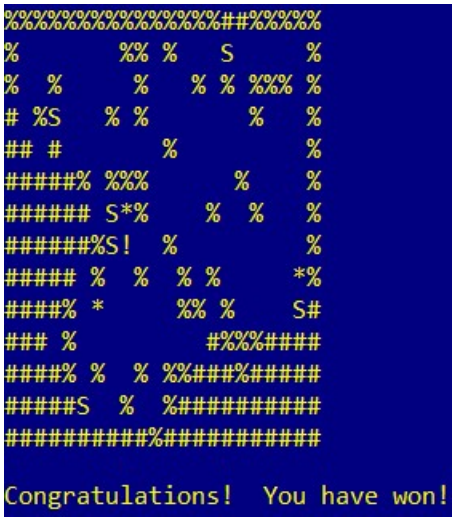
**ECE220: Computer Systems & Programming** **Fall 2018 ZJUI**
**Lab 6** **Tuesday 23 October / Thursday 25 October**

# Cave Explorer: Event-Driven Function Execution




In today's lab, your group must write C functions to move an intrepid explorer around in a cave. A screenshot of the game is shown to the right. The goal of this exercise is to give you some experience writing functions that make use of and are used by other functions. Such function composition skills are needed for MP5.

Begin by checking out the **`lab6`** subdirectory in your Subversion repository. The directory contains a copy of this document and a starting point for your solution in the **`lab6.c`** file. The directory also contains a header file, **`lab6.h`**, which contains a description of the functions that your functions must use (mentioned below), a separate source file, **`main.c`**, which implements most of the game logic, and a **`Makefile`** used to compile your code (see below).

## The Task

You must write five functions in the file **`lab6.c`**. Four of these functions move the player in one direction, and are called whenever the human pushes the corresponding key.

Build your program by typing "**`make`**" (no quotes) in the **`lab6`** subdirectory, then type "**`./cave`**" to execute the game. Until you write your functions, the player will not be able to move, so press Q to quit once you've seen the interface.

The cave (the game world) is **`WORLD_HEIGHT`** spaces high (coordinate values from **`0`** to **`WORLD_HEIGHT - 1`**) and **`WORLD_WIDTH`** spaces wide (coordinate values from **`0`** to **`WORLD_WIDTH - 1`**). The player's position (**`px`**,**`py`**) is available as file-scope variables, as is the player's current amount of gold, **`pgold`**.

You should start by writing the function **`player_move_right`**. The function tries to move the player right and returns a value indicating what happened (see **`lab6.c`** for the meanings of return values). First, it must check whether moving right might move the player out of the world (beyond coordinate **`WORLD_WIDTH - 1)`**. If so, the function must return 0. Otherwise, the function must check what is in the world at coordinates (**`px+1`**,**`py`**) by calling this function

```
int32_t world_has (int32_t xpos, int32_t ypos);
```

If the result is a wall (**`WORLD_WALL`**), the function must also return 0. Finally, the function should execute the move by incrementing **`px`**, then return the value of a call to

```
int32_t check_move (void);
```

which checks for gold and snakes, and which you must write later.

Once you finish **`player_move_right`**, you can test the function by again typing "**`make`**" (no quotes) in the **`lab6`** subdirectory, typing "**`./cave`**" to execute the game, and pressing D to move right until you a wall. (You can make the world smaller by editing **`lab6.h`** in order to test the world boundary check, but the world has to have enough space for gold and snakes.)

Next, implement the movement functions for the other three directions:

```
int32_t player_move_left (void);

int32_t player_move_up (void);

int32_t player_move_down (void);
```

Once movement is working, you can write the **check_move** function. Start by checking whether the player has moved into gold (**world_has** returns **WORLD_GOLD**). If so, collect the gold by setting the world contents at that position to **WORLD_EMPTY** using

```
int32_t world_set (int32_t xpos, int32_t ypos, int32_t item_type);
```

then increment the player's gold (**pgold**), and, if they have collected 5 gold, return 2 to allow them to win.

If not, return 1 to indicate a successful move (this return code is already in the function for you).

You may want to play again at this time. Snakes won't bother you … yet.

Finally, use the **world_has** function to check whether the player has moved adjacent to a snake (**WORLD_SNAKE**). Be sure to check that the player is not at a world boundary before calling **world_has**. If the player steps next to a snake, return 3 to indicate that they have been bitten.

That's it! Have fun playing the game!