COMP 188 Computer Science Senior Project Fables of Sylvus© Monster Minder

Ryan Nish, Marissa Melen, Victor Aw

2013-10-02

1 Contributions

Features list - All team members contributed equally

Contents

2 Customer Statement of requirements

Fables of Sylvus© is a tabletop roleplaying game created by Matt Anderson, and is currently being played by the Pacific RPG club. Currently, the game master, the person in charge of running the game, must by hand keep track of the status of every monster, which includes all 6 of its combat stats, its health and mana, and any special effects it currently has. Additionally the GM must be able to apply modifications to these elements by hand. This is very difficult and time consuming. Our project seeks to simplify the process of managing monsters by presenting a simple interface and handling all math and stat management automatically.

3 Glossary

- Game Master: The person in charge of running a game of Fables of Sylvus©.
- GM: Game Master
- Player: A person controlling a character within Fables of Sylvus©
- Monster: A creature controlled by the game master
- stat: A specific attribute of the monster, such as how much damage this monster inflicts

4 Functional Requirements

4.1 Stakeholders

The system is primarily of interest to game masters, though players will also benefit greatly from the improved efficiency with which the game is run. As such, the entire Pacific RPG club could be considered a stakeholder.

4.2 Actors and Goals

Game masters are the main human actor in the system. They will initiate all actions made in the system, and their goal will be to have the monsters be affected based on the rules and events within the game.
The other actor will be our data, which is most certainly passive. The data will contain all possible monsters, how and when they may appear, what their stats are, and other information related to what the monster.

4.3 Use Cases

4.3.1 Casual Description

The system will begin with a single menu, in which the user can select what action they wish to take.

First, and by far the most common, will be that the game master wishes to input a player attack on a monster. There must be monsters present in the system already for this to occur. The game master will accomplish this by selection from a menu that a player has attacked a monster, selecting which monster was attacked, with what weapon, what the player rolled for the attack, and how much damage the player does with that weapon. The system will then update the health of the monster accordingly, and, if the monster has died, inform the game master of this and delete the monster if the game master wishes.

The next use case would applying a special effect to a monster. This also require monsters to be present. This would be accomplished by selecting to apply a special effect from a menu, selecting which monster, selecting what effect, and for how long and at what severity. The system will then apply the effect to the monster, modifying the monsters stats as appropriate.

The system will also be able to generate battles for the game master to use. To do this, the game master will select to generate monsters, indicate what part of the game world the players are in, how many players there are, and how powerful the players are. The system will then ask the game master if they want a normal set of monsters, a large horde, a few elite monsters, a single, very powerful monster, or if they want custom monsters. If the game master does not select custom monsters, the system will look at the data for monsters appropriate to the location, number and strength of the players and add them to the list of current monsters. If the game master does select custom monster, they will be prompted to enter in the name of the monsters and their stats, after which the monsters will be added to the list of monsters.

Finally, the user may wish to pass turns. To accomplish this, the user will indicate which monsters are attacking players, and in response, the system will determine how the monsters will attack, prompting the game master for any needed data to determine this. Afterward, the system will look at any special effects on the monsters, and modify the monsters to reflect the passage of time.

4.3.2 Fully Dressed use case

As a precondition for attacking a monster, there must be at least one monster present within the system. If the user attempts to have a player attack a monster while there are no monsters present, the player will be informed of the error and returned to main system menu.

First, the game master must select the menu option to have a player attack a monster. If the game master selects a different menu option, the action associated with that menu option will be taken. If the game master does not select an option, no action will be taken.

Next, the game master must select which monster is being attacked. If the game master selects a monster which cannot be attacked, the system will inform the game master of the error and prompt them again. If the GM selects to return button, the system will return the game master to the main menu. If the GM does not select a monster, the system will wait for the GM to select a monster.

4.3.3 Use Case Diagram

4.4 System Sequence Diagrams

5 User Interface Design

5.1 Preliminary Design

The interface will be menu driven, and will begin with the user at a main menu from which other functions of the system can be accessed. The user will be shown which key will result in which action. After this, the system will act as described above in the use cases section. Prompts will be given to the user in the form of text entry boxes, or, when deciding between a fixed number of options, assign each option to the key and determine the response by the user's keypress.

5.2 User Effort Information

For the case of attacking a monster, approximately 9 keystrokes are required, of which only 3 are to navigate the interface.

Applying an effect to a monster requires the same amount of effort as attacking.

For monster generation, about 12 keystrokes are required, due the necessity to type in a place name, with 2 of these being UI navigation. If a custom monster is to be created, far more keystrokes are needed due to the greatly increased amount of data required, approximately 35, with 9 being UI navigation.

Passing a turn will require somewhere around 7 keypresses due to the variablity of the game world, of which 2 are UI navigation.

6 Plan of Work

We intend to begin coding with the onset of the new year, as per standard for a computer science senior project.