

情報学群実験第 1 最終レポート

1250373 溝口洸熙 *

2022 年 7 月 28 日

概要

目次

はじめに	2
仕様 1	3
1.1 処理概要	3
1.2 処理	4
仕様 2	5
2.1 処理概要	5
2.2 処理	6
仕様 3	7
ソースコード	8

* 高知工科大学 情報学群 2 年生

はじめに

レポートについて

このレポートは， $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X } 2_{\epsilon}$ を用いて作成している．図やグラフは TikZ を用いて描画しており，ソースコードは `listing` を用いて表記している．

符号化と変数

あるパネルのステータスを示す符号と，新たに追加したグローバル変数を，以下に示す．

符号とステータス		新たに追加した変数	
符号	ステータス	変数名	役割
0	爆弾以外	<code>int originalTable</code>	生成した盤面の初期状態を記憶する．
1	開かれたパネル		
-1	爆弾	<code>Boolean tr</code>	2 手目以降で <code>true</code> になる変数．
-2	フラグが立っている		1 手目で爆弾に当たることを回避するため．

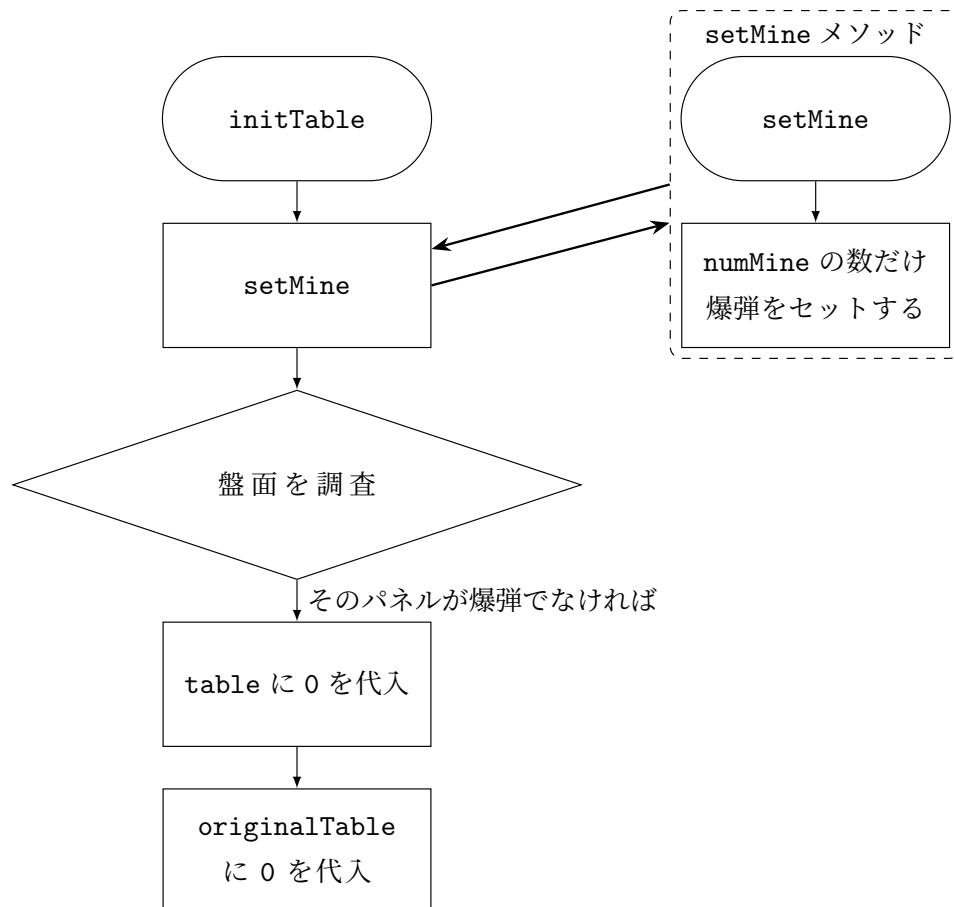
仕様 1

仕様 1.

ゲーム開始時に、盤面上へランダムに地雷を設置する.

1.1 処理概要

Fig 1.1: 盤面上へランダムに地雷を設置する



1.2 処理

initTable (src. 1) の処理

盤面を初期化するにあたって、以下の処理を行う。

- 1) 爆弾の設置パネルを setMine メソッドで定める。
- 2) table の全ての行と列に 0 を代入する。ただし、爆弾であるパネルは上書きしない。

```
for (int x = 0; x < this.height; x++) {
    for (int y = 0; y < this.width; y++) {
        if (this.table[x][y] == -1) { // 爆弾がセットされている場所は避ける
            continue;
        }
        this.table[x][y] = 0; // 爆弾の場所以外は0で初期化
    }
}
```

- 3) originalTable の全ての行に 0 を代入する。ただし、爆弾であるパネルは上書きしない。

```
this.originalTable[x][y] = 0;
```

setMine (src. 2) の処理

盤面に爆弾を配置するにあたって、以下の処理を行う。

- 1) 爆弾の個数を数える count 変数を定義する。
- 2) 指定された爆弾の個数が count になるまで、爆弾を配置する。

```
while (count != this.numMine) { //
    numMine の数だけ爆弾をセットできたらループを抜ける
    ...
}
```

- 3) 爆弾の配置はランダムである。乱数で指定されたパネルが既に爆弾であれば再度乱数を生成し、爆弾が新たにセットできる場所では、table,originalTable の乱数値 Index を -1 に設定し、count をインクリメントする。

```
...
int x = new java.util.Random().nextInt(getHeight());
int y = new java.util.Random().nextInt(getWidth());
if (this.table[x][y] == -1) {
    // [x][y] にすでに爆弾がセットされていたら、もう一度乱数を決め直す
    continue;
}
count++;
...
```

仕様 1 終

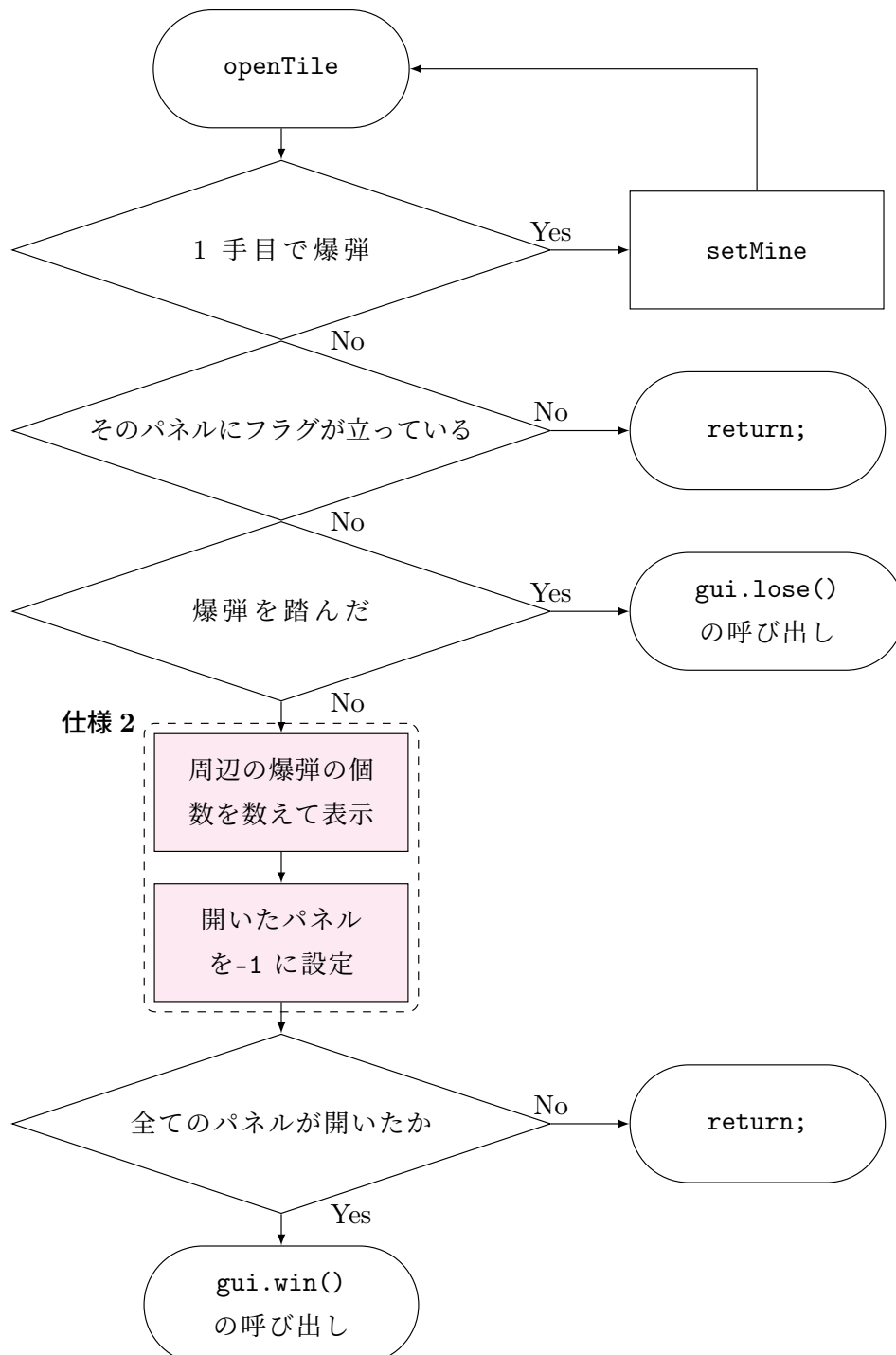
仕様 2

処理 2.

パネルを左クリックした際、クリックしたパネルを開く.

2.1 処理概要

Fig 2.2: タイルを開くときの処理



2.2 処理

openTile (src. 3) の処理

- 1) 1 手目で爆弾のタイルを開けてゲームオーバーではゲーム性が失われかねないので、1 手目で爆弾である場合は、再度爆弾をセットし直し、もう一度同じタイルを開ける処理を行う。この際に仕様する変数が tr であり、tr=false の時は 1 手目、tr=true の時は 2 手目以上であることを示している。

```
if (this.table[x][y] == 1 && !this.tr) { // 1手目で爆弾ならば再度爆弾をセ  
    ット  
    this.setMine();  
    this.openTile(x, y, gui);  
}
```

- 2) 開けたパネルに爆弾があれば、全てのタイルを表示後、"You lose"のポップアップを表示する。

```
if (this.table[x][y] == -1) { // パネルに爆弾があった場合  
    this.openAllTiles(gui);  
    gui.lose();  
}
```

- 3) もし、その場所にフラグが立っているのであれば、何もせずに return;。この機能は追加した。

```
else if (this.table[x][y] == -2) { // フラグがパネルに立っている場合  
    return;  
}
```

- 4) パネルに爆弾がない場合、その周辺の爆弾個数を returnMine メソッドで取得し、そのパネルに表示する。

- 5) 全てのパネルが開いたか否か確認する。

```
int mineCount = this.returnMine(x, y, gui); // 周辺の爆弾の個数を調査  
this.table[x][y] = 1; // 開かれたパネルの値を1に設定  
...  
Boolean jud = true;  
for (int i = 0; i < getHeight(); i++) {  
    for (int j = 0; j < getWidth(); j++) {  
        if (this.table[i][j] == 0) {  
            // 開いていないパネルがある場合 judの値をtrueにする  
            jud = true;  
            break;  
        } else { jud = false; }  
    }  
    if (jud) { break; }  
}  
if (!jud) { gui.win(); }
```

仕様 3

仕様 3.

ソースコード

src. 1: initTable

```
1 void initTable() { // 盤面を初期化する
2     this.setMine();
3     for (int x = 0; x < this.height; x++) {
4         for (int y = 0; y < this.width; y++) {
5             if (this.table[x][y] == -1) { // 爆弾がセットされている場所は避ける
6                 continue;
7             }
8             this.table[x][y] = 0; // 爆弾の場所以外は0で初期化
9             this.originalTable[x][y] = 0;
10        }
11    }
12 }
```

src. 2: setMine

```
1 void setMine() { // 爆弾をセット
2     int count = 0;
3     while (count != this.numMine) {
4         // numMineの数だけ爆弾をセットできたらループを抜ける
5         int x = new java.util.Random().nextInt(getHeight());
6         int y = new java.util.Random().nextInt(getWidth());
7         if (this.table[x][y] == -1) {
8             // [x][y]にすでに爆弾がセットされていたら、もう一度乱数を決め直す
9             continue;
10        }
11        this.table[x][y] = -1; // 爆弾の場所を値-1としてセットする
12        this.originalTable[x][y] = -1;
13        count++;
14    }
15 }
```

src. 3: openTile

```
1 public void openTile(int x, int y, MinesweeperGUI gui) {
2     if (this.table[x][y] == 1 && !this.tr) { // 1手目で爆弾ならば再度爆弾をセ
3         ット
4         this.setMine();
5         this.openTile(x, y, gui);
6     }
7     this.tr = true;
8     if (this.table[x][y] == -1) { // パネルに爆弾があった場合
9         this.openAllTiles(gui); gui.lose();
10    } else if (this.table[x][y] == -2) { return; } // パネルにフラグが立っ
11    ている場合
12    else { // パネルに爆弾がなかった場合
```



```

11         int mineCount = this.returnMine(x, y, gui); // 周辺の爆弾の個数を
           調査
12         this.table[x][y] = 1; // 開かれたパネルの値を1に設定
13         String mc = String.valueOf(mineCount);
14         gui.setColorText(x, y, mineCount); gui.setTextToTile(x, y, mc);
15         Boolean jud = true;
16         for (int i = 0; i < getHeight(); i++) { // 爆弾以外のパネルが全て
           開いているか確認
17             for (int j = 0; j < getWidth(); j++) {
18                 if (this.table[i][j] == 0) {
19                     // 開いていないパネルがある場合 judの値をtrueにする
20                     jud = true;
21                     break;
22                 } else { jud = false; } // 開いている場合 falseにする
23             }
24             if (jud) { break; }
25         }
26         if (!jud) { gui.win(); }
27     }
28 }

```