ESTR3106 Principles of Programming Languages

Project Overview and Ncurses Library

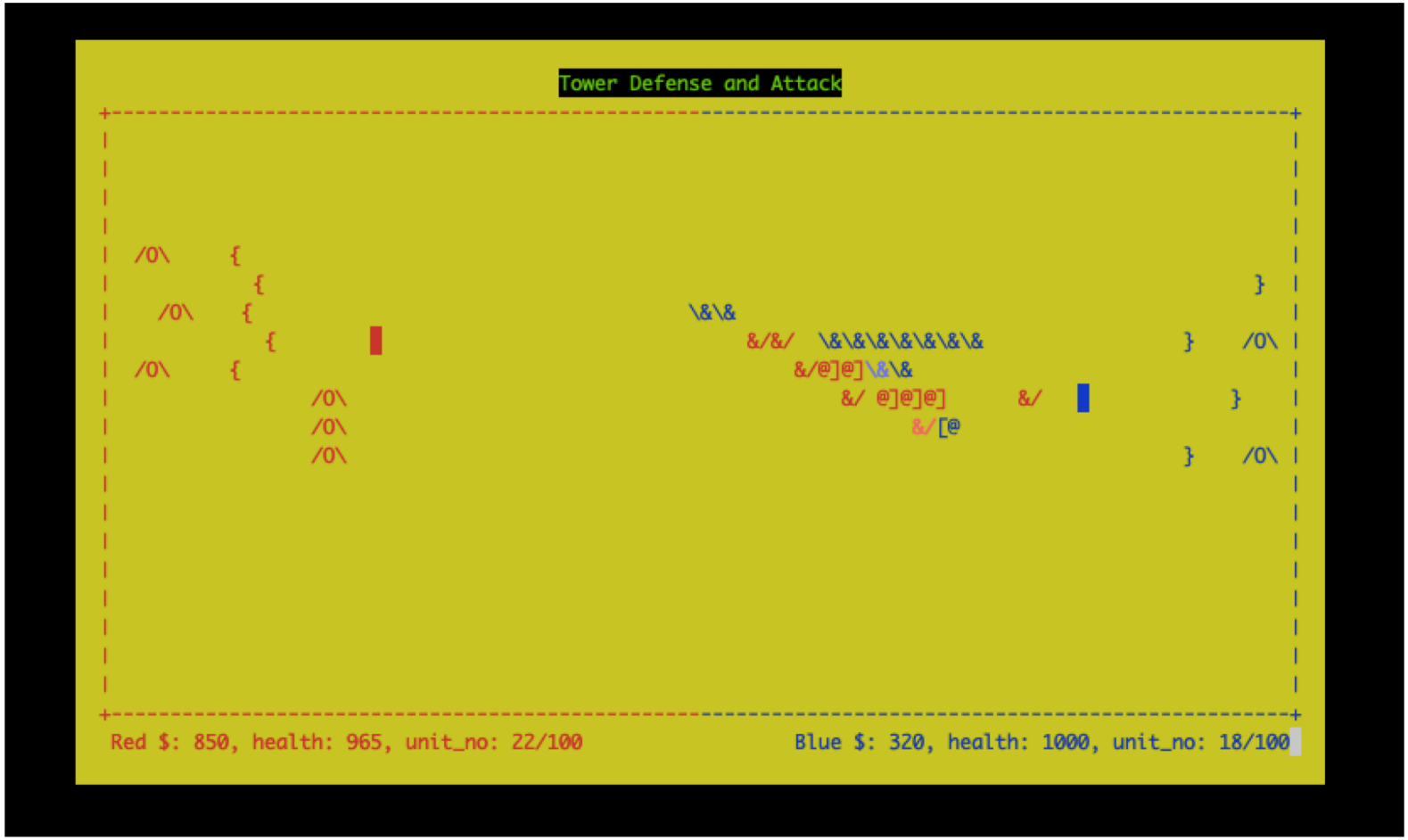
Tutorial 1

Project Overview

Project Overview

- *Tower Defense and Attack*
  - Two-player real-time strategy game
  - Concurrent programming exercise (Administrator-and-Worker)
- Implementation: the C language
  - GNU/Linux with the GCC compiler
  - User Interface: Ncurses Library
  - Concurrent programming: Synchronous Interprocess Messaging Project for LINUX (SIMPL)

Tower Defense and Attack



Tower Defense and Attack

- Two human control players, RED and BLUE, competing on a 20 by 100 arena.
- Each player needs to
  - manage resources
  - use resources to build units (robots/buildings)
  - strategically send robots to attack the opponent

Player

- Basic properties:
  - Health Points (HP)
  - Resources
  - Unit Number
  - Baseline
- Controls:
  - Move the marker
  - Place units onto the arena
  - Remove buildings from the arena

Player

- Initially, 500 resources, 1000 health points, and no units on the arena.
- Placing units costs resources
- Place units on left/right half of the arena
- If the baseline is reached by robots from the opponent, the player's HP is reduced
- A player loses immediately if his HP is reduced to zero, and its opponents wins the game

Unit

- Basic properties:
  - Type
  - Health Points (HP)
  - Movement Interval
  - Position
- Types:
  - Buildings
    - Mine generates resources
    - Wall blocks robots
  - Robots
    - Lancer / Hoplite attacks enemies or reduces the opponent's HP

More on Units

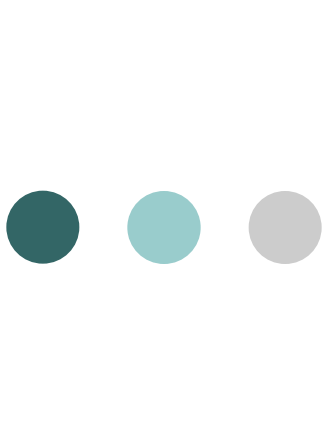
- All units should be within the arena.
- No units can overlap each other.
- Take actions (move forward, attack or generate resource) automatically for every movement interval.
- Removed from the arena if its HP is reduced to zero.

More on Units

- Buildings:
  - Mine generates 10 resources every second
  - Wall can block robots
- Buildings cannot move
- Buildings can be removed by the player and half of its costs will be returned.

More on Units

- Robots:
  - Move forward or attack for every movement interval
  - The opponent's HP is reduced if a robot reaches the baseline.
  - If blocked by a unit in the same team, the robot will wait
  - If blocked by a unit in the opposite team, the robot will attack (cause damage to) that unit.



Ncurses Library

Ncurses (new curses) Library

- Free software emulation of the original curses library in UNIX
- Provide an efficient API to terminal IO operations: move cursor, create windows, produce colors, etc.
- Need not worry about the underlying terminal capabilities
- Header file: `<curses.h>`
- To compile:

```
gcc prog.c -o prog -lcurses
```

Initialization and Termination

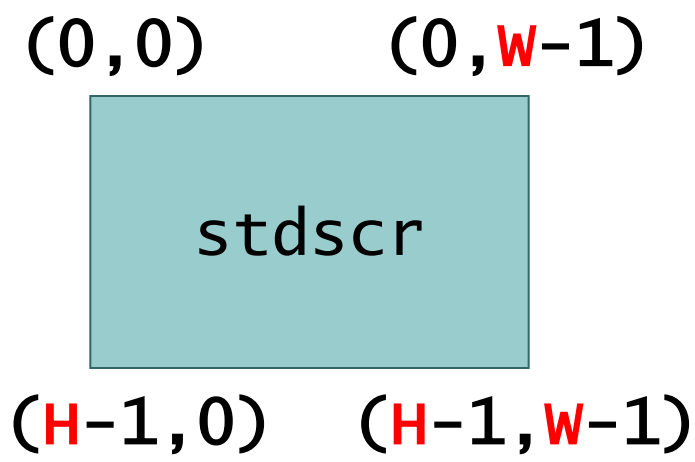
```
#include <curses.h>
int main() {
    initscr();    /*start curses mode*/
    cbreak();     /*disable input buffering*/
    noecho();     /*disable echoing input*/

        /* ... screen handling ... */

    endwin();     /*end curses mode*/
    return 0;
}
```

Window – The Imaginary Screen

- 2D arrays of characters representing all or parts of screen
- I/O should pertain to specific window
- Coordinates: (y, x)
- Default window: `stdscr`
  - To get the dimension:
    ```
    int H,W;
    getmaxyx(stdscr,H,W);
    ```
  - Changes to window are not reflected until:
    `refresh();` or `wrefresh(stdscr);`



Moving Cursor and Output

- Example: say “Hi” to a user in (5,10) of `stdscr`

```
char* user = “Bob”; int y = 5, x = 10;
move(y,x); printw(“Hi %s”,user);
```

- Syntactic sugar:

```
mvprintw(y,x,“Hi %s”,user);
```

- Other output functions:
  - Print a character: `addch` and `mvaddch`
- Remember to `refresh` to see the changes!

Input

- To read a character from `stdscr`:

  ```
  int ch = getch();
  ```

- To capture special keys such as arrow keys:

  ```
  keypad(stdscr,TRUE);
  ```

- Definitions of special keys' integer value:

  `KEY_LEFT`, `KEY_RIGHT`, `KEY_UP`, etc...

- To disable `getch` of waiting for a key hit:

  ```
  nodelay(stdscr,TRUE);
  ```

  - Then, `getch` return `ERR` if no key is hit.

Other Topics

- Drawing border
- Clear the screen
- Create and destroy new windows
- Colors Handlings
- Etc…
- Read from the online resources

http://tldp.org/HOWTO/NCURSES-Programming-HOWTO/

Tips

- Drawing the marker
  - The following attribute can be useful in writing the character ▌ for the cursor :

    ```
    A_REVERSE        Reverse video
    A_BLINK          Blinking
    ```

  - To turn on an attribute:

    ```
    attron(A_REVERSE);
    ```

  - To turn off an attribute:

    ```
    attroff(A_REVERSE);
    ```

- More details can be found on:

http://tldp.org/HOWTO/NCURSES-Programming-HOWTO/attrib.html

Homework

- Download the sample implementation from the course web page and play around
  - Note: sample programs can only be run on linux6 – linux9
  - **Programs can only be run** on Linux with **Putty** or **terminal in Mac OS**
- Read the assignment specification
- Ask questions on the forum

Running the Sample Programs

- Download the package from the course webpage
- Follow the instruction on the **README file**
- Important points:
  - Create a new directory where the FIFOs to live e.g. `mkdir $HOME/fifo`
  - Set up the `FIFO_PATH` and `SIMPL_HOME` environment variables
  - KILL all your useless processes using ./flush
  - Reset your screen using command reset
  - TIPS: If you got strange problem, make sure the directory pointed by `FIFO_PATH` is clean

END