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

* 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，LaTeX 2_ε 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

符号	ステータス
0	爆弾以外
1	開かれたパネル
-1	爆弾
-2	フラグが立っている

新たに追加した変数

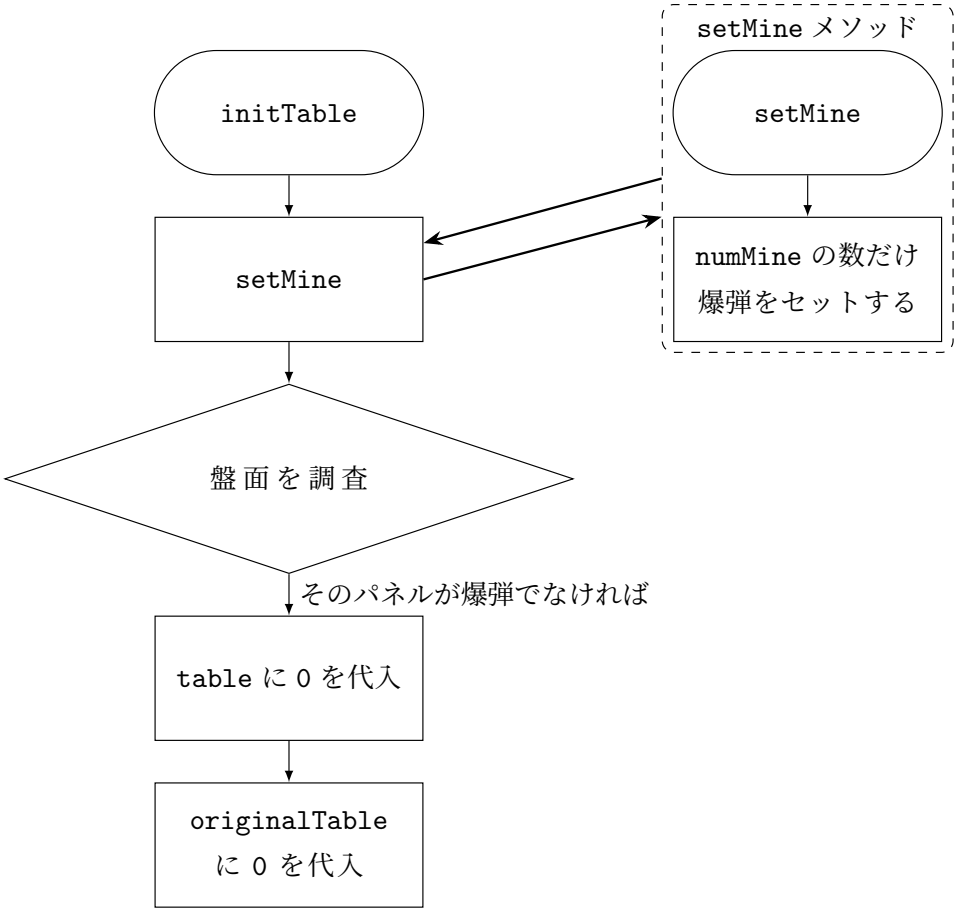
変数名	役割
`int originalTable`	生成した盤面の初期状態を記憶する．
`Boolean tr`	2 手目以降で `true` になる変数． 1 手目で爆弾に当たることを回避するため．

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++;
...
```

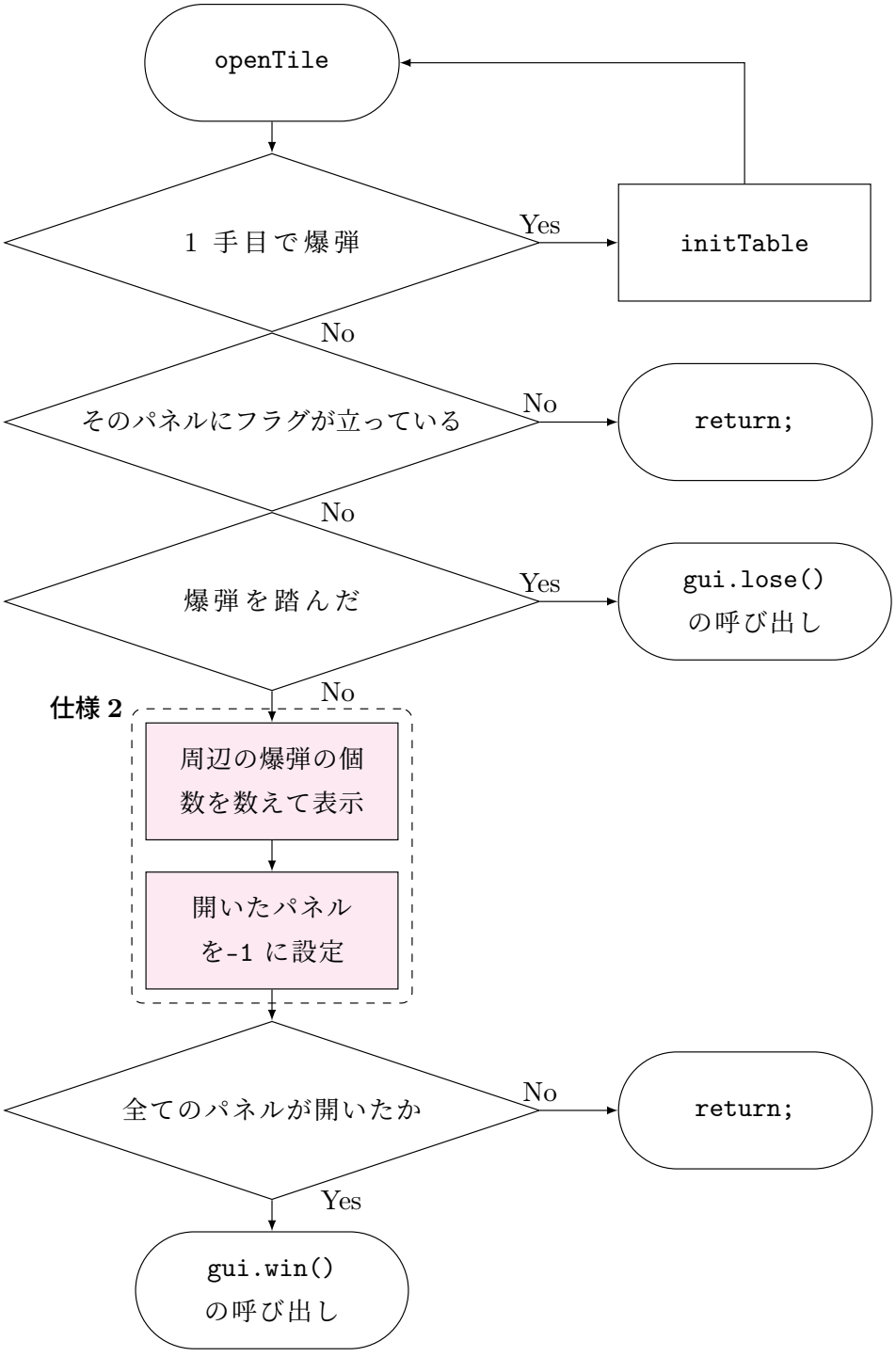
仕様 1 終

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) の処理の一部

1) パネルに爆弾がない場合，その周辺の爆弾個数を `returnMine`（src. 4）メソッドで取得し，そのパネルに表示する．
2) 全てのパネルが開いたか否か確認する．もし，全てのパネルが開いていたら勝利となるので `gui.win` を呼び出し，開いていないパネルが存在すれば，`return;` 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
...
// 爆弾以外のパネルが全て開いているか確認
for (int i = 0; i < getHeight(); i++) {
    for (int j = 0; j < getWidth(); j++) {
        if (this.table[i][j] == 0) { return;}
    }
}
gui.win();
```

仕様 3

仕様 3.

ソースコード

src. 1: initTable

```
void initTable() {// 盤面を初期化する
  this.setMine();
    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
          this.originalTable[x][y] = 0;
      }
  }
}
```

src. 2: setMine

```
void setMine() {// 爆弾をセット
  int count = 0;
  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
    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;
    }
    this.table[x][y] = -1; // 爆弾の場所を値-1としてセットする
    this.originalTable[x][y] = -1;
    count++;
  }
}
```

src. 3: openTile

```
public void openTile(int x, int y, MineSweeperGUI gui) {
    if (this.table[x][y] == 1) {
        return;
    }
    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.initTable();
        this.openTile(x, y, gui);
    }
    this.tr = true;
    if (this.table[x][y] == -1) { // パネルに爆弾があった場合
        this.openAllTiles(gui);
```

```
12          gui.lose();
13      } else if (this.table[x][y] == -2) { // パネルに旗が立っている場合
14          return;
15      } else { // パネルに爆弾がなかった場合
16          int mineCount = this.returnMine(x, y, gui);// 周辺の爆弾の個数を調査
17          this.table[x][y] = 1; // 開かれたパネルの値を1に設定
18          if (mineCount == 0) {// もし周辺に爆弾がなければ，そのマスも開ける．
19              for (int i = x - 1; i < x + 2; i++) {
20                  if (i < 0 || i >= getHeight()) { // パネルの範囲外は除く
21                      continue;
22                  }
23                  for (int j = y - 1; j < y + 2; j++) {
24                      if (j < 0 || j >= getWidth()) { // パネルの範囲外は除く
25                          continue;
26                      }
27                      openTile(i, j, gui);
28                  }
29              }
30          }
31          String mc = String.valueOf(mineCount);
32          if (mineCount == 0) {
33              gui.setGray(x, y);
34          } else {
35              gui.setColorText(x, y, mineCount);// 色を設定
36              gui.setTextToTile(x, y, mc); // 爆弾の個数を表示
37          }
38          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
39              for (int j = 0; j < getWidth(); j++) {
40                  if (this.table[i][j] == 0) {
41                      return;
42                  }
43              }
44          }
45          gui.win();
46      }
47  }
```

src. 4: returnMine

```
1   public int returnMine(int x, int y, MineSweeperGUI gui) {
2   // x,yの周り8マス分の爆弾を調査
3       int mineCount = 0;
4       for (int i = x - 1; i < x + 2; i++) {
5           if (i < 0 || i >= getHeight()) { // パネルの範囲外は除く
6               continue;
7           }
8           for (int j = y - 1; j < y + 2; j++) {
9               if (j < 0 || j >= getWidth()) { // パネルの範囲外は除く
10                  continue;
11              }
```

```
12                  if (this.originalTable[i][j] == -1) { // 爆弾の個数をカウント
13                      mineCount++;
14                  }
15              }
16          }
17          return mineCount;
18      }
```