Chapter 7

ゲームを作ろう

とうとう最後の課題である。コンピューターを使ったゲームが好きな学生は諸君の中に少なからずいるであろう。であるからして、最後の課題はゲームの制作としよう。

ただし、3Dなワールドで「ぐりぐり」動くゲームを作るのは敷居が高い。そこで、テキストベースで簡単に作れるゲームを作ってみる。

7.1 ncursesライブラリを使う

ncursesは、端末上でテキストベースのユーザーインターフェイスを実現するための APIを提供する、C言語のためのライブラリである。mathライブラリ同様[1]、自分で作ったプログラムにライブラリをリンクして利用する。

ncursesライブラリには、いろいろな関数が用意されている[2]。以下はその一部である[3]。

まずは、初期化に必要となる関数を挙げる。

- initscr()

  画面の初期化を行う。この関数が呼ばれると、ncurses.hで定義されているグローバル変数COLS、LINESが使えるようになる。COLSはターミナル画面の横幅（横に何列分の文字が入るか）を示し、LINESはターミナル画面の縦幅（縦に何行分の文字が入るか）を示す。

[1] 言わずと知れた数学の関数などが使えるライブラリである。
[2] 後に述べるが、必ずncurses.hをインクルードすること。
[3] より詳しく知りたい場合、ターミナル上でman ncursesを実行せよ。

- noecho()

  この関数が呼ばれると、入力されたキーの値が画面に直接出力されない。

- cbreak()

  この関数が呼ばれると、入力されたキーの値が即座に反映されるモードになる。

- curs_set(int i)

  カーソルの色を設定する。本演習では、カーソルを見えなくするためcurs_set(0) とする。

- timeout(int delay)

  ミリ秒単位で画面を止める。timeout(1000)で1秒画面が止まることになる。

また、画面の描画に関係する関数に次のようなものがある。

- int getch()

  キーボードから一文字読み込む。ただし、事前にcbreak関数を呼んでおかないと、 Enterキーが押されるまで待ちに入ってしまうことに注意。

- mvprintw(int row, int column,const char *s , ···)

  printf関数と似ている。例えば、mvprintw(1,3,“1+1=%d”,2);とすると、 %dのところに2が入り、1+1=2が画面に表示される。第一引数の1、第二引数の3は、それぞれ、 1+1=2の先頭（一番左の1)が上から2行目、左から4列目の位置に来るように表示させることを示している。

- refresh()

  以上の関数を使って画面を操作した内容を画面に反映する。

- endwin()

  ncursesの画面モードから抜けて、もとのターミナルの状態に戻す。処理終了部分で呼び出す必要がある。

繰り返しになるが、これらの関数を利用する際には、次のようにヘッダファイル ncurses.hをインクルードする必要がある[4]。

```
#include <ncurses.h>
```

また、このヘッダファイルに定義されている関数は自動的にリンクされるわけではないので、次のようにgccコマンドのオプションでリンクする旨を明示的に示す必要がある。

```
gcc ncurse-test.c redraw.c -lncurses -o ncurse-test
```

というように -lncursesオプションをコンパイル時に付けておく必要がある [5]。

7.2 小手調べ:「モンスター」を動かしてみよう

モンスターといったって、ものすごい形相のモンスターをかっこいいグラフィックで・・・なんていうのはこの道具立てだと手に余る。であるので、大胆な単純化を行う。すなわち

モンスターをMで表す

ものとする[6]。また、ncurseライブラリを初めて使う人がほとんどであろう。まずは、簡単にモンスターにブラブラ[7]してもらおう。

今回の課題では以下の二つの関数を実装しよう。

- void redraw()

  以下の処理を行う。

  - ncursesの「初期化に必要となる関数」で挙げた5つの関数を呼び出す。ただし、curs_set関数の引数は0とし、timeoutの引数は100とすること。

[4]他にも必要なヘッダファイルがあれば、それもインクルードしないといけないことはいうまでもない。

[5]これを忘れると、あれやこれやが定義されていないからリンクできない！とgccに怒られる。

[6]昔、Unixの黎明期のゲームであるRogueでは、モンスター=ホブゴブリンはHで表されていた。

[7]Random walkってやつだ。

- 構造体POSITIONを型とするポインタ変数monsterを用意する[8]。 monsterのxを列数(COLS)の半分、yを行数(LINES)の半分となるようにする[9]。
- char型の変数chを用意し、‘q’と等しくない間は以下を繰り返す。
  - 現在のmonsterの位置に半角スペースを書き込む [10]。
  - 次に作るgetMonsterLocation関数を呼び出す。引数にはmonsterをセットする [11]。
  - 更新されたmonsterの位置に半角アルファベットのMを書き込む。
  - getch関数を呼びだし、戻り値をchに代入する。
  - refresh関数を呼び出す。
- 繰り返しから抜けたらendwin関数を呼び出し、処理を終了する。

- void getMonsterLocation(POSITION *monster)
  - int型のdx、dyを宣言する。getMonsterLocation関数を実行するたびに、 dxとdyにそれぞれ、-1,0,1のいずれかの値をランダムに代入する。 [12]
  - monsterのxの値が0よりも大きくかつdxが負である場合、または、monsterのxの値が COLS-1より小さくかつdxが正である場合、monsterのxの値にdxを加える。 [13]、
  - monsterのyの値が0よりも大きくかつdyが負である場合、または、monsterのyの値が LINES-1より小さくかつdyが正である場合、monsterのyの値にdyを加える。 [14]

[8]構造体POSITIONはヘッダファイルredraw.hにて定義する。
[9]モンスターを画面の真ん中に置きたいのだ。
[10]mvprintw関数を使うが、第一引数がyに、第二引数がxに相当することに注意。
[11]新しい位置を設定することに相当する。
[12]rand関数を用いよ。
[13]つまり、横方向に画面からはみ出ないようにしたいわけであるな。
[14]同様に、縦方向に画面からはみ出ないようにしたいのだ。

なお、構造体POSITIONは以下に示すヘッダファイルredraw.hにて定義する。

```
#ifndef REDRAW_H_
#define REDRAW_H_

struct position{
 int x, y;
};
typedef struct position POSITION;

void redraw();
void getMonsterLocation(POSITION *monster);

#endif /* REDRAW_H_ */
```

また、redraw関数とgetMonsterLocation関数の実装はファイルredraw.cに記述するものとする。実行するためのプログラムは下のように与えられる(ncurse-test.c)。

```
#include "redraw.h"

int main(int argc, char **argv){
        redraw();
}
```

今回の課題は以上である。