UNIVERSIDADE FEDERAL RURAL DO RIO DE JANEIRO

Ciência da Computação

Lucas de Oliveira Nóbrega

Rita Zhen Chen

ATIVIDADE ACADÊMICA DE COMPUTAÇÃO II

NOVA IGUAÇU

2018

Sumário

1. Introdução

A Atividade Acadêmica da disciplina de Computação II propõe a implementação do jogo "Combate" utilizando a linguagem Java e conceitos de Programação Orientada a Objetos. O objetivo é capturar a bandeira adversária e é jogado por dois jogadores: o usuário, o qual é representado pelas peças vermelhas, e o computador, pelas peças azuis.

O jogo desenvolvido possui duas dificuldades: fácil, a qual é um modo com as peças inimigas visíveis (DEBUG), e normal, em que não tem a visibilidade das peças do computador. Além disso, possui dois modos de distribuição: de forma aleatória e a que o usuário escolhe as posições delas.

O jogo acaba quando alguma bandeira for capturada ou quando algum jogador não tiver mais opção de se movimentar.

2. Funcionalidades implementadas

- Som: Foi implementado o áudio de fundo do jogo e pode ser silenciado ou ter a música retomada ao apertar o botão relacionado ao som, localizado no canto superior direito. O método correspondente a essa funcionalidade é o "play".
- Dica: Foi implementada uma classe relacionada à dica, na qual seu objeto é instanciado assim que o usuário solicita informações sobre a bomba adversária. São apenas duas dicas a serem solicitadas. Quando se tem ou não uma bomba na coluna escolhida pelo usuário, a mesma janela o informará se possui ou não. Além disso quando o jogador não possuir mais dicas, uma janela surgirá indicando que não tem mais ajuda.
- Pause: O pause está localizado dentro do jogo e pode ser ativado ao apertar o botão correspondente ou com o botão "escape" do teclado. Nesse painel, há as opções de "Voltar ao Jogo", a qual retorna ao jogo normalmente, "Reiniciar", que reseta todos os contadores de dicas e reposiciona as peças com as posições iniciais — caso esteja dentro do jogo —, ou zera todas as peças — caso o usuário esteja na tela de escolher peças — , "Menu", a qual se cria um novo objeto da classe Game, recriando o jogo do zero, retornando ao menu principal, e a opção de "Sair", que fecha a janela.
- Tela de vitória/derrota: Assim que o usuário ganha ou perde, surgirá um painel com a imagem de vitória ou derrota, respectivamente. Além disso, possui dois botões: "Novo Jogo", que se cria um novo objeto da classe Game, recriando o jogo do zero e retorna ao menu principal, e "Reiniciar Jogo", que reposiciona as peças com as posições iniciais.

- Dificuldades: O jogo apresenta duas dificuldades: fácil e normal. O modo fácil é na forma DEBUG, no qual todas as peças do inimigo (computador) são visíveis, enquanto que no modo normal é sem DEBUG, no qual não há visibilidade das peças do adversário no início do jogo, porém após um ataque e a peça inimiga ganhar do combate, esta fica visível.
- Distribuição de peças: O jogo possui dois modos de distribuição de peças: de forma aleatória e o de que o usuário escolhe as posições de cada uma de suas peças.
- Movimento do computador: São sorteados uma das peças do computador restantes e uma direção — para cima, baixo, direita, esquerda —, mas quando se tem uma peça do usuário próxima, ele a ataca. O método correspondente a essa funcionalidade é o "movimentosComputador".
- Aleatoriedade: A aleatoriedade de peças e direções é gerada com o método "randInt" com o auxílio do método Random do pacote java.util.
- Combate: Para verificar os ataques do inimigo, foi utilizado o método "movimentos", e para verificar os ataques do usuário, foi utilizado o método "processClick".
- Atualização do tabuleiro: Altera as posições das peças e atualiza os contadores de peças restantes.
- Fim de jogo: O jogo termina quando captura a bandeira inimiga, ou quando as peças restantes não podem se mover — verificada pelo método "verificaEncurralado" — ou quando apenas restam as peças imóveis (bombas e bandeira) — verificada pelo método "verificaBombaBand". Todas essas verificações estão dentro do método "verificaFim".

3. Compilação do programa

Para compilar o programa corretamente, siga os passos a seguir:

1. Instale o Java e a IDE NetBeans;
2. Faça o download do ZIP do site https://github.com/ritazchen/CatchTheFlag;
3. Crie um projeto no Netbeans e, em seguida, um package chamado AA;
4. Copie todos os arquivos do ZIP e cole na pasta criada;
5. Abra o arquivo Game.java e o execute com o comando SHIFT+F6.

4. Exemplos de utilização

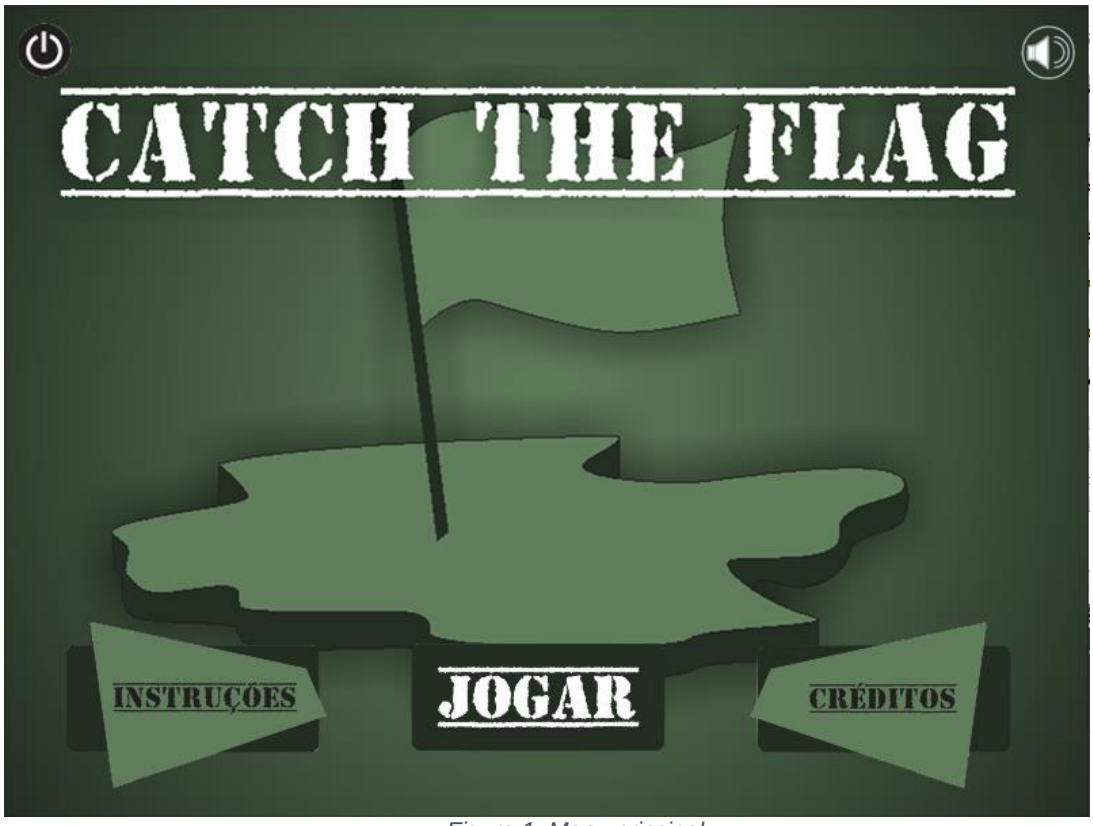

Figura 1. Menu principal

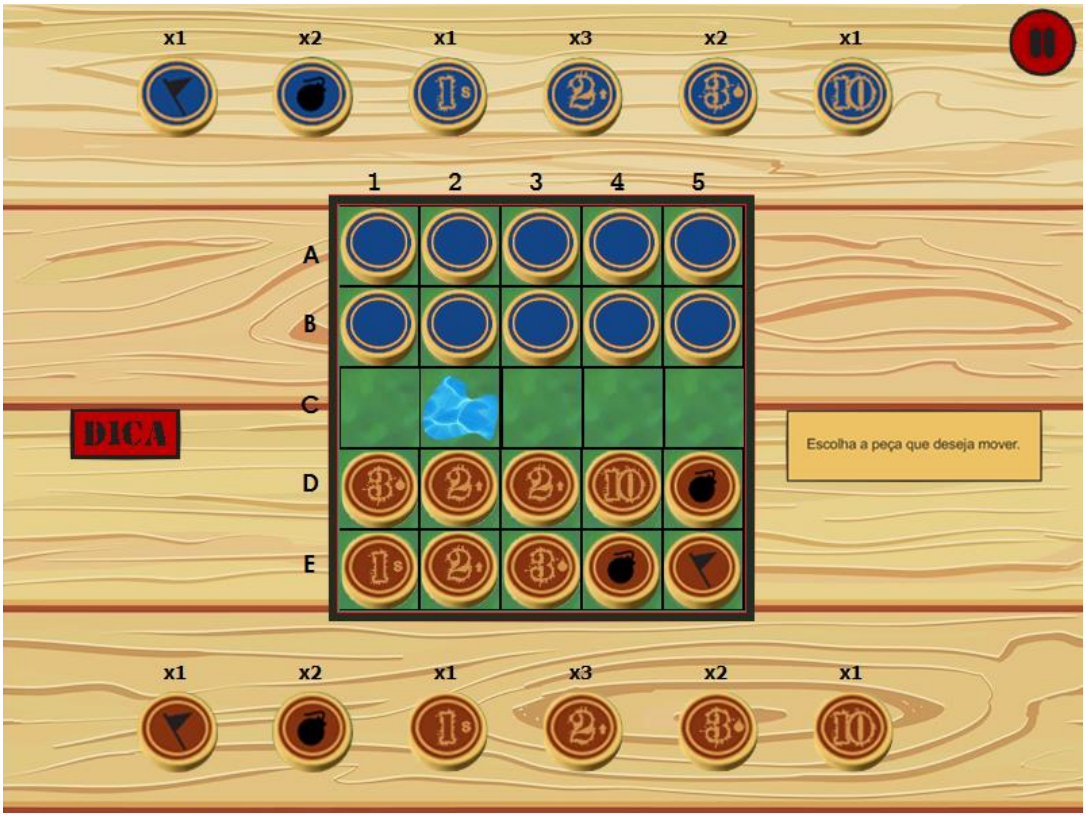

Figura 2. Modo de distribuição aleatória e com dificuldade normal

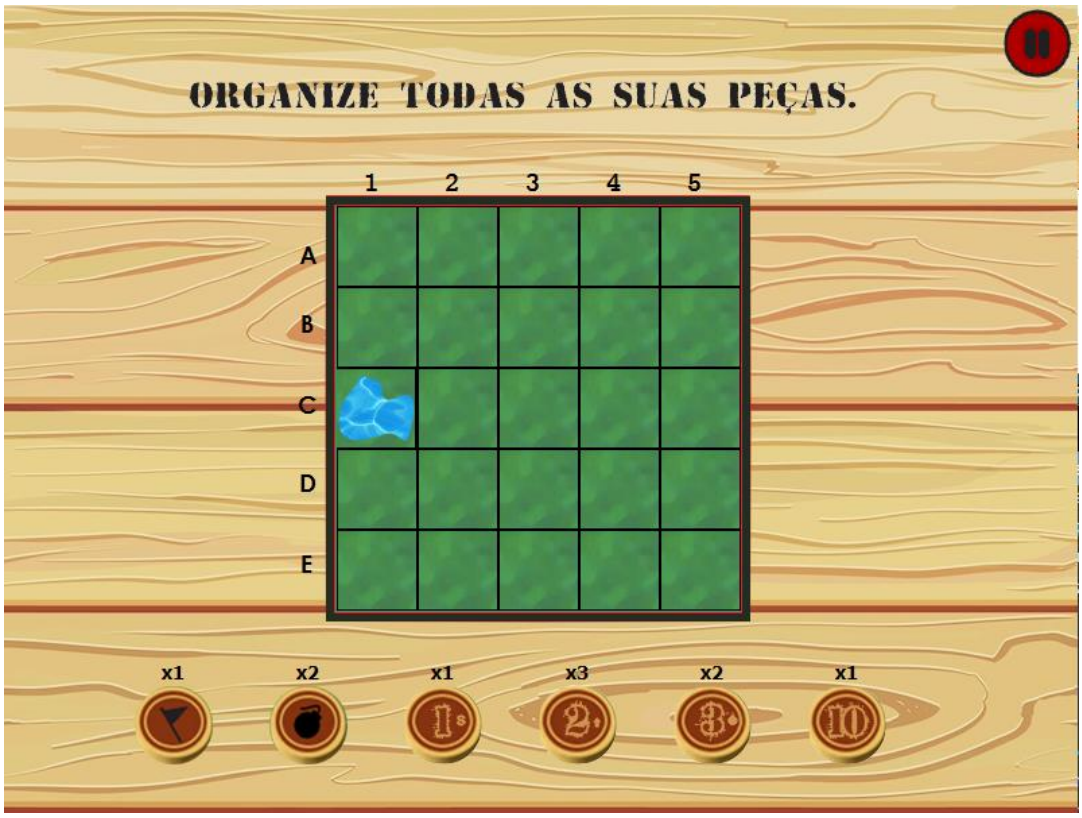

Figura 3. Modo de distribuição de peças à escolha do usuário

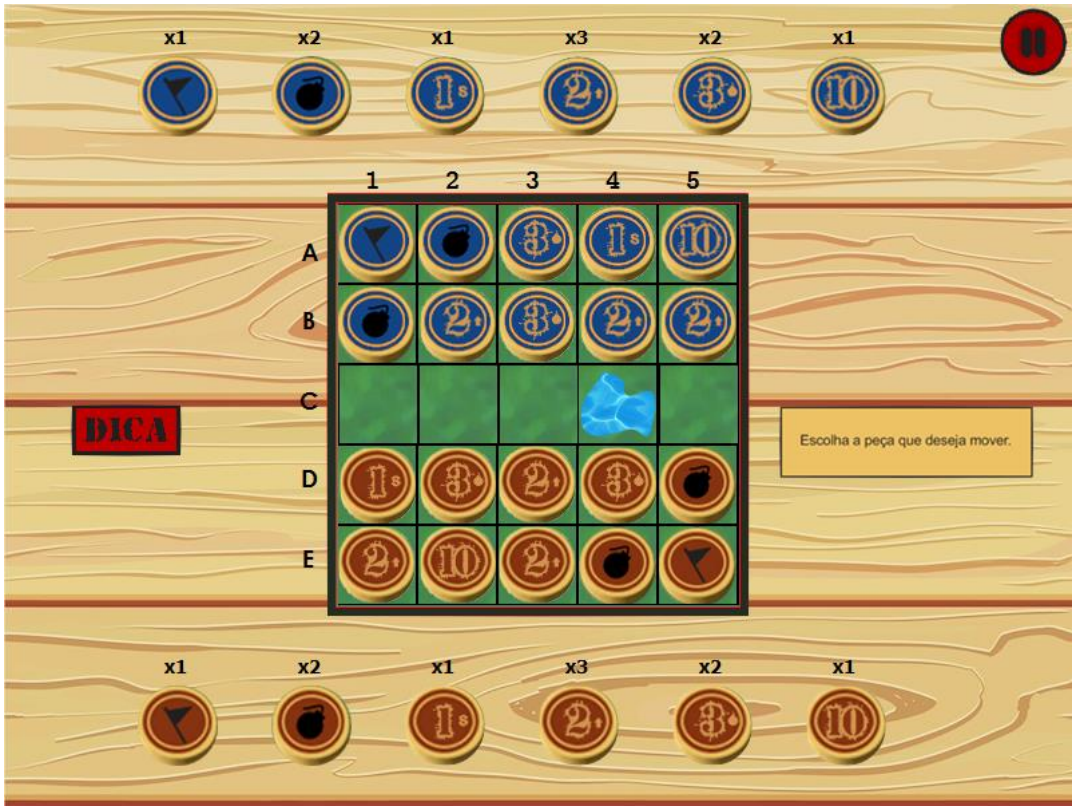

Figura 4. Dificuldade fácil

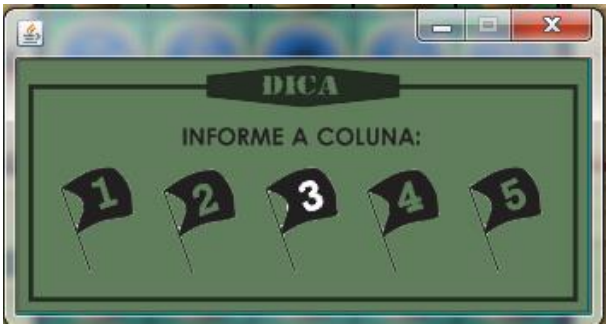

Figura 5. Janela de dica

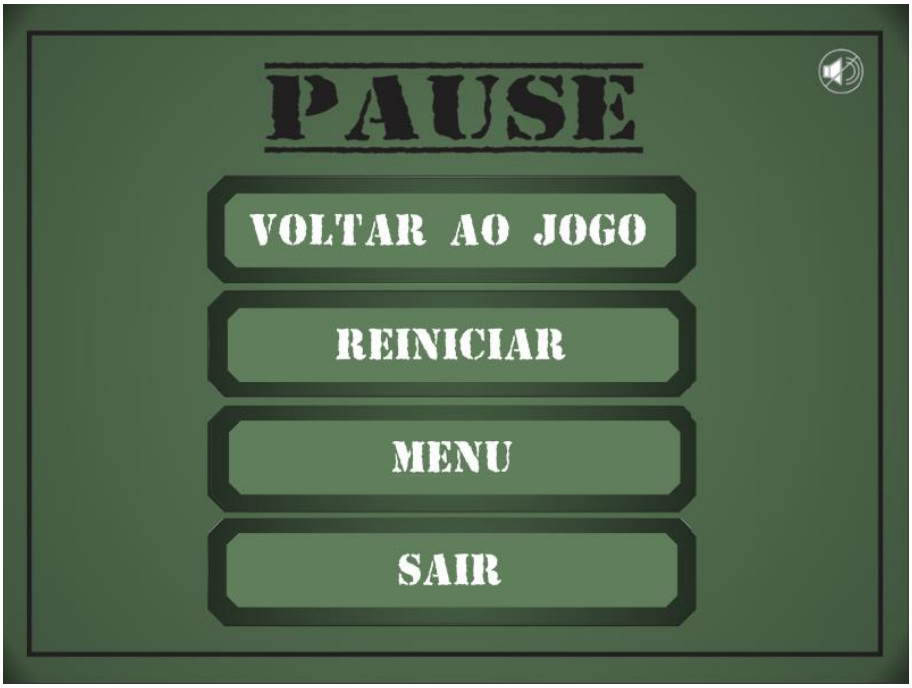

Figura 6. Tela de pause

Figura 7. Tela de fim de jogo indicando que o usuário perdeu

5. Aplicação de conceitos de Programação Orientado a Objeto

Para desenvolver este jogo, foram utilizados os seguintes conceitos de Programação Orientado a Objeto (POO): os quatro pilares — Encapsulamento, Composição, Herança e Polimorfismo —, variável "final" e o controle de fluxo "instanceof", além dos comandos básicos da linguagem Java.

A Composição foi utilizada para implementar o áudio de fundo do jogo, utilizando as classes URL e AudioClip importadas dos pacotes padrões do Java: java.net e java.applet, respectivamente. Além disso, a composição está presente ao criar uma matriz como atributo, usando a classe Peças, para representar o tabuleiro do jogo, possibilitando o reaproveitamento de classes e melhora na qualidade do programa.

O Encapsulamento é evidente ao utilizar getters e setters para possibilitar a visibilidade de alguns atributos que possuem modificadores de acesso como privados, sem poder alterá-los diretamente em outra classe. Um exemplo disso é que o contador de dicas (cont) da classe Game é um atributo privado, logo, é necessário ter um método público get para retornar o valor deste dado na classe DICA2 para ter o controle de dicas restantes que o usuário possui.

A Herança está presente quando as peças específicas — classes Vazio, Lago, Bandeira, Bomba, Espião, Soldado, CaboArmeiro e Marechal — herdam atributos e métodos da superclasse Peças. Trata-se de uma herança simples, pois todas as classes mencionadas anteriormente possuem apenas uma classe-mãe (Peças) e, quando são herdadas, é necessário utilizar o termo "extends" e em seguida o nome da superclasse, após a declaração do nome de cada classe-filha.

O Polimorfismo é visível ao conferir ao tabuleiro, que é um atributo da classe Peças, a possibilidade de cada posição da matriz possuir objetos diferentes herdados de Peças. Logo, em cada posição do tabuleiro, pode-se ter um objeto, como por exemplo, de bomba ou bandeira, caracterizando, então, o polimorfismo de inclusão.

Para se descobrir o tipo de peça que está em uma determinada posição do tabuleiro, utiliza-se o "instanceof", possibilitando conferir se o objeto em questão é uma instância de uma classe que se deseja e, em seguida, realizar operações com ela. Além disso, vale comentar sobre o termo "final", que foi utilizado apenas para manter constante a posição do lago a cada objeto da classe Game criado. Ressalta-se também que mesmo se reiniciar o jogo, o lago permanece na mesma posição inicial gerada aleatoriamente, a menos que se crie um novo jogo ou retorne ao menu principal.

6. Conclusão

A Atividade Acadêmica proposta possibilitou melhorar a compreensão acerca da Programação Orientada a Objeto utilizando a linguagem Java, permitindo assim um bom desenvolvimento do programa.

O jogo “Catch The Flag” foi desenvolvido conforme os requisitos do orientador e foi necessária uma lógica não muito problemática, logo, a sua implementação foi simples. Porém, destaca-se algumas das partes do programa um pouco mais complicadas de se implementar, que foram a construção do grid layout do tabuleiro e a movimentação com cliques do usuário, esta sendo possível após a verificação de condições (regras do jogo).