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

Ryan Nish, Marissa Melen, Victor Aw

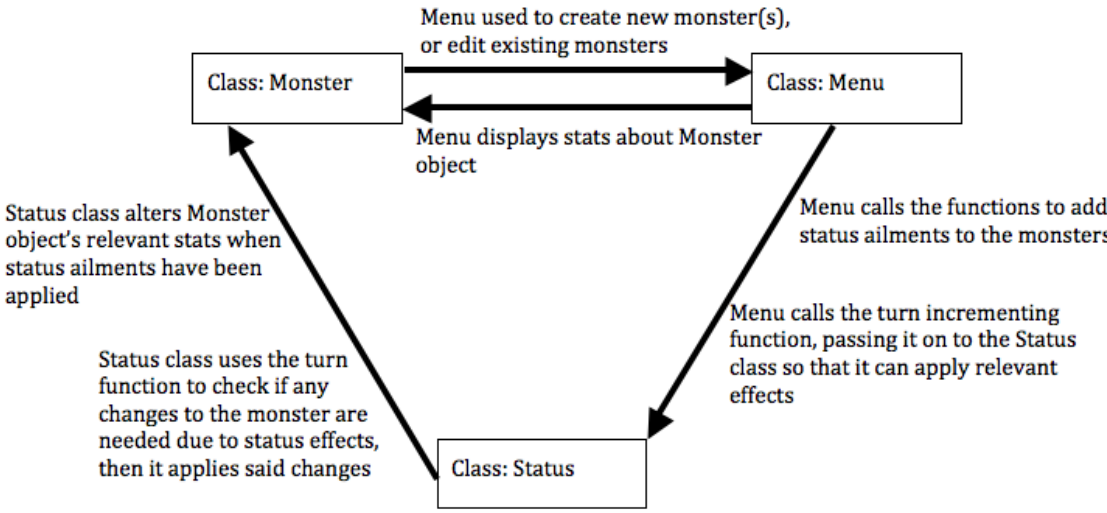
2013-11-13

1 Contributions

Features list - All team members contributed equally

Contents

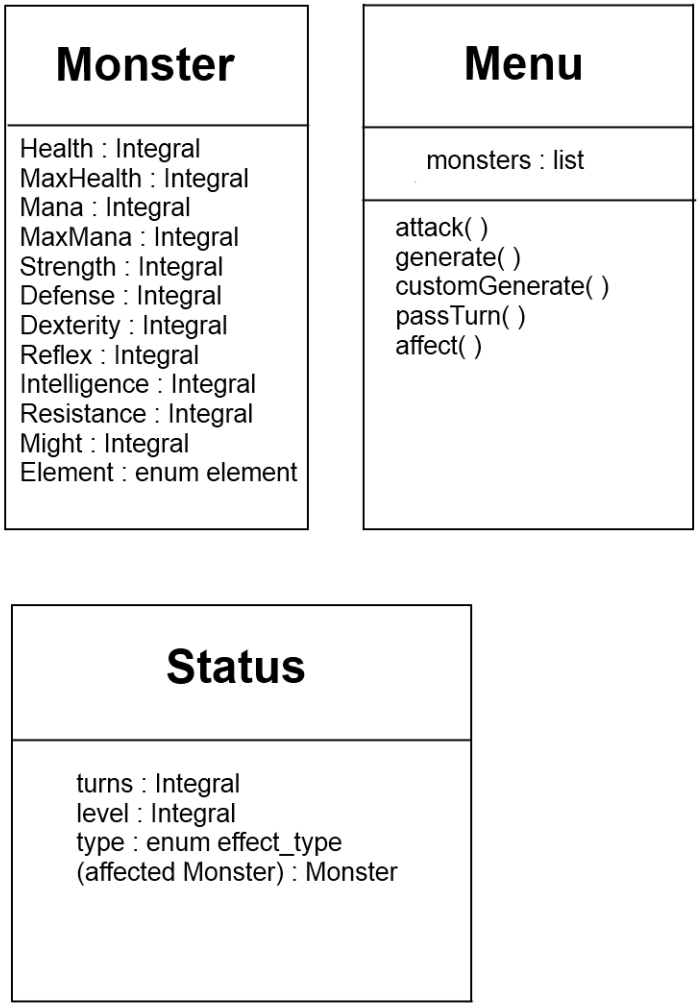
2 Interaction Diagrams



Interaction.png

3 Class Diagrams and Interaction Specifications

3.1 Class Diagrams



Diagrams.png

3.2 Interface Specifications

3.2.1 Monster

Health : Integral
MaxHealth : Integral
Mana : Integral
MaxMana : Integral
Strength : Integral
Defense : Integral
Dexterity : Integral
Reflex : Integral
Intelligence : Integral
Resistance : Integral
Might : Integral
Element : enum element

3.2.2 Menu

monsters : list

attack()
generate()
customGenerate()
passTurn()
affect()

3.2.3 status

turns : Integral
level : Integral
type : enum effect_type

(affected Monster) : Monster

4 System Architecture and Design

4.1 Hardware Mapping

Our system runs on only one machine

4.2 Network Protocols

Our system does not use the network

4.3 System Requirements

At a bare minimum our system could run on a computer with only an 80x25 terminal and minimal storage space, CPU, and RAM. However, our system would work much better on a system with a graphical display at a resolution the user is comfortable with.

5 Algorithms and Data Structures

5.1 Algorithms

Our project does not use any complex algorithms.

5.2 Data structures

First, our system will use a list-like structure, which may, for example, be a c++ vector. This was chosen because there is not need to order the data, but that its size cannot be known in advance, and so an expandable structure is needed. In addition, we will use functions to update the monsters as they are created, since all monsters will be created at level 1, and will need to be upgraded to a higher level.

We will also be using a functor, or mutable function, to represent status ailments and effects, because these by nature modify monsters, however we still need to have data associated with them, such as level, turns remaining, and the like.

6 User Interface Design

6.1 Basic Functions

Our user, the Game Master or GM, will use an interface that will have some basic functionality, such as the ability to input the number of players, and their average level so that the program can generate an appropriate number of monsters at an acceptable level. In addition, a monster can be selected as the target of an attack, and a menu will allow the choosing of the attack type (magic or physical), the weapon used (axe, spear, ect) the value the player rolled (this affects whether the weapon misses, grazes, or crits), and any additional effects that might happen. Such as weapon specific chances of status ailments or double strike. The monster menu will also allow the user to change the monster's stats directly, if such a need arises via player-specific ability or GM intervention. This menu will also facilitate the use of monster spells. Since monsters have their own spells, it is up to the GM to decide when they are used, and the menu will in turn make the necesary changes to monster mana or other stats.

7 Plan of Work

8 References