CSCI 120 Intro to CS II

Project 1

Crafton Hills College
Total Points: 50

Overview

In this project, you will apply object-oriented design principles to enhance a two-dimensional adventure game. In this game, the user controls the movement of a player character using the arrow keys. Currently, two types of game entities beside the player are available: wall and prize. The player is not allowed to occupy the location of a wall entity. When a player moves into the location of a prize entity, the prize entity disappears, and a sound is played. You will be adding functionality to this game, such as keep track of score, adding new game entities, ending the game when the user reaches the destination. The message board and score board towards the bottom of the window displays a message spoken by an entity and the current score, respectively.

The program is implemented in C++ using SFML (Simple and Fast Multimedia Library). For more information about SFML, see http://www.sfml-dev.org/. API is available online at http://www.sfml-dev.org/documentation/2.4.0/. Read the following tutorials to learn about the basics of SFML:

- Handling time: http://www.sfml-dev.org/tutorials/2.4/system-time.php
- Opening and managing a SFML window: http://www.sfml-dev.org/tutorials/2.4/window-window.php
- Events explained: http://www.sfml-dev.org/tutorials/2.4/window-events.php
- Keyboard, mouse and joystick: http://www.sfml-dev.org/tutorials/2.4/window-inputs.php
- Drawing 2D stuff: http://www.sfml-dev.org/tutorials/2.4/graphics-draw.php
- Sprites and textures http://www.sfml-dev.org/tutorials/2.4/graphics-sprite.php
- Text and fonts: http://www.sfml-dev.org/tutorials/2.4/graphics-text.php
- Playing sounds and music: http://www.sfml-dev.org/tutorials/2.4/audio-sounds.php

DETAILED PROJECT REQUIREMENTS

1. Object-oriented design principles, including data encapsulation and information hiding, must be used in this project.
2. Modify the **Entity** class:
    - Add data member so that each entity has a sf::Sound or sf::SoundBuffer variable accessing the sound of the entity.
    - Add a virtual function **getScoreUpdate(int timeDelay)** that returns 0. This function will be overridden in the derived classes to provide different scoring functions for different types of entities.
    - Add a virtual function **say()** that returns a string representing a message spoken by an entity.
    - Add any other member functions and variables as needed.
3. Design and implement
    - a class **Prize** that derives from **Entity**
        - Data member should include (but are not limited to) **prizeLevel**, which is an integer between 1 and 10 representing the worth of the prize.
        - Override **getScoreUpdate(int timeDelay)** to return a positive score update, which should be directly proportional to **prizeLevel** but inversely proportional to **timeDelay** (i.e. the higher the prize level, the higher the score update; the higher the time delay, the lower the score update).
        - Override **say()** to play the sound of the entity and return a string expressing what prize the entity represents (e.g. “Magic potion increases your energy to win!”).
    - a class **Trap** that derives from Entity
        - Data member should include (but are not limited to) **penaltyLevel**, which is an integer between 1 and 10 representing the level of penalty.
        - Override **getScoreUpdate(int timeDelay)** to return a negative score update, which has a magnitude of 10 times **penaltyLevel**.
        - Override **say()** to play the sound of the entity and return a string expressing the trap that the entity represents (e.g. “Falling down a deep abyss causes you bodily damage.”).
4. Place all the classes in separates files (.h and .cpp) from the main program.
5. Modify the main program such that:
    - It creates and shows the following entities at random locations on the adventure space, *making sure that they do not overlap with each other*:
        - Five prize entities (of class **Prize**)
        - 20 wall entities
        - Five trap entities (of class **Trap**)
        - One destination entity

    - The game ends when the player reaches the destination entity. A message should be displayed on the message board, and the user can no longer moves the player.
    - The message board is updated accordingly when the player collides with other game entities.
    - The score board is updated accordingly when the player collides with prize and trap entities.
6. Polymorphism is used because the 2-D array of **Entity** pointers are used to store different game entities. You must use this array in the main program (do not remove it or change its type).
7. Code should be well-documented, including all classes and class members.
8. All source code and executable must be submitted properly on Blackboard. (IMPORTANT: if I am not able to read your source files and/or run your executable, your project will not be graded. It is your responsibility to make sure all files are submitted correctly before the deadline.)

The **resources** folder in the provided zip file contains various images and sound clips that you may use. However, you are welcome to use other resources as appropriate. You may change the colors and sprites of the game interface as you like. If you like to further enhance the game, you may do so but make sure that all the project requirements are still met.

WHAT YOU NEED TO SUBMIT FOR THE PROJECT:

- A report containing the following items:
    - Your name
    - UML class diagram
        - All classes (other than SFML classes) used should be included in the diagram. You should show all members in each class in the diagram.
    - Sample screenshots: use the [Alt+Prt Scr] buttons on your keyboard to capture a screenshot of your program running with representative inputs
    - A brief discussion of your project experience
        - Did you enjoy this project? What problems did you encounter?
        - What did you get out from the project?
        - How did you find the project (too easy, easy, just right, difficult, too difficult)?
        - What type of help/references did you use in your project (e.g. book, web sites, classmates, tutors)?

GRADING CRITERIA

- Satisfaction of project requirements (40 points)
- Report (7 points)
- Coding style (3 points)

- Refer to the *C++ Language Coding Guidelines* available at http://horstmann.com/bigcpp/BigC2_CodingGuidelines.html
- Provide sufficient documentation in the source code.
- You must use the program template provided on Blackboard that has a program comment block containing filename, description, author, class, and date.
- Use descriptive identifiers.
- Use proper spacing and indentation (refer to the textbook's program style).