Universidade Do Minho

Curso G005 - Mestrado Integrado em Engenharia Informática

Disciplina Laboratórios de Informática II

Ano Letivo 2018/2019

Relatório

Projeto Reversi

Autores:

[a87963] Maria Sofia Martinho Gonçalves Jordão Marques

[a90122] José Nuno Baptista Martins

[a89232] Pedro Nuno Nogueira Pereira

Turno PL6 Grupo 4

Data 02/05/2019

Docente

Resumo: encontraremos neste relatório algumas explicações sobre as funções mais cruciais para o desenrolamento deste trabalho, tal como, uma vista mais superficial quanto ao bot e as estratégias tomadas para o desenvolvimento dos diferentes níveis.

Universidade Do Minho

Introdução

Será neste relatório que iremos passar a explicar as funcionalidades de algumas das nossas funções utilizadas para o desenvolvimento do jogo.

Sendo uma delas e a responsável pelo comando <H> (hint "?").

Esta e dada pela função maxPlay (Jogada.c) e basicamente consiste na obtenção da melhor jogada que e possível fazer de acordo com o estado atual do jogo.

As funções getValidPlays, isValidPlay e checkCerco (Jogada.c) assumiram um papel extremamente importante para a implementação deste comando e do comando <S> (jogadas validas "."), pois foi com estas que conseguimos avaliar a melhor jogada possível a fazer.

A função checkCerco com o auxílio da função cercoByDir verifica através de vetores se uma determinada jogada é ou não valida. Isto é possível gracas a variável contador, presente na função cercoByDir, que nos diz o número de pecas do adversário que esta jogada premitirá voltar, tendo em conta a posição dada para a jogada e o estado atual do jogo. Caso nenhuma peca adversaria seja voltada em nenhuma direção (ou seja contador=0) então e porque a jogada não e valida, caso contrário, onde a soma de todos os contadores para as diferentes direções (cima, baixo, direita, esquerda, e os eixos diagonais) é diferente de zero, então é porque a jogada é possível, pelo menos para uma dessas direções. Assim termina retornando-nos como valida essa jogada.

E também assim que se vai construindo a lista das posições (graças as coordenadas das jogadas validas retornadas) na função getValidPlays.

Ou seja o comando <H> é nos possíveis uma vez que, dado o estado do jogo no momento, este com auxílio das funções faladas anteriormente, vai verificar cada uma dessas posições, da lista das posições obtida em getValidPlays, e determinar qual a jogada a ela associada, que permite obter uma maior score. Sendo essa a que ira ocupar a posição da hint.

A função responsável pelo comando <U> (undo) foi também conseguida através de várias outras funções. Sendo que no entanto a ideia geral consiste na utilização de um histórico (em memoria dinâmica) cujo funcionamento se relaciona com o funcionamento de uma stack, isto é, apos cada jogada feita o estado do jogo e alterado e guardado no histórico. Onde a próxima alteração ficara guardada "no topo" do histórico. Por sua vez possui um apontador que liga ao estado anterior.

Estes acontecimentos, relativamente ao funcionamento do histórico, são dados na função pushGame (SaveNLoad.c).

No entanto a função Undo é em si dada pela função popGame (SaveNLoad.c).Que consiste então, e tendo em conta a estrutura do histórico, em retirar do mesmo o ultimo estado, passando o estado atual a ser igual ao estado anterior a jogada feita.

Não esquecendo a necessidade de atualizar a pontuação dos jogadores apos essa alteração.

Relativamente as estratégias do bot, fomos capazes de criar um bot com 3 níveis de dificuldade sendo o 1º nível dito o mais "fácil", o 2º o nível de dificuldade intermédia, e o 3º um nível mais desafiante.

Cada nível esta associado a diferentes funções que nos permitem obter diferentes resultados como o pretendido.

No 1º nível criamos um função semelhante a função responsável pelo 3ºnivel, onde com o auxílio da lista das posições de jogada possíveis, procuramos a jogada que menos pecas do adversário vira.

No 2º nível criamos uma jogada aleatória, com o auxílio do sistema rand, em que consiste na obtenção da lista das jogadas possíveis, através da qual o bot escolhera aleatoriamente uma posição para efetuar a jogada.

No 3º e último nível vamos usar o mesmo mecanismo que o nosso comando <H> utiliza, ou seja, utilizaremos a lista das posições possíveis, de acordo com o estado do jogo, e iremos escolher como posição para a peca do bot aquela que permitira obter uma maior score.

Conclusão

No desenrolar deste trabalho deparamo-nos com algumas dificuldades as quais conseguimos superar, e neste sentido fomos capazes de desenvolver as nossas capacidades de escrever código em c.

Conseguimos concluir a maioria das tarefas, deixando apenas a tarefa relativa ao campeonato entre bots.

Sabemos que não prestamos tanta importância a objetividade e qualidade do código como demos a execução do mesmo, no entanto, concluímos o nosso objetivo de entregar um jogo funcional.

Universidade Do Minho