Getting Characters from the Terminal

- getch()
  - Get a character from the terminal
- getstr(str)
  - Get a string from the terminal
- scanw(fmt, arg1, arg2, …)
  - Formatted input from the terminal like scanf().

pdcurses_3.cpp

```
#include <curses.h>

int main()
{
   char text[10];
   int i, j;
   initscr();
   getstr(text);              // input the string "1,2"
   addstr(text); addch('\n');

   scanw("%d,%d", &i, &j);    // input the string "1,2" again
   printw("%d\t%d\n", i, j);

   getch();
   endwin();
   return 0;
}
```

noecho()

```
#include <curses.h>

int main()
{
   int c;
   initscr();
   // noecho();
   do {
        c = getch();
        printw(" %d\n", c);
   } while (c != '0');

   endwin();
   return 0;
}
```

pdcurses_4.cpp

```
// pdcurses_4.cpp
#include <curses.h>

int main()
{
   int y=10, x=10;
   char c;
   initscr();
   noecho();
   do {
        move(y, x); addch('Q');
        c = getch();
        move(y, x); addch(' ');
        switch (c)
        {
        case 'h':
                x--;
                break;
        case 'l':
                x++;
                break;
        case 'j':
                y++;
                break;
        case 'k':
                y--;
                break;
        }
   } while (c != 'q');

   endwin();
   return 0;
}
```

`curs_set()`

- `curs_set()` alters the appearance of the text cursor.
- `int curs_set(int visibility);`
  - A value of 0 for visibility makes the cursor disappear;
  - a value of 1 makes the cursor appear "normal" (usually an underline)
  - 2 makes the cursor "highly visible" (usually a block).

pdcurses_4a.cpp

```
// pdcurses_4.cpp
#include <curses.h>

int main()
{
   int y=10, x=10;
   char c;
   initscr();
   noecho();
   curs_set(0);  // no cursor
   do {
        move(y, x); addch('Q');
        c = getch();
        move(y, x); addch(' ');
        switch (c)
        {
        case 'h':
                x--;
                break;
        case 'l':
                x++;
                break;
        case 'j':
                y++;
                break;
        case 'k':
                y--;
                break;
        }
   } while (c != 'q');

   endwin();
   return 0;
}
```

Function Keys

- Call `keypad()` to enable the handling of Function keys and arrow keys.
  - `int keypad(WINDOW *win, bool bf);`
  - `keypad(stdscr, TRUE);`
- `getch()` returns an integer corresponding to the key pressed.
  - If it is a normal character, the integer value will be equivalent to the ASCII code of the character.
  - Otherwise it returns a number which can be matched with the constants defined in `curses.h`.
    - For example if the user presses F1, the integer returned is 265.

Function Keys (cont.)

- With keypad() enabled, you can check the returned value of getch() with the constants defined in curses.h
  - KEY_UP, KEY_DOWN, KEY_LEFT, KEY_RIGHT
  - KEY_HOME, KEY_END,
  - KEY_F(n)

Key Definitions

- `#define KEY_IC        0x14b   /* insert char or enter ins mode (Insert) */`
- `#define KEY_DC        0x14a   /* delete character (Delete) */`
- `#define KEY_HOME      0x106   /* home key */`
- `#define KEY_END       0x166   /* end key */`
- `#define KEY_PPAGE     0x153   /* previous page (PageUp) */`
- `#define KEY_NPAGE     0x152   /* next page (PageDown) */`
- `#define PADENTER      0x1cb   /* enter on keypad */`

You may check curses.h to see more definitions.

Colors

- To start using color, you should first call the function `start_color()`.
  - To find out whether a terminal has color capabilities or not, you can use `has_colors()` function, which returns FALSE if the terminal does not support color.
- Colors are always used in pairs.
  - A color-pair consists of a foreground color and a background color.
  - Initializes a color-pair with the routine `init_pair()`. After it has been initialized, `COLOR_PAIR(n)` is used to represent the color attribute.

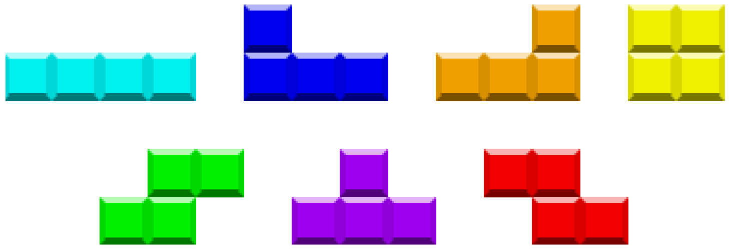
pdcurses_2.cpp

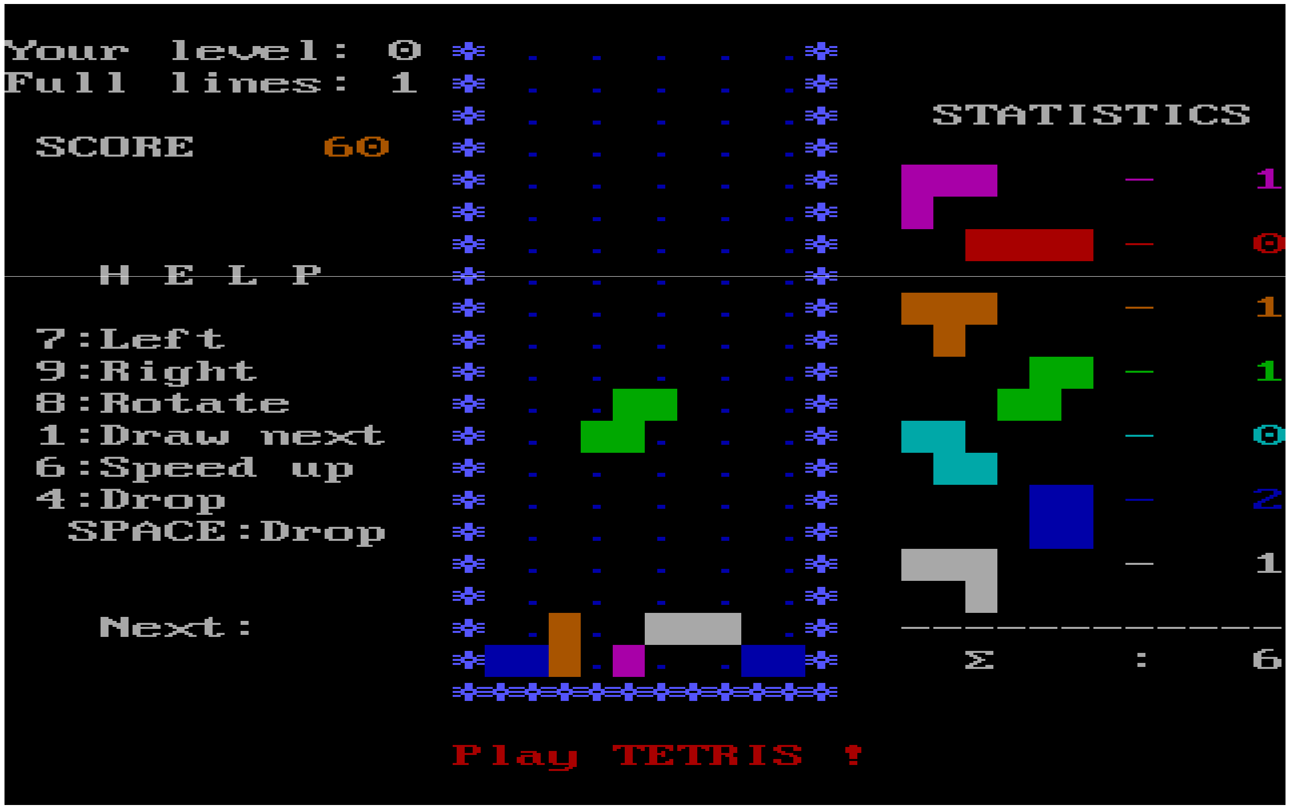
```
#include <curses.h>

int main()
{
        initscr();
        start_color();

        init_pair( 1, COLOR_WHITE, COLOR_RED );
        attron( COLOR_PAIR(1) );
        printw("Background red");
        attroff( COLOR_PAIR(1) );

        refresh();
        getch();
        endwin();
        return 0;
}
```

Curses Provides Pre-defined Colors

COLOR_BLACK	=	0
COLOR_RED	=	1
COLOR_GREEN	=	2
COLOR_YELLOW	=	3
COLOR_BLUE	=	4
COLOR_MAGENTA	=	5
COLOR_CYAN	=	6
COLOR_WHITE	=	7

HW: Tetris





```
Your  level:  0
Full  lines:  1

 SCORE      60

     H E L P

 7:Left
 9:Right
 8:Rotate
 1:Draw next
 6:Speed up
 4:Drop
  SPACE:Drop

    Next:
```

STATISTICS

```
— 1
— 0
— 1
— 1
— 0
— 2
— 1
------------
Σ    :    6
```

Play TETRIS !