CS 202 Homework 4

Bryan Beus

March 6, 2020

Source Code Link: `https://github.com/siddhartha-crypto/cs202/tree/master/hw4`

1 Design

1.1 Main - Hunt the Wumpus

The Spelunking Assignment prepared the foundation for this assignment. I expect to be able to utilize the room structure from that assignment for the cave in this assignment.

Aside from the room structure, everything else does not appear to be overly challenging. I expect to use the random device discussed in class to add some entropy to the room selection. The wumpus is at the back of the cave, the bats and pits towards the middle.

I'll give each type of obstacle its own class to make management easier.

1.2 Speed Coding

This is an intriguing assignment, and I hear that these concepts are utilized on the test.

I would like to do these types of things on a regular basis someday anyway – memorizing many useful tricks to have a "logical vocabulary."

The assignments are all simple, so there is no need to go into greater detail about the design.

2 Post Mortem

2.1 Main

Seeing a game come to a higher level of completion than before is an enjoyable experience. With the room structure from the Spelunking assignment, this project was relatively simple.

There were no notable challenges to this assignment, although on the whole it did take several hours.

2.2 Speed Coding

I was able to complete two rounds of each type of test of the three. For all but one of the exercises (of six), I completed the work in less than twenty minutes. The only one that I needed help on was the first pass through memory coding test. I could not remember how to properly add data to a map. Everything else was fairly straightforward.

3 Commit History

3.1 Main

2020-03-03 initiate hw4
2020-03-03 initiate bats and pit classes
2020-03-03 adjusting file structure in hw4
2020-03-03 rename wump dir to main in hw4
2020-03-03 initiate new classes for wumpus, bat, and pit
2020-03-03 debug simplified pass in hw4 main
2020-03-03 convert user input to appropriate format for this game in hw4
2020-03-04 develop hunt the wumpus in hw4

2020-03-04 rename wumpus getroom()
2020-03-04 remove bats implemented in hw4 in wumpus
2020-03-04 implement wumpus death and bat removal in hunt the wumpus in hw4
2020-03-04 add arrows to wumpus in hw4
2020-03-04 update miscellaneous.cpp in hw4 main
2020-03-04 fine tune and debug wumpus in hw4
2020-03-04 tune and debug additional files in wumpus hunt in hw4
2020-03-05 simplify console readout in main hw4

3.2 Speed

2020-03-04 initiate speed hw4
2020-03-04 add first speed assignment
2020-03-04 initiate containers in hw4
2020-03-04 container aspect (first pass) of speed in hw4
2020-03-04 initiate memory of speed in hw4
2020-03-05 create memory portion of speed in hw4
2020-03-05 two more iterations of speed tests
2020-03-05 update containers2
2020-03-05 test stream2
2020-03-05 develop containers2 in speed in hw4
2020-03-05 develop memory comments in speed in hw4
2020-03-05 develop speed in hw4

4 Answers to Questions

(No questions listed)

5 Sample Output

5.1 Main - Win Scenario

```
Current Room: 16
I smell a wumpus

Adjacent Rooms:
Choice A) 15

Choice B) 17

Choice C) 18


Make a choice for the next room to visit (A, B, C).
Shoot an arrow by entering (S).
Enter (Q) to quit.
S

Arrows away!


Press enter to continue...
Kapow!
The wumpus is dead.
Dang it. And I wanted to destroy you.
Care to... try again?
```

5.2 Main - Lose Scenario

```
GARMPHHH
Tasty...
```

5.3 Speed Coding - Containers

```
mydata
found it
0
```

5.4 Speed Coding - Memory

```
mydata
otherdata
```

5.5 Speed Coding - Memory

```
1 2 3 4 5 6 7 8 9 10
```

6 My Programs

6.1 Main

```
/*
 * main.cpp
 * CS 202
 * March 3, 2020
 * Bryan Beus
 * Main file for Hunt the Wumpus project
 */

#include <iostream>
#include <iomanip>
#include <string>
#include <vector>
#include <list>
#include <iterator>
#include <memory>
#include <fstream>
#include <sstream>
#include <random>
#include <cmath>
#include <stdlib.h>
#include <algorithm>
```

```cpp

#include "Miscellaneous.hpp"
#include "Cave.hpp"
#include "Wumpus.hpp"
#include "Bats.hpp"
#include "Pit.hpp"

using std::cout;
using std::cin;
using std::endl;
using std::vector;
using std::string;
using std::list;
using std::right;
using std::ifstream;
using std::ofstream;
using std::istream;
using std::getline;
using std::istringstream;
using std::random_device;
using std::seed_seq;
using std::mt19937;
using std::random_shuffle;

int main(int argc, char* argv[])
{
    // Inform user of the nature of the software
    clearConsole();
    cout << "Welcome to Hunt the Wumpus" << endl;
    cout << "Initiate User Destruction" << endl;
    waitForContinue();

  // Create pseudo-random device
  random_device r;
  seed_seq seedObj{r(), r(), r(), r(), r(), r(), r(), r()};
  mt19937 e1(seedObj);

    // Max room
    int max_room = 18;

    // Initiate creature objects
    Wumpus wumpus(e1, max_room);
    Pit pit(e1, max_room);
    Bats bats(e1,max_room);

    // Arm user with arrows
    int arrows = 5;

    // Create initial environment
    Cave cave;

    // Create a string that holds a default cave
    string def_cave = cave.createDefaultCave(max_room);

    // Read in the default cave
    istringstream default_cave(def_cave);
    cave.readRooms(default_cave, max_room);


    // Initiate user input while loop
    vector<string> choiceList{"A", "B", "C", "S", "X"};
    while (true) {
        clearConsole();

        // Discover current room
        int currentRoom = cave.getCurrentRoom();

        // If the wumpus is awake, move him to the next room and
        ↪  set him to go back to sleep
        if (!wumpus.getStatus()) {
            wumpus.moveToAdjacentRoom(pit, max_room, e1);
            wumpus.switchStatus();
        }

        // Check for player and enemy/obstacle collisions
        if (currentRoom == wumpus.getRoom()) {
            cout << "GARMPHHH" << endl;
            cout << "Tasty..." << endl;
            break;

        // Check for the pit
        } else if (currentRoom == pit.getRoom()) {
            cout << "AAARRRGHGGHHHHHhhhhhhhhrrmmm..........." <<
            ↪  endl;
            cout << "... *splat*" << endl;
            break;

        // Check for bats
        } else if (currentRoom == bats.getRoom()) {

            cout << "Wheeeeee!" << endl;
            waitForContinue();

            // Move to a random room
            mt19937 *_e1 = nullptr;
            _e1 = &e1;
            int randomRoom = chooseRandomRoom(_e1, 0, max_room -
            ↪  1);
            cave.gotoRoom(randomRoom);

            // Restart loop
            continue;
        }

        cout << "Current Room: " << currentRoom + 1 << endl;

        // Discover adjacent rooms
        vector<int> adjacent_rooms =
         ↪  cave.getAdjacentRooms(currentRoom);

        // Discover potential dangers
        vector<string> warnings;
        for (int i = 0; i < 3; i++) {
            if (adjacent_rooms.at(i) == wumpus.getRoom()) {
                warnings.push_back("I smell a wumpus");
            }

            if (adjacent_rooms.at(i) == bats.getRoom()) {
                warnings.push_back("I hear a bat");
            }

            if (adjacent_rooms.at(i) == pit.getRoom()) {
                warnings.push_back("I feel a breeze");
            }
        }

        // Scramble warnings
        random_shuffle(warnings.begin(), warnings.end());
        for (size_t i = 0; i < warnings.size(); i++) {
            cout << warnings.at(i) << endl;
        }

        cout << endl;

        cout << "Adjacent Rooms: " << endl;

        // Present user with description of adjacent rooms and
         ↪  choices
        for (int i = 0; i < 3; i++) {
            cout << "Choice " << choiceList.at(i) << ") ";
            cave.printShortDescription(adjacent_rooms.at(i));
            cout << endl;
        }

        // Capture user input
        int userInput;
        capture_user_input(userInput);

        // If indicated, shoot arrows or quit
        if (userInput == 3) {

            // Check that the user has arrows remaining
            if (arrows > 0) {
                arrows--;
            } else {
                cout << "Out of arrows" << endl;
                continue;
            }

```

```
                cout << "Arrows away!" << endl;
                cout << endl;

                waitForContinue();

                // Check to see if the arrows struck any obstacle
                for (size_t i = 0; i < warnings.size(); i++) {
                    if (warnings.at(i) == "I feel a breeze") {
                        bats.removeBats();
                    }

                    if (warnings.at(i) == "I smell a wumpus") {
                        cout << "Kapow!" << endl;
                        cout << "The wumpus is dead." << endl;
                        cout << "Dang it. And I wanted to destroy
                        ↪   you." << endl;
                        cout << "Care to... try again?" << endl;
                        exit(0);
                    }

                }

                // If the wumpus is still alive, he is awake after
                ↪   hearing the arrows
                wumpus.switchStatus();

            // User quit option
            } else if (userInput == 4) {
                cout << "Weakling" << endl;
                break;
            } else {

            // Proceed to adjacent room
            cave.gotoAdjacentRoom(adjacent_rooms.at(userInput));
            }
        }

    return 0;
}
```

6.2 Speed Coding - Containers

```
/*
 * main.cpp
 * CS 202
 * March 3, 2020
 * Bryan Beus
 * Main file
 */

```

```cpp
#include <string>
#include <iostream>
#include <iomanip>
#include <map>

#include "MyClass.hpp"

using std::cin;
using std::cout;
using std::endl;
using std::string;
using std::map;

int main() {

    // Declare new object
    MyClass a;

    // Declare data to store
    string myData = "mydata";

    // Create default data
    a.addData(myData);

    // Retrieve data
    string myReturnedData = a.getData(0);

    // Print data here
    for (auto s: myReturnedData) {
        cout << s;
    }
    cout << endl;

    // Find position
    int pos = a.findData(myData);
    cout << pos << endl;

    return 0;

}
```

6.3 Speed Coding - Memory

```cpp
/*
 * main.cpp
 * CS 202
 * March 3, 2020
 * Bryan Beus
 * Main file
 */

```

```cpp
#include <string>
#include <iostream>
#include <iomanip>
#include <map>
#include <memory>

#include "MyClass.hpp"

using std::cin;
using std::cout;
using std::endl;
using std::string;
using std::map;

int main() {

    // Declare new MyClass instance
    MyClass a;

    // Add data
    string myData = "mydata";
    a.alterData(myData);

    // Retrieve data
    string d = a.retrieveData();
    cout << d << endl;

    // Test that data is loadable and alterable
    string otherData = "otherdata";

    a.alterData(otherData);
    string b = a.retrieveData();
    cout << b << endl;


    return 0;

}
```

6.4 Speed Coding - Streams

```cpp
/*
 * main.cpp
 * CS 202
 * March 3, 2020
 * Bryan Beus
 * Main file
 */

#include "MyClass.hpp"

```

```cpp
int main() {

    // Declare new object
    MyClass a;

    // Create default data
    a.createData();

    // Save default data to file
    a.saveData();

    // Declare new object
    MyClass b;

    // Load default data from object a
    b.loadData();

    // Report data
    b.printData();

    return 0;

}
```