ALGORITIMOS E PROGRAMAÇÃO – INF01202

RELATÓRIO TRABALHO FINAL

Felipe Leite - 296969 e Thiago Gawlinski - 327000

No presente relatório descreveremos a estruturação de implementação do jogo objeto do trabalho final da disciplina.

Funcionalidades implementadas:

Representação gráfica com a biblioteca conio, pontuação, cheat codes, aura, diferentes dificuldades, salvamento e carregamento de jogo, menu, contador de vidas.

Sumário de funções implementadas:

Introdução: esta função mostra uma mensagem inicial ao usuário, da um contexto ao jogo.

Conclusão normal: mostra mensagem de conclusão ao jogador caso tenha jogado em dificuldade fácil.

Conclusão difícil: mostra mensagem de conclusão ao jogador caso tenha jogado em dificuldade difícil e da dica sobre os cheat codes que podem ser ativados.

Seleciona dificuldade: esta função faz a seleção da dificuldade do jogo.

Ler mapa: esta função é responsável por verificar a existência e ler o arquivo de texto do mapa e transforma-lo em uma matriz que o programa ira utilizar para operar o jogo.

Controle de colisão: controla a colisão do jogador com os elementos do mapa.

Move: move jogador para diferentes posições, checa com a colisão se é possível fazer o movimento.

Move inimigo: faz a movimentação dos inimigos no mapa.

Aura: esta função quando chamada faz o surgimento da aura magica na tela e verifica se existe algum inimigo no domínio da aura, decrementando sua vida.

Pinta mapa: esta função mostra o mapa na tela através de elementos gráficos.

Execução: executa o jogo utilizando funções de leitura de mapa e recebendo parâmetros para.

Carregamento: realiza o carregamento de jogo salvo. Carrega um estado de jogo de um arquivo binário.

Salvamento: realiza o salvamento de um jogo em andamento. Salva um arquivo binário com o estado de jogo.

Cheat code: realiza a ativação dos chest codes covarde (vidas infinitas) e greyskull(morte instantânea)

Loop de jogo: insere o programa no loop de jogo, fazendo a troca de mapas.

Calcula pontuação: a função calcula a pontuação através da verificação dos inimigos mortos e captura do tesouro.

E por último, mas não menos importante a main, que é responsável pelas operações mais primarias para o funcionamento do programa.

Detalhes:

Para funcionar o jogo utiliza uma struct chamada ‘estado de jogo’, que passa informações essências, além das structs ‘inimigos’ que passa informações dos inimigos, ‘jogador’ que passa a posição do jogador, ‘dificuldade’ que passa as informações para iniciar o jogo em determinada dificuldade.

Basicamente através desses paramentos as funções trocam dados e realizam suas atividades.

Descrição da operação:

Assim que o programa executado ele mostra o plot do jogo e depois do input do usuário ele é levado ao menu principal, onde ele pode escolher criar um novo jogo, carregar um save existente, fazer um novo save ou retornar ao jogo. Quando o usuário seleciona criar novo jogo ele deve escolher a dificuldade desejada, após confirmação o jogo começa carregando o primeiro mapa.

Após coletar o tesouro do mapa ele é apresentado uma tela de pontuação e passa para a próxima fase. Isso ocorre até o usuário chegar a ultima fase que possui uma tela de vitória com a conclusão do jogo. Dependendo da dificuldade ele será apresentado uma tela diferente, deste modo ele toma conhecimento sobre os cheat codes presentes no jogo.

Durante o jogo o usuário pode pausar a qualquer momento e salvar o estado atual do jogo. Ele deve escolher o nome a ser dado a este save e assim o nome é arquivado em um arquivo .txt com todos os nomes de saves feitos. O estado de jogo atual é salvo em arquivo binário (.dat), possuindo todas as informações relevantes ao jogador como mapa atual, vidas restantes, dificuldade, pontuação, etc.

O carregamento é feito também através do menu principal, onde o usuário recebe a lista com todos os nomes de saves já feitos e deve inserir o save que deseja carregar.

Dentro do menu principal o usuário também pode habilitar códigos de trapaça, para isso ele deve apertar (‘) apóstrofe, as trapaças implementadas foram “covarde” que dá ao jogador vidas infinitas e “grayskull” que causa a morte instantânea do monstro que for atingido por ela.

Conclusões finais:

Ambos temos pouca experiência com programação e este trabalho foi excelente para nos acostumarmos com o ofício, eu (Felipe) tive um pouco de experiência com C# e Java e a linguagem C foi um desafio, pois muitas das funções já prontas para estas linguagens de alto nível não estão presentes e devem ser criadas do zero. Um exemplo que posso trazer é a função Split do C# que divide uma string em vetores de string quase que automaticamente, enquanto que em C deve ser feita usando strtok e outras funções da biblioteca de strings.

A minha (Thiago) percepção como iniciante foi muito positiva em relação ao trabalho, não tinha nenhum tipo de conhecimento prévio ou contato com a programação antes da disciplina e o desenvolvimento do jogo foi importante para entregar a experiencia de resolução de problemas mais complexos, onde é necessário buscar soluções de uma forma que haja a integração com os outras partes e a coesão do código, diferente das tarefas semanais onde exercitávamos conceitos isolados, além disso o trabalho nos permitiu a experiencia de trabalhar em grupo, onde é necessário termos a habilidade de ler e compreender os trechos de códigos escritos pelo colega.

Também gostaríamos de defender o nosso posicionamento quanto ao jogador poder passar de fase sem eliminar todos os monstros do mapa, se o jogador for obrigado a matar todos os monstros para ter permissão para mudar de fase isto significa que qualquer jogador que chegar ao final do jogo terá a mesma pontuação do outro. Para haver diferença de pontuação entre jogadores distintos que vencem o jogo é necessário que o extermínio dos monstros não seja necessário. Deste modo a pontuação não é redundante e tem importância.