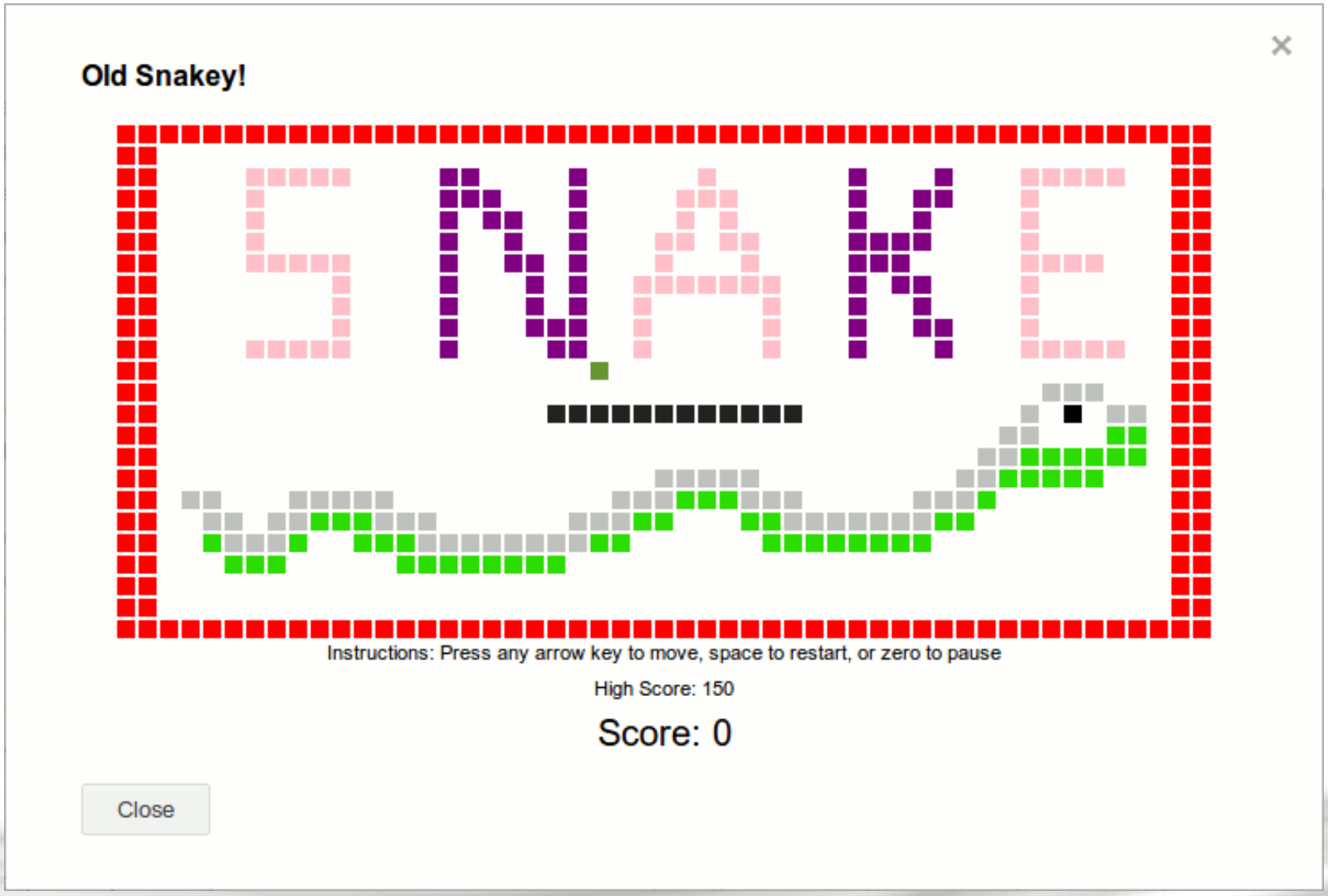
Project 1 - Snake

Meng-Hsun Tsai
CSIE, NCKU

Task: Developing a Snake Game



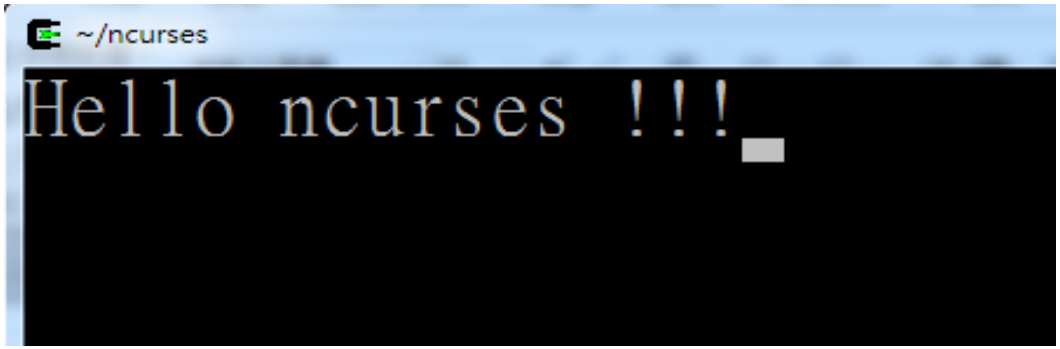
Snake Game

- Snake is a video game concept which originated during the late 1970s in arcades.
- After it became the standard pre-loaded game on Nokia mobile phones in 1998, there was a resurgence of interest in the game as it found a larger audience.
- The player controls a dot, square, or object on a bordered plane. As it moves forward, it leaves a trail behind, so that it resembles a moving snake.
- The player loses when the snake runs into either the border of the screen or the trail left by the snake.

ncurses

- ncurses is a programming library providing an API, allowing the programmer to write text user interfaces in a terminal-independent manner.
- In cygwin, remember to install package Libs->libncurses-devel

hello_ncurses.cpp




```
$ cat hello_ncurses.cpp
...
#include <ncurses.h>
int main()
{
    initscr();                  // start curses mode
    mvprintw(0 , 0, "Hello ncurses !!!");
    refresh();                  // print it on to the real screen
    getch();                    // wait for user input
    endwin();                   // end curses mode
    return 0;
}
```

```
$ g++ -o hello_ncurses hello_ncurses.cpp –lncurses
$ ./hello_ncurses
```

keypad.cpp


```
#include <ncurses.h>
int main()
{
    int width, height, go_on = TRUE;
    int c, x = 0, y = 1;
    initscr();
    cbreak();       // disable key buffering
    noecho();       // disable echoing
    keypad(stdscr, TRUE);  // enable keypad reading
    getmaxyx(stdscr, height, width);  // get screen size
    mvaddstr(0, 0, "Pressed Key: ");
    while (go_on) {
      move(y, x);
      c = getch();
      switch (c) {
        case KEY_LEFT:  --x;
          mvaddstr(0, 14, "Left ");  break;
        case KEY_RIGHT: ++x;
          mvaddstr(0, 14, "Right"); break;
        case KEY_UP:    --y;
          mvaddstr(0, 14, "Up   ");  break;
        case KEY_DOWN:  ++y;
          mvaddstr(0, 14, "Down "); break;
        default:        go_on = FALSE;
      }  // switch (c)
      while (x < 0) x += width;
      while (x >= width) x -= width;
      while (y < 0) y += height;
      while (y >= height) y -= height;
    }    // while (go_on)
    endwin();
    return 0;
}  // main( )
```

ctime library

```
#include <ctime>          // for time_t and time( )
#include <unistd.h>       // for sleep( )
int main()
{
        time_t t1, t2;
        t1 = time(NULL);   // get elapsed seconds since 1970/1/1 00:00:00
        sleep(3);              // sleep for 3 seconds
        time(&t2);             // you can also pass a pointer to time( )
        cout << "t1 = " << t1 << endl;
       cout << "t2 = " << t2 << endl;
       cout << "elapsed time = " << t2 – t1 << endl;
       return 0;
}
```

Output:
t1 = 1300007706
t2 = 1300007709
elapsed time = 3

Using *nodelay()* in ncurses

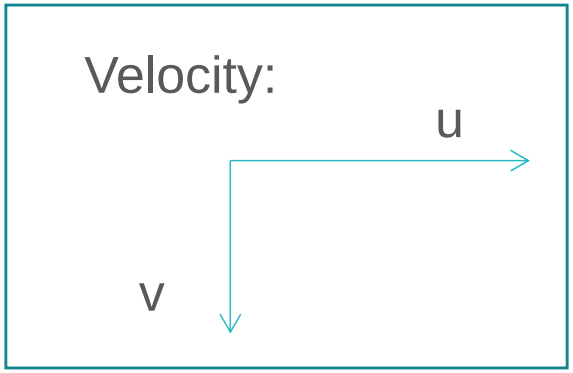
Demo with keys: R R D D R R U U U U
http://imslab.org/~tsaimh/PD2_ncurses_move_example/

```
#include <ncurses/curses.h>
int main()
{
    int width, height;    double u, v, x, y;    int c, go_on;

    initscr();    cbreak();    noecho();   keypad(stdscr, TRUE);
    nodelay(stdscr, TRUE);      getmaxyx(stdscr, height, width);
    mvaddstr(0, 0, "arrow keys: move and accelerate the cursor");
    mvaddstr(1, 0, "other keys: quit");
    u = v = 0;     x = y = 0;     go_on = 1;
    while (go_on) {
            while ((c = getch()) == ERR) {   // while nothing entered
```

Using *nodelay()* in ncurses (cont.)

```
13                  move(y, x);
14                  x += u / 1000;
15                  y += v / 1000;
16                  while (x < 0) x += width;
17                  while (x >= width) x -= width;
18                  while (y < 0) y += height;
19                  while (y >= height) y -= height;
20              } // while ((c = getch()) == ERR)
21              switch (c) {
22                  case KEY_LEFT:  --u; break;
23                  case KEY_RIGHT: ++u; break;
24                  case KEY_UP:    --v; break;
25                  case KEY_DOWN:  ++v; break;
26                  default:        go_on = 0;
27              }  // switch (c)
28      }  // while (go_on)
29      endwin();    return 0;
30  }
```



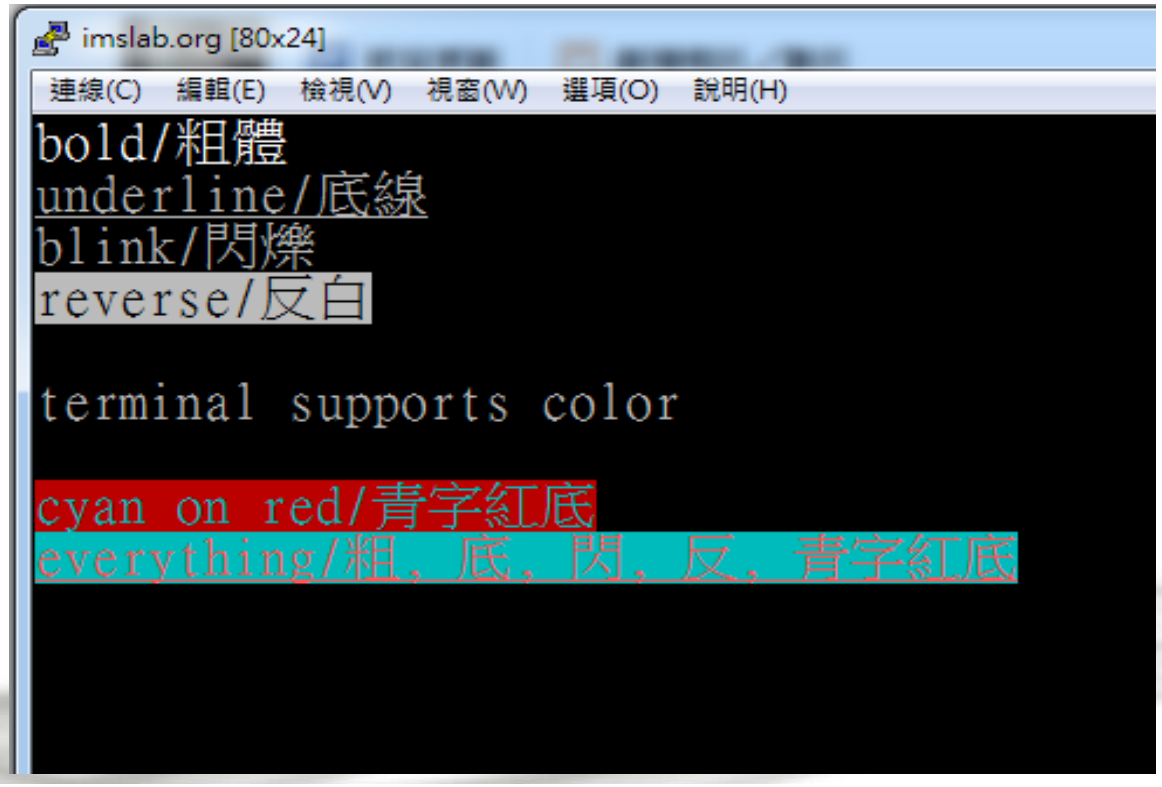
Modifying Character Attributes

```
#include <ncurses.h>
int main()
{
    initscr();
    attrset(A_BOLD);            mvaddstr(0, 0, "bold/粗體");
    attrset(A_UNDERLINE);       mvaddstr(1, 0, "underline/底線");
    attrset(A_BLINK);           mvaddstr(2, 0, "blink/閃爍");
    attrset(A_REVERSE);         mvaddstr(3, 0, "reverse/反白");
    attrset(A_NORMAL);
    if (has_colors())
            mvaddstr(5, 0, "terminal supports color");
    else
            mvaddstr(5, 0, "terminal does not support color");
```

Modifying Character Attributes (cont.)

```
14    start_color();
15    init_pair(1, COLOR_CYAN, COLOR_RED);
16    attrset(COLOR_PAIR(1));
17    mvaddstr(7, 0, "cyan on red/青字紅底");
18    attrset(A_BOLD | A_UNDERLINE | A_BLINK | A_REVERSE | COLOR_PAIR(1));
19    mvaddstr(8, 0, "everything/粗, 底, 閃, 反, 青字紅底");
20    attrset(A_NORMAL);
21
22    cbreak();
23    noecho();
24    getch();
25    endwin();
26    return 0;
27 }
```

Usually color pair 0 is assumed to be white on black. It cannot be modified by the application.



Any Other Colors?

```
$ grep COLOR_ /usr/include/ncurses.h
#define COLOR_BLACK     0
#define COLOR_RED       1
#define COLOR_GREEN     2
#define COLOR_YELLOW    3
#define COLOR_BLUE      4
#define COLOR_MAGENTA   5
#define COLOR_CYAN      6
#define COLOR_WHITE     7
```

Requirements

- Develop an nondeterministic and interactive game
  - You can use ncurses library for a text-based Unix environment or other libraries (study by yourself!) for various GUI environments.
- Write a Makefile to compile your program.
- Include at least two classes (in separate files).
- Draw UML class diagram in your report.

Requirements (cont.)

1. Display the following information on the screen:
   - Time(in second)
   - Score (design your own scoring policy)
   - The descriptions of $, $ and ?.
2. Make the map circular. When the snake crosses the border of the map, it appears on the other side. (e.g. Right side out, left side in.)
3. As time passed, the snake becomes longer and faster.
4. Create obstacles on the map.

Requirements (cont.)

5. Randomly generate three items for eating :
   - $: Good money. The snake gets some rewarded scores.
   - $: Poisoned money. It makes the snake die.
   - ?: Something **SURPRISE.** One of the following effects occurs:
     - The snake can go through the obstacles in a short period.
     - The snake becomes longer.
6. Game Over Conditions:
   - The snake hits the obstacles.
   - The snake eats the poisoned **item** $.
   - The snake touches its own body.

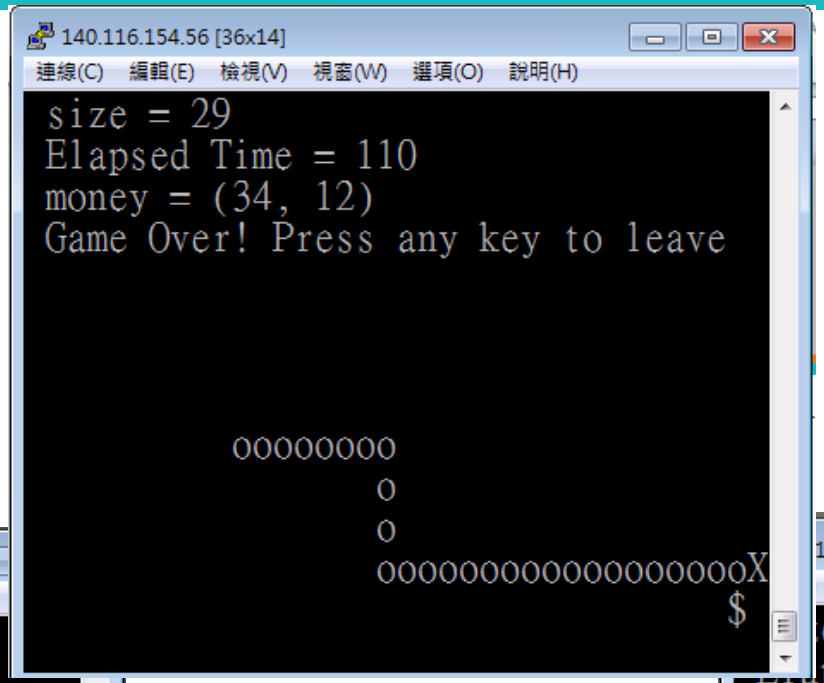
作業要求 (中文版)

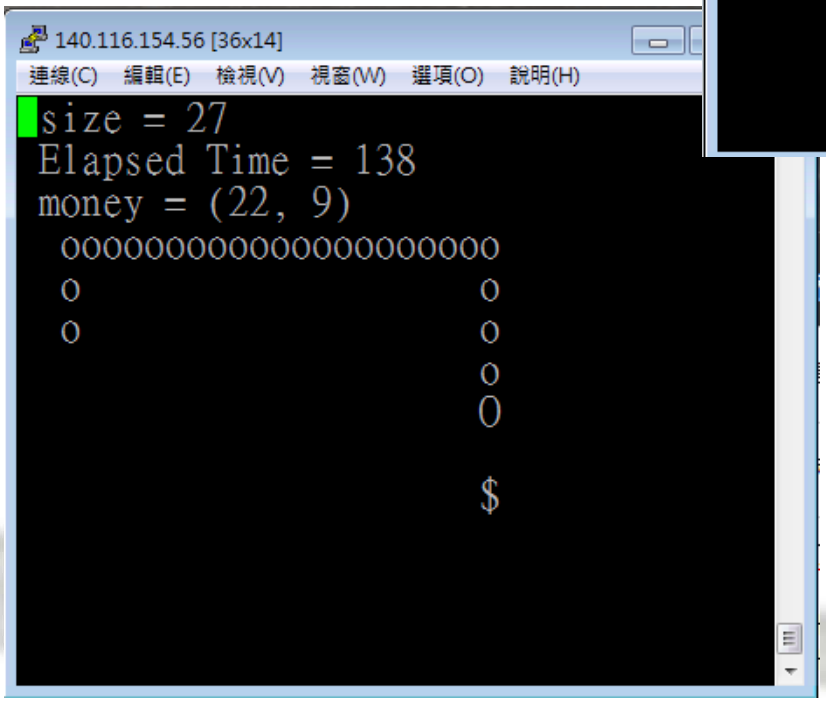
1. 螢幕上會顯示：分數，可吃東西的說明，遊戲目前經過多少秒。
2. 蛇可以穿越邊界，從右/左邊進去會從左/右邊出來，從上/下面進去會從下/上面出來。
3. 根據經過的秒數增加蛇的長度和速度。
4. 地圖中間要有牆(障礙物)，碰到**會死掉**。
5. 可以吃的東西有三種：
   - 綠色$：得到分數。
   - 紅色$：有毒吃到**會死掉**，出現一段時間後會消失。
   - (隨機)?：1. 蛇身體變長四格或五格。
     2. 無敵狀態，蛇的身體會閃爍，可以穿越牆壁，效果一段時間後消失。
6. 死亡條件：撞到牆 or 吃到紅色$ or 碰到自己，顯示Game Over。

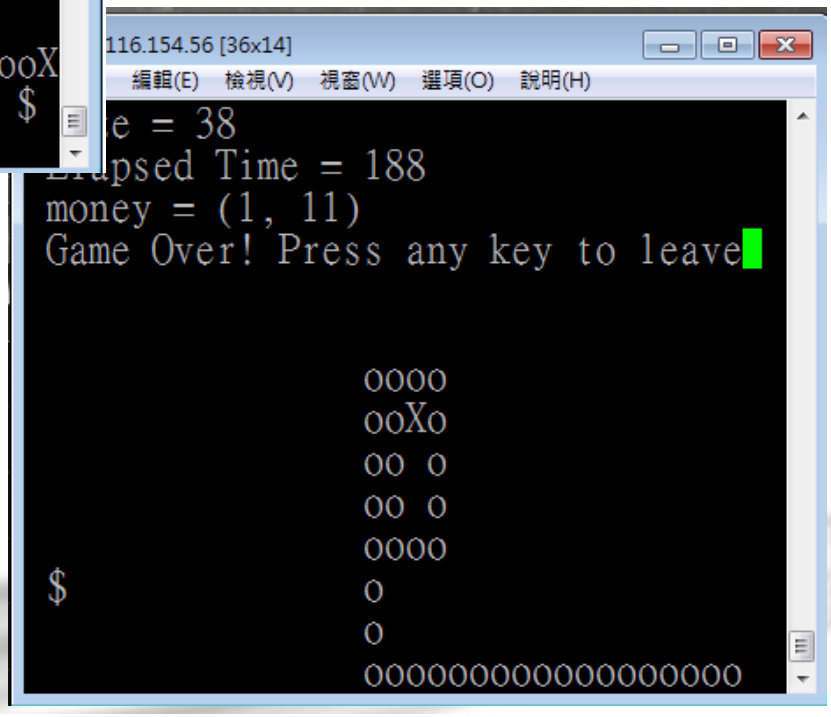
Evaluation

- Demo Date: 5/12 ~ 5/16, By appointment
- Demo Room: TBA
- You should upload your source code and report to Moodle before 5/11 11:00pm.
- Grading Policy
  - Report (1 ~ 5 pages) 10%
  - Demo to TA 40%
  - On-site modification 50%
  - Bonus up to 20%

Simple Example







Reference

- Snake Game Introduction
  http://en.wikipedia.org/wiki/Snake_(video_game)
- ncurses HOWTO
  http://tldp.org/HOWTO/NCURSES-Programming-HOWTO/helloworld.html
- Writing Programs with NCURSES
  http://invisible-island.net/ncurses/ncurses-intro.html
- TRIBUTE TO TEXT-MODE GAMES
  http://www.textmodegames.com/