Manju Muralidharan Priya

CS210 Data Structures Fall 2024

Programming Assignment 2

Text-Based RPG Using Decision Trees

Overview:

In this assignment, you will build a text-based RPG game using a **decision tree** to drive the storyline. Each decision will lead the player through different game paths and outcomes. The decision tree will be represented as a **binary tree**, where each event can lead to two possible outcomes, except for one special case where multiple events can converge to the same outcome.

Additionally, the story data for the game must be imported from a **text file**. Each line in the file will contain event descriptions, event numbers, and pointers to two outcomes (left and right), separated by a **delimiter** of your choice (e.g., `|` or `,`). Using the imported data, you will build a decision tree that models the flow of the game.

Requirements:

1. **Node Class Template**
   - Create a templated `Node<T>` class where `T` is the `Story` class.
   - The `Node` class should have pointers to its **left child** and **right child**.
2. **Story Class**
   - Define a `Story` class containing:
     - `string description`: The text representing what happens at this point in the game.
     - `int eventNumber`: A unique identifier for each event.
     - `int leftEventNumber`: The event number of the next event if the player chooses the left path.
     - `int rightEventNumber`: The event number of the next event if the player chooses the right path.
3. **GameDecisionTree Class Template**
   - Create a templated `GameDecisionTree<T>` class where `T` is the `Story` class.
   - The `GameDecisionTree` class should have methods for:
     - Loading the story data from a file and creating the tree.
     - Traversing the decision tree based on player input.

4. **Binary Tree Structure**
    - Use a **binary decision tree** structure to represent game events. Each node will have two possible outcomes (left and right child).
    - Implement one exception where multiple events may lead to the same outcome (i.e., shared child nodes).
5. **Text-Based RPG with External File Input**
    - The **game story** must be loaded from a **text file**. Each line in the file should contain the event description, event number, left child event number, and right child event number, separated by a delimiter.
    - The game will start by loading these events into a binary tree, allowing the player to make decisions and follow paths through the tree based on their choices.
6. **Main Function:** The main function of the program should call:
    - The load file function to load the story into the decision tree.
    - The play game function to begin the game.

Story File Specification:

1. **Text File Format**:

Each line in the file represents an event in the game, formatted as follows:
Copy code
`eventNumber | description | leftChildEventNumber | rightChildEventNumber`

A `-1` indicates that there is no further path (i.e., it's a leaf node).

Example file is included in the zip file attached.

2. **File Loading**:
    - Implement a function in `GameDecisionTree.h` that reads the file, parses each line based on the delimiter, and stores the events in `Story` objects.
    - Construct the binary tree by linking the nodes according to the left and right event numbers.
    - The skeletal function is provided in the zip file

Using the Makefile:

For macOS:

1. Open the **Terminal**.
2. Navigate to the project directory containing the `Makefile`.

3. Run the command: `make`
4. Execute the program with: `./main`

For Windows with MinGW:

1. Open **Command Prompt**
2. Navigate to the project directory containing the `Makefile`.
3. Run the command: `mingw32-make`
4. Execute the program with: `main.exe`

Submission Guidelines:

Folder Structure:

You are required to create a well-organized folder structure for your text-based RPG project. The folder should include all necessary files to run the project seamlessly. Below is the required structure:

```
LastName_FirstNameRPG/

├── CMakeLists.txt        # CLion project file for CMake (optional)

├── Makefile              # Makefile for command line compilation

├── main.cpp              # Main function file

├── GameDecisionTree.h    # Header file for GameDecisionTree class

├── Node.h                # Header file for Node class

├── Story.h               # Header file for Story class

└── story.txt             # Text file containing the game story
```

Steps for Submission:

1. **Create the Folder Structure:**
    - Organize your project files as shown above. Ensure that each file is correctly placed in the `LastName_FirstNameRPG` folder.

2. **Test Your Project:**
    - Compile and run your project to ensure that all functionalities are working correctly and that there are no errors.
    - Especially test that your Makefiles are running without errors and that you can run your files using the terminal.
3. **Compress the Folder:**
    - Once your project is complete and tested, compress the entire `LastName_FirstNameRPG` folder into a `.zip` file.
4. **Submit on Canvas:**
    - Upload the `.zip` file containing your project folder.

Constraint:

It is important to note that **none of the existing provided code should be modified.** This includes all class definitions and the `Makefile`, which are designed to work together seamlessly.

However, you are encouraged to write any additional functions that you deem necessary to enhance the gameplay experience or improve the organization of your code. Feel free to expand upon the functionality of your RPG by adding methods, utility functions, or features that align with your vision for the game.

If you have any questions or require assistance during the project, don't hesitate to reach out for help. Good luck, and have fun creating your text-based RPG!