02] 2025-06-05 05:00:02.5910|Teste                                                  mandN
2025-06-05 05:00:02.5819|Editor|ERROR| String le
Expected: "1"
But was:  "0"
-----------^
```

Test Report

Total tests:1
Tests passed:0
Tests failed:1
Duration:0.2503351 seconds

Ok

```
public class GameManager : MonoBehaviour
{
    public enum State {init, gaming, win, loss};
    public State GameState = State.init;

    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }

    public void ChangeState(State st)
    {
        GameState = st;
    }
}
```

```
// start game, turn game state to "gaming"
GMaster.ChangeState(GameManager.State.gaming);
```

AltTester© Editor

Assembly: All

Tests list: Refresh

- AltTester.AltTesterUnitySDK.Editor.dll(1)
  - Minesweeper_Game(1)
    - GameStateTest
  - Minesweeper_Spawn
  - TD_SimpleTest
  - Trd_SimpleTest

Test Report

Total tests:1
Tests passed:0
Tests failed:1
Duration:0.044967 seconds

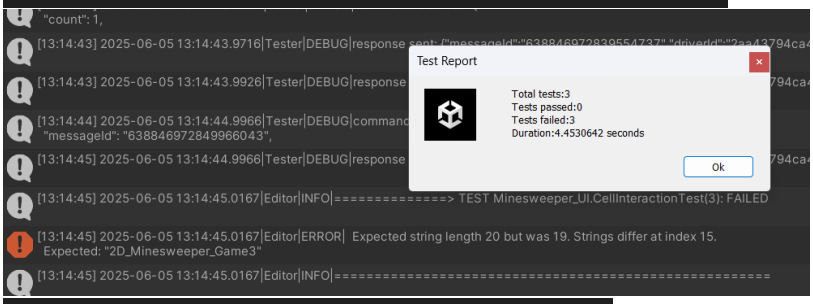
Ok

Build Settings

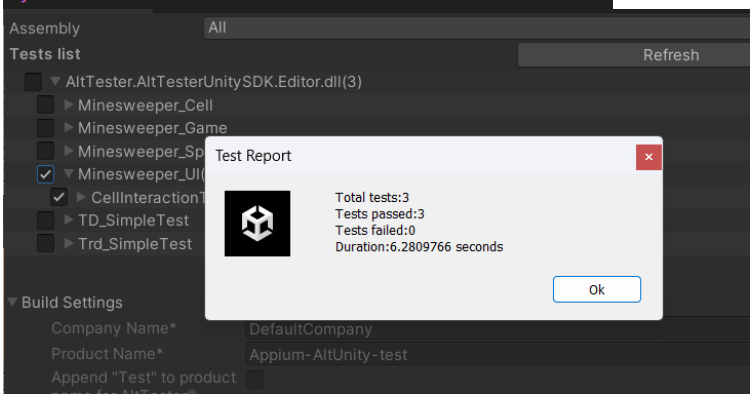
Company Name*: DefaultCompany

7. UITest：

```
[TestCase(1)]
[TestCase(2)]
[TestCase(3)]
public void CellInteractionTest(int mode)
{
    //click Easy UI
    if(mode == 1)
    {
        altDriver.LoadScene("2D_Minesweeper_Menu");
        var cells = altDriver.FindObjectsWhichContain(By.NAME, "ButtonEasy");
        cells[0].Click();
        Thread.Sleep(1000);
        Assert.AreEqual("2D_Minesweeper_Game1", altDriver.GetCurrentScene());
    }
    //click Medium UI
    if(mode == 2)
    {
        altDriver.LoadScene("2D_Minesweeper_Menu");
        var cells = altDriver.FindObjectsWhichContain(By.NAME, "ButtonNormal");
        cells[0].Click();
        Thread.Sleep(1000);
        Assert.AreEqual("2D_Minesweeper_Game2", altDriver.GetCurrentScene());
    }
    //click Hard UI
    if(mode == 3)
    {
        altDriver.LoadScene("2D_Minesweeper_Menu");
        var cells = altDriver.FindObjectsWhichContain(By.NAME, "ButtonHard");
        cells[0].Click();
        Thread.Sleep(1000);
        Assert.AreEqual("2D_Minesweeper_Game3", altDriver.GetCurrentScene());
    }
}
```




```
public void OnButtonEasyClicked()
{
    SceneManager.LoadScene("2D_Minesweeper_Game1");
}

public void OnButtonNormalClicked()
{
    SceneManager.LoadScene("2D_Minesweeper_Game2");
}

public void OnButtonHardClicked()
{
    SceneManager.LoadScene("2D_Minesweeper_Game3");
}
```



------還有其他……

8. Cell Logic：Click to show up
9. Cell Logic：Right Click to Flag
10. Cell Logic：Recursive Case
11. Cell Logic：Click to explosion
12. GameTest：Start/Reopen the game
13. GameTest：Menu Difficulty
14. TimerTest