UNIVERSIDADE FEDERAL DE SÃO CARLOS
CAMPUS SÃO CARLOS
JOÃO VITOR BRANDÃO MOREIRA - 407496
LUCAS OLIVEIRA DAVID – 407917
THIAGO FARIA NOGUEIRA - 407534

DOCUMENTAÇÃO TRABALHO 3 - O APRENDIZ DE FEITICEIRO

SÃO CARLOS – SP
2012

1. INTRODUÇÃO

O objetivo deste trabalho é implementar uma aplicação para alguma(s) das estruturas estudadas anteriormente, além, obviamente, de identificar qual delas seria a melhor para tal implementação. Tal aplicação foi definida inicialmente como um jogo similar à consagrada franquia “Guitar Hero”, onde comandos são exibidos na tela e é exigido do usuário que ele execute-os de forma correta, resultando em acertou ou, caso contrário, em erro. Reaproveitamos o TAD “**lista duplamente encadeada com Header**” criada para o trabalho 2. Já que este havia sido implementado através do conceito de *Template*, foi simples utilizá-lo para criar listas de *CommandSpells*, objetos essenciais para o nosso jogo.

2. ESTRATÉGIA DE IMPLEMENTAÇÃO

O jogo Aprendiz de Feiticeiro se baseia em uma sequência contínua de comandos que devem bater com os comandos lançados pelo jogador com seu teclado.

O método escolhido para alcançar esse objetivo foi criar uma classe fase, onde contem todas as informações sobre a fase que será jogada, dentro dela, existe uma lista de "CommandSpells", classe com as informações de cada comando que aparecerá na tela e deverá ser igual à informação inserida pelo jogador, essa lista rege toda a lógica do jogo.

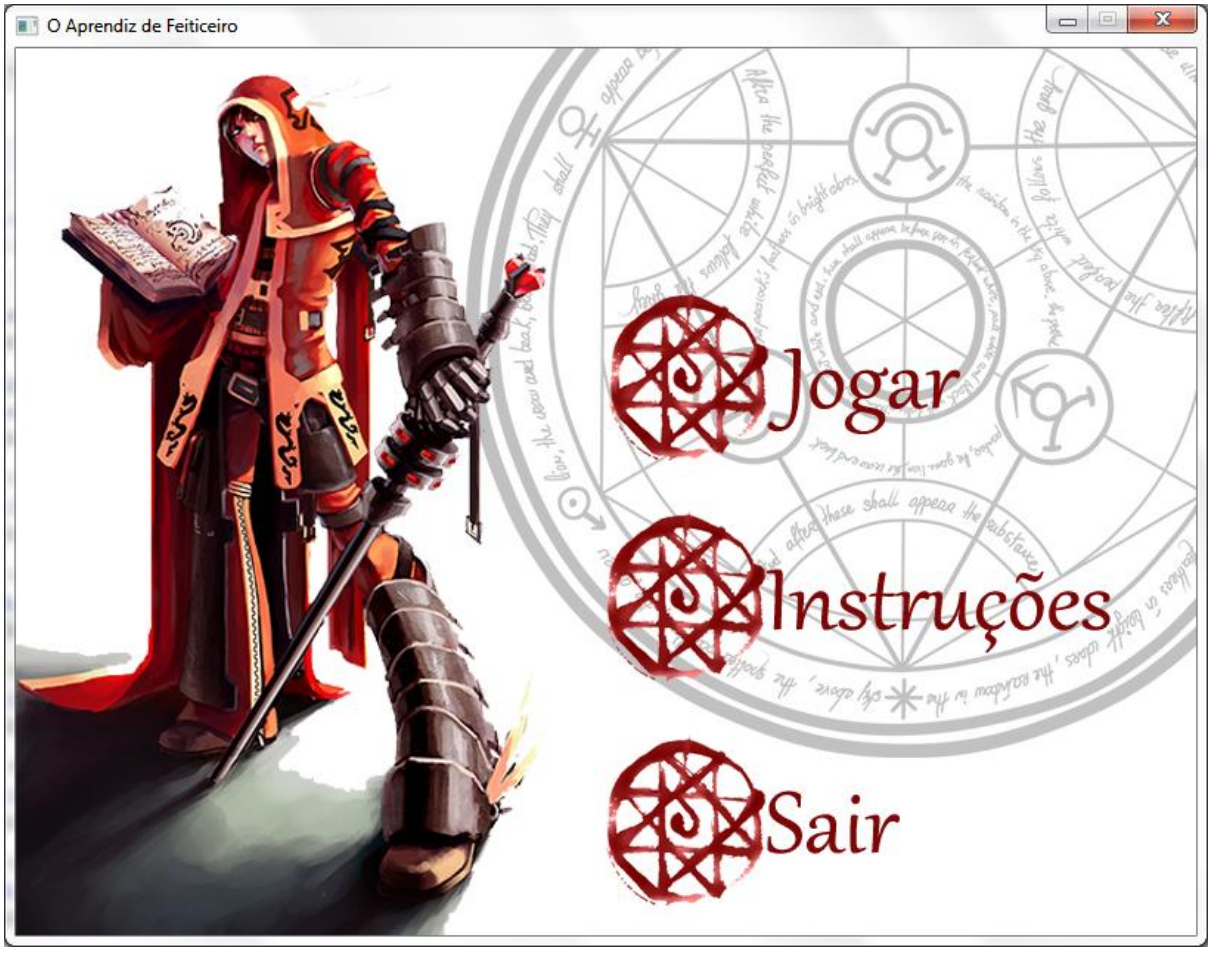
Quando uma fase começa o jogo procura pelo arquivo save.txt, nele fica guardada em qual fase o jogador parou de jogar, sabendo qual é a fase, ele busca o arquivo referente a essa fase e com as informações tiradas de lá cria a fase com a velocidade e espaçamento retirados do arquivo e monta a lista de CommandSpells.

Depois de montada a lista de *CommandSpells* é criado um ponteiro de node<CommandSpell> que começa apontando para a direita do header da lista de comandos, toda vez que o comando passa pelo círculo de invocação o jogo compara a entrada do teclado com o CommandSpell referente ao node apontado, troca o estado do comando para CERTO ou ERRADO e depois passa para o próximo node, isso é feito até que o node volte a apontar para o header, chegando assim ao final da fase.

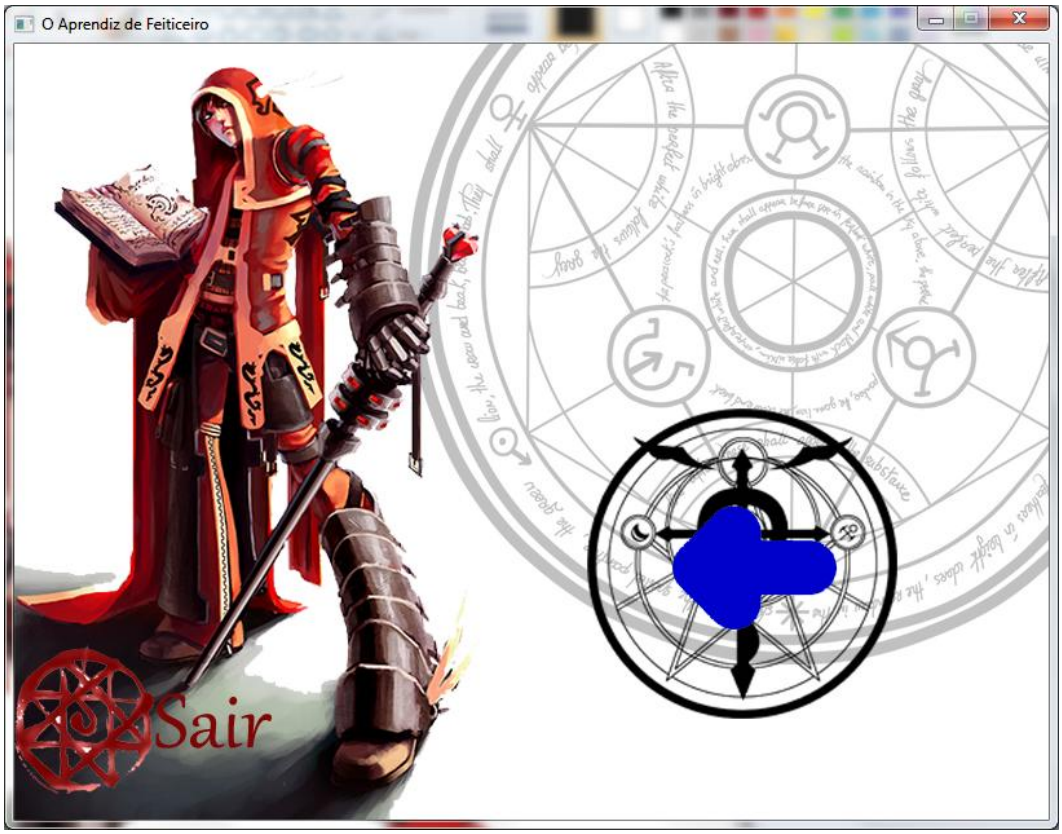
Com o fim da fase, é checado se o jogador acertou mais que 75% da fase, isso é feito ao percorrer toda a lista e checar quantos dos comandos estão em ERRADO e quantos estão em CERTO, se o jogador ganhou, surge a criatura invocada pela magia que o jogador acabou de executar, a criatura está ligada à fase que o jogador terminou, e o jogador pode escolher ir para a próxima fase ou voltar para o menu, quando ele for jogar novamente ele retornará à fase que ele parou.

4. DESCRIÇÃO DA JOGABILIDADE

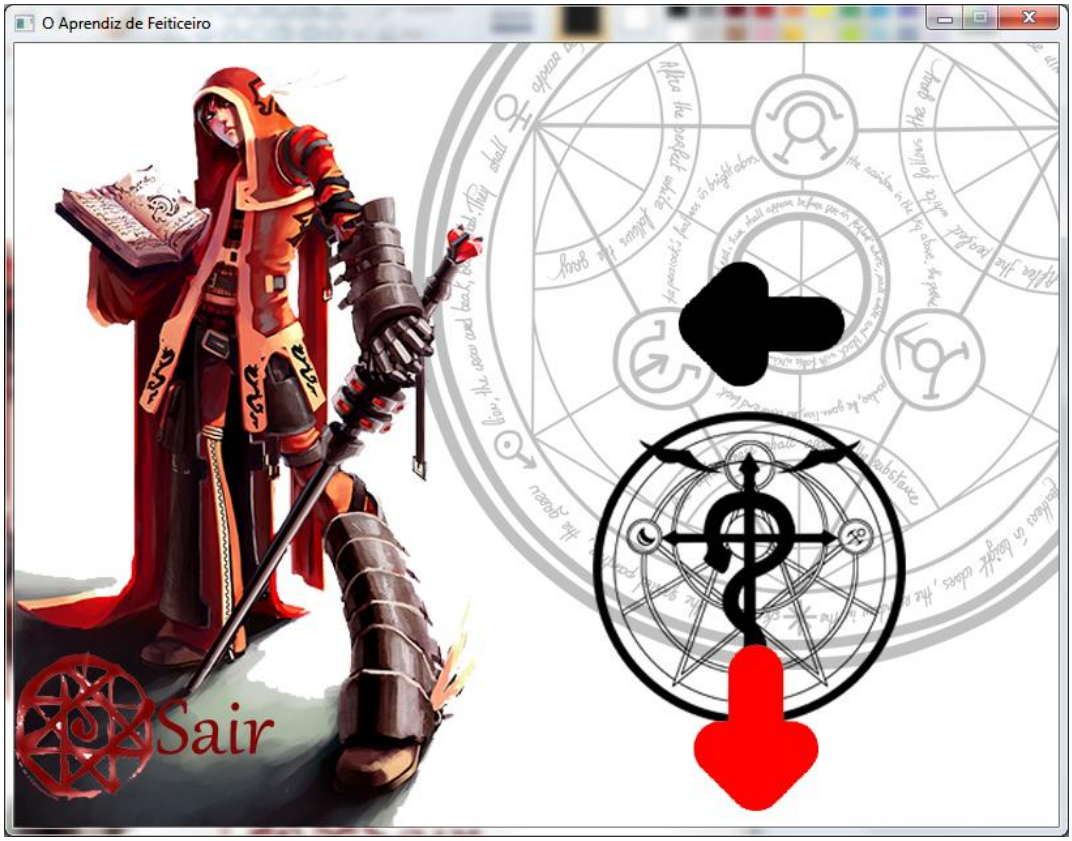
O mouse é utilizado para a navegação nos menus:



Um comando da lista durante o evento ACERTO



Um comando da lista durante o evento ERRO



5. CONCLUSÃO

Primeiramente, foi pensado em criar a sequência de comandos utilizando um TAD do tipo Fila, entretanto logo ficou claro a necessidade de se utilizar lista: desejávamos, no futuro, criar uma interface em que o usuário poderia desenvolver seus próprios feitiços e editar os existentes. Com a estrutura Fila, tal edição seria dificultada, uma vez que precisaríamos retirar os elementos anteriores ao que seria editado, retirar esse, editá-lo, recolocar todos os elementos retirados EM ORDEM, recolocar o elemento referido e retirar todos os outros e recoloca-los, a fim de que estes sigam para o fim da fila e mantenham a formação original. Com a utilização da lista, seria possível a simples identificação do *CommandSpell* desejado e sua edição direta.