Relatório do Projeto Final GDINF0107

Cícero Augusto Alcântara de Sousa
Ricardo Alexandre Cabral de Godói Filho

October 21, 2024

1 Objetivo do projeto

Implementar um jogo de xadrez. Deve ser possível mover peças e capturar adversárias.

2 Desenvolvimento

2.1 Técnicas e ferramentas empregadas

O projeto foi desenvolvido utilizando-se o Gedit e o GCC via linha de comando. O programa make também foi empregado para executar a compilação e, em alguns momentos, o GDB para depurar.

Usamos a plataforma Notion para o planejamento do projeto e o git e GitHub para o compartilhamento dos arquivos.

Desenvolvemos cada função separadamente. Cada tarefa correspondia a, no mínimo, um arquivo de cabeçalho e um arquivo fonte que continham a declaração e a definição de uma única função. A ideia era ir colocando funções correlatas que estivessem plenamente concluídas num mesmo arquivo. Cada função era testada utilizando-se uma main() num arquivo de testes.

2.2 Tipos de dados

Os tipos de dados mais importantes do programa são a estrutura Peca, a estrutura Movimento e a estrutura Tabuleiro.

A estrutura Peca armazena as informações relevantes de uma peça: seu tipo, cor, posição e se ela já foi movida – embora esta última só seja utilizada nos casos do peão, torre e rei. Essa estrutura, por conveniência, armazena também uma constante que a interface usa para acessar a sprite da peça.

Na estrutura Peca, as informações de tipo e cor têm seus valores representados em enumerações. A enumeração Pecas é uma enumeração simples; a enumeração Cores tem os valores BRANCA = 1 e PRETA = -1. Isso é proposital, dessa forma fica mais simples de implementar o movimento dos peões usando

uma única função sem ter que recorrer a *ifs* para decidir se a linha é decrementada ou incrementada.

A estrutura Movimento contém as informações sobre os movimentos do jogo: dois campos representam as coordenadas da casa de destino do movimento e o terceiro representa sua natureza: deslocamento ou captura.

A estrutura Tabuleiro armazena as informações sobre o estado do tabuleiro: as peças que estão em jogo, a última peça aliada que foi clicada e os movimentos possíveis para ela, e a cor das peças que estão jogando. As peças em jogo são representadas por listas duplamente encadeadas de estruturas Peca e os movimentos possíveis por listas do mesmo tipo, mas de estruturas Movimento.

As estruturas ElementoPeca e ElementoMovimento são interfaces para que as estruturas Peca e Movimento, respectivamente, possam funcionar como listas duplamente encadeadas. Elas possuem um ponteiro para o tipo de estrutura que elas adaptam e dois ponteiros para seu próprio tipo de estrutura, que devem apontar para o elemento anterior e o posterior da lista. Os elementos são acessados usando ponteiros para essas estruturas. Algumas funções foram criadas para manipular essas listas, elas performam, por exemplo, as operações de inserção e remoção de elemento.

Foram criadas funções para inicializar as estruturas de Peca, Movimento e Tabuleiro. Elas usam alocação dinâmica de memória para criar as estruturas. Isso é importante porque, na maioria das vezes, essas estruturas são usadas em associação com suas interfaces para lista, e seria muito trabalhoso declarar e manusear todas as variáveis para todos os dados envolvidos. Então, chama-se uma única função com os dados necessários para criar tudo.

2.3 Interface

A interface foi implementada em OpenGL. Usou-se projeção ortográfica – afinal de contas, não era necessário 3D. Além disso, é mais fácil manipular os cliques do mouse com esse tipo de projeção.

O método natural para se desenhar as *sprites* em OpenGL seria usar texturas, no entanto, isso se mostrou um pouco mais complexo do que o esperado, já que envolve o uso de *shaders*, os quais têm uma linguagem de programação própria. O desenho das *sprites* foi feito com uma técnica um pouco menos elegante, mas bastante eficaz: usou-se o software GIMP para exportar as informações dos *pixels* em um formato que o programa pudesse ler. O GIMP, além de *raw data*, consegue exportar imagens como arquivos fonte e cabeçalho de C. Decidiu-se pela versão em arquivo fonte, que consiste essencialmente das informações das dimensões da imagem e uma grande *string* cujos caracteres representam os *bytes*. Através dessas informações, os *pixels* foram desenhados como pequenos quadrados na tela. Não houve grande prejuízo computacional ou estético em aplicar essa técnica: as *spites* escolhidas são em *pixelart* e tem tamanhos irrisórios (16x16).

A interface consiste somente de uma tela com as casas do tabuleiro e as peças em 2D. Para mover uma peça, basta clicar nela e os movimentos possíveis serão destacados no tabuleiro. Depois, é só clicar na casa correspondente ao

movimento desejado. O programa não se preocupa em determinar qual objeto estava mais a frente no ponto em que ocorreu o clique. O *buffer* de profundidade da OpenGL sequer é usado. Usa-se apenas a ordem do desenho para criar a sobreposição: primeiro, as casas, segundo, os movimentos possíveis e por último, as peças. Como tudo é dado em coordenadas de tabuleiro, basta determinar em qual casa ocorreu clique, passar as coordenadas ao administrador de jogo e deixá-lo determinar o que deve ser feito.

2.4 Administrador de jogo

O administrador de jogo recebe as coordenadas de tabuleiro do último clique e testa se já há uma peça tocada na variável global TABULEIRO. Se não houver, ele procura nas peças da vez uma que se localize na posição clicada. Se encontrar alguma, esta será a peça tocada e o campo PecaTocada apontará para ela. Em seguida, os seus movimentos são gerados e salvos no campo MovimentosPecaTocada. Se já houver uma peça tocada, ele procura em MovimentosPecaTocada um movimento com as coordenadas da casa clicada. Se sim, o movimento é executado, se não, testa se a casa clicada é ocupada por uma peça aliada. Se sim, esta é a nova peça tocada, se não, nada acontece.

2.5 Depuração

A maior parte da depuração foi feita via logs no terminal. Foram criadas funções para mostrar informações de estrutura na tela e algumas para criar valores fixos para teste.

Em certos momentos, foram empregados scripts para o gdb.

3 Conclusão

O projeto entrega o prometido, porém, ainda pode melhorar bastante em termos de implementação. Existem partes do código que podem ser melhor organizadas e seria interessante aperfeiçoar as operações com listas, para manipular melhor as listas e criar um código mais uniforme e simples. Adicionar convenções de escrita ao código para que fique tudo o mais uniforme possível.

No que diz respeito à interface, seria interessante implementar mais opções de jogo: jogar contra a máquina, níveis de dificuldade, opções de relógio, salvamento.

Pretendemos medir algumas implementações – referente ao tempo de execução, por exemplo – e ver como elas se saem.

Também gostaríamos de desenvolver algumas ferramentas para facilitar o desenvolvimento via linha de comando: uma ferramenta para adicionar header guarders nos arquivos de cabeçalho; uma ferramenta que inspecione os arquivos e decida os cabeçalhos que cada um deve incluir; outra que inspecione os arquivos e gere um makefile.

Por último, pretendemos refazer o projeto em C++ e comparar os dois códigos.