

プログラミング及び演習

第15回 プログラミングプロジェクト

(2017/07/28)

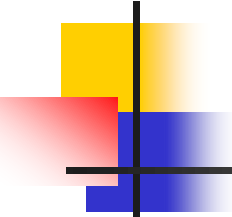
講義担当

大学院情報学研究科知能システム学専攻

教授 森 健策

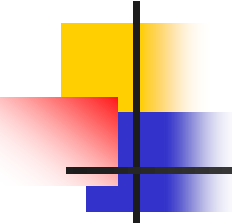
大学院情報学研究科知能システム学専攻

助教 小田 昌宏



本日の講義・演習の内容

- いよいよこれでプログラミング及び演習も最終回です
 - みなさんのプログラミング能力は向上しましたか？
 - 講義に対する感想、ならびに、この半期の自分の進歩の程度などを回答1に記入してください
 - 回答2、3は空白で構いません。
 - 2017/07/28 第2限出席テストに回答してください
- 内容
 - プログラミングプロジェクト課題の再確認
 - 演習



デバッグの基本1

■ その1 「状態を知る」

■ プログラム内部の状態を表示する

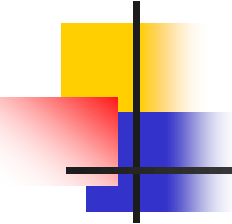
- ブロック崩しの場合,各変数の内容を右側に一時的に表示

- ```
char text[1024];
 sprintf(text, "Delay %d", delay);
 mvaddstr(y, x, text);
```

- プロセスアタッチによるデバッグも有効です

## ■ その2 「問題を切り分ける」

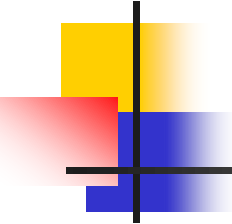
- いったんすべての関数呼び出しをコメントアウトし、順次コメントアウトを外す
- ある関数をテストする小さなプログラムを作る



# デバッグの基本2

---

- その3 「ポインタには要注意」
  - Segmentation faultは発生しないが、不正なポインタアクセスをしている場合が多々ある
- その4 「アルゴリズムの間違いも結構多い」
  - 正しいと思っていたアルゴリズムが間違っている場合はよくあるはず。自分の考えたアルゴリズムが間違っていないか常に疑う



# プログラミングプロジェクト課題

- テトリスゲームを作成
- 各自のアイデアを取り込み、より充実したゲームを作成欲しい
- 構造体、線形リスト、双方向リスト、ファイルなど講義で学んだことを最大限生かしてプログラムを作成する
- 余力があればネットワーク対戦型を作成
- 本課題提出は単位取得の必須条件
  - (通称: 夏休み課題)
- 締め切り
  - 2017/8/10 17:00
  - NUCTに記されている指示に従ってください。NUCTに掲載されている情報が優先されます。

# 仕様 (1/4)

## ■ 画面構成

- 次ページ参照
- 画面サイズに合わせて縦・横に配置できるブロックの数・各種情報の表示位置を適宜変化させる
- 画面サイズを変更しても描画エリアが変化しないプログラムは認められない
- ブロックは#記号、枠は|,-記号で表示
- 右側に経過時間、Level, 消した行数 次とその次のブロック, Score, High Score, High Scoreを出した人の名前を表示

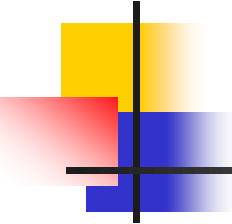
## ■ ブロック



ならびにこれらを90, 180, 270度回転させたもの

# 画面構成

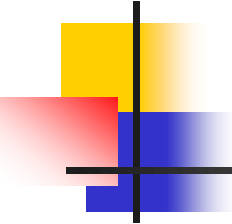
```
0 1 2 3 4 5 6 7
01234567890123456789012345678901234567890123456789
00 *=====*
 1 | |
 2 | | Time AAAAA
 3 | | Level AAAAA
 4 | | Score AAAAA
 5 | | High Score AAAAA
 6 | | Name Def GHi
 7 | # Next Blocks
 8 | # First Second
 9 | # +---+ +---+
10 | | | | # |
 1 | | | | # |
 2 | | | | # |
 3 | | | | # |
 4 | | | | +---+ +---+
 5 | | | |
 6 | | | |
 7 | | | |
 8 | | | |
 9 | ###
20 | ###
 1 | ###
 2 | ### # ###
 3 | ### # ###
 4 | #####
```



# 仕様 (2/4)

---

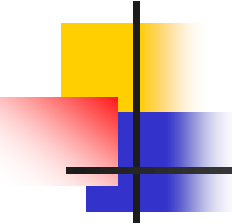
- ブロックの動き
  - 下方向に落ちブロックが積み上げる
    - ブロックが食い込むことのないように
  - 画面上部・中央付近から落ちる
  - スペースを押すと回転する
  - 下向き矢印を押すとブロックは強制落下
  - 一行そろうとその行は消滅。その上にあるブロックは一行さがる
  - ブロックがある高さまで積みあがるとゲームオーバー
  - 一定点数以上になればレベルアップ
  - レベルアップ毎に落下速度を上げる



## 仕様 (3/4)

---

- ゲームエリア
  - ゲームエリアはプログラム内で変更できるようにする
- Scoreの計算
  - 消された行数と落下速度から計算
- 経過時間
  - ゲーム開始からの経過時間を表示する
- 次に落下予定のブロックを右側に表示する



# 仕様 (4/4)

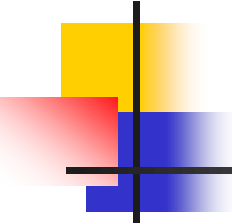
---

- ゲーム開始時

- 過去上位5位のHigh Score List (得点、氏名)をファイルから読み込む

- ゲーム終了時

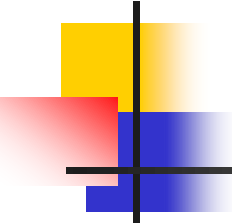
- 上位5位以内であれば、名前を入力を促す
- 過去上位5位のHigh Score Listを表示
- 過去上位5位のHigh Score List (得点、氏名)をファイルに書き出す



# 簡単なゲーム (1/8)

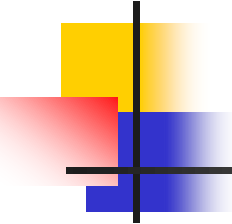
---

```
#include < curses.h>
void go();
int main(int argc, char **argv) {
 initscr();
 start_color();
 noecho();
 cbreak();
 curs_set(0);
 keypad(stdscr, TRUE);
 go();
 endwin();
 return 0;
}
```



## 簡単なゲーム (2/8)

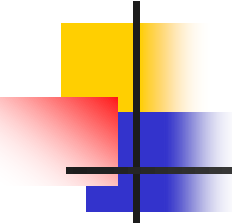
```
#define PATNUM 3
#define PATSIZEW 3
#define PATSIZEH 3
static char
 blockPattern[PATNUM][PATSIZEH][PATSIZEW] =
 {
 {{ '#', '#', '#'}, {' #', '#', '#'}, {' #', '#', '#'}},
 {{ ' ', '#', ' '}, {' ', '#', ' '}, {' ', '#', ' '}},
 {{ ' ', ' ', ' '}, {' #', '#', '#'}, {' ', ' ', ' '}}
 };
```



## 簡単なゲーム (3/8)

---

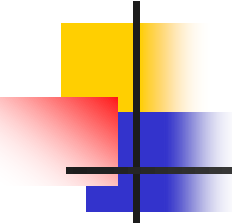
```
int collisionBottomWall(int blocX,
 int blocY, int bottom, int
 pattern)
{
 int flag = 0;
 if(blocY+2 == bottom) {
 flag = 1;
 }
 return flag;
}
```



# 簡単なゲーム (4/8)

---

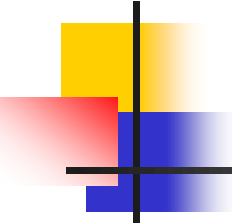
```
void go()
{
 int locX, locY;
 int locXdir = 1;
 int blocX, blocY;
 int ch;
 int delay=0;
 int waitCount = 20000;
 char msg[256];
 int pattern = 0;
 char buffer[256][256];
 int i, j;
 int ii, jj;
 blocX = COLS/4;
 blocY = 5;
 for(j=0;j<256;j++){
 for(i=0;i<256;i++){
 buffer[j][i]=' ';
 }
 }
 timeout(0);
```



# 簡単なゲーム (5/8)

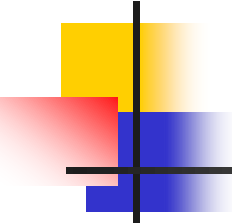
```
while ((ch=getch()) != 'Q') {
 mvaddstr(blocY,blocX," ");
 mvaddstr(blocY+1,blocX," ");
 mvaddstr(blocY+2,blocX," ");

 if(delay%waitCount==0) {
 blocY += 1;
 }
 if(blocX<0) {
 blocX =0;
 beep();
 }
 if(blocX>=COLS/2) {
 blocX =COLS/2-1;
 beep();
 }
 delay++;
}
```



# 簡単なゲーム (6/8)

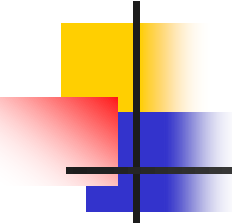
```
switch(ch) {
 case KEY_LEFT:
 blocX -=1;
 break;
 case KEY_RIGHT:
 blocX +=1;
 break;
 case ` `:
 if(pattern == 1) {
 pattern = 2;
 }else if(pattern == 2) {
 pattern = 1;
 }
 beep();
 break;
 default:
 break;
}
```



## 簡単なゲーム (7/8)

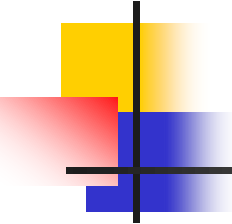
```
for (j=0; j<LINES; j++) {
 for (i=0; i<COLS/2; i++) {
 mvaddch(j, i, buffer[j][i]);
 }
}

for (jj=0; jj<3; jj++) {
 for (ii=0; ii<3; ii++) {
 mvaddch(blocY+jj, blocX+ii,
 blockPattern[pattern][jj][ii]);
 }
}
```



# 簡単なゲーム (8/8)

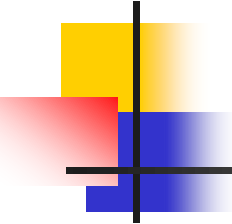
```
if(collisionBottomWall(blocX, blocY, LINES-3,
 pattern)) {
 for(jj=0; jj<3; jj++){
 for(ii=0; ii<3; ii++){
 buffer[blocY+jj][blocX+ii] =
 blockPattern[pattern][jj][ii];
 }
 }
 beep();
 blocY = 5;
 pattern = (pattern+1)%PATNUM;
}
sprintf(msg, "X %03d Y %03d Pat %02d", blocX, blocY,
 pattern);
mvaddstr(2, COLS-20, msg);
}
}
```



# cursesで色付け

---

```
int main(int argc, char **argv){
 char text[128];
 initscr();
 start_color();
 noecho();
 cbreak();
 curs_set(0);
 init_pair(1, COLOR_CYAN, COLOR_BLACK);
 init_pair(2, COLOR_RED, COLOR_BLACK);
 init_pair(3, COLOR_YELLOW, COLOR_BLACK);
 init_pair(4, COLOR_WHITE, COLOR_BLACK);
 bkgd(COLOR_PAIR(4));
 keypad(stdscr, TRUE);
 go();
 endwin();
 return 0;
}
```



# cursesで色付け

```
void go()
{
 ...
 char buffer[256][256];
 char color_buffer[256][256];
 ...
 for(j=0;j<256;j++){
 for(i=0;i<256;i++){
 buffer[j][i]=' ';
 color_buffer[j][i] = 0;
 }
 }
 while((ch=getch())!='Q'){
 attron(COLOR_PAIR(4));
 mvaddstr(blocY,blocX," ");
 mvaddstr(blocY+1,blocX," ");
 mvaddstr(blocY+2,blocX," ");
 ...
 for(j=0; j<LINES; j++){
 for(i=0; i<COLS/2; i++){
 attron(COLOR_PAIR(color_buffer[j][i]));
 mvaddch(j, i, buffer[j][i]);
 }
 }
 }
}
```

```
attron(COLOR_PAIR(pattern+1));
for(jj=0; jj<3; jj++){
 for(ii=0; ii<3; ii++){
 mvaddch(blocY+jj, blocX+ii, blockPattern[pattern][jj][ii]);
 }
}
if(collisionBottomWall(blocX, blocY, LINES-3, pattern)){
 for(jj=0; jj<3; jj++){
 for(ii=0; ii<3; ii++){
 buffer[blocY+jj][blocX+ii] = blockPattern[pattern][jj][ii];
 color_buffer[blocY+jj][blocX+ii] = pattern+1;
 }
 }
 beep();
 blocY = 5;
 pattern = (pattern+1)%PATNUM;
}
sprintf(msg, "X %03d Y %03d Pat %02d", blocX, blocY, pattern);
attron(COLOR_PAIR(4));
mvaddstr(2,COLS-20,msg);
}
}
```

# さらに工夫するとこんな画面も

