Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Bacharelado em Ciências da Computação
Disciplina de Programação Orientada a Objetos - SSC0103-1
Prof. Dr. Márcio Delamaro
Documentação Externa do Trabalho Final

Especificação do Jogo Duck Hunt

Alunos:

Ewerton Patrick Silva do Amaral	**nºUSP:10346975**
Vinicius Torres Dutra Maia da Costa	**n°USP:10262781**
Gabriel Citroni Uliana	**nºUSP: 9779367**
Caio Abreu de Oliveira Ribeiro	**n°USP:10262839**

26 de junho de 2018

Sumário:

1. Introdução

O jogo implementado foi baseado no título Duck Hunt da Nintendo, porém foram realizadas modificações estéticas, na jogabilidade e na lógica, além de um novo modo multiplayer, em que o segundo player assume o papel do pato e tem que desviar do caçador.

Para a implementação foram utilizados os conceitos de classes para modularizar o código, dividindo em classes que controlam cada pato que aparece na tela, que desenham as imagens na tela, que identificam as entradas do mouse e do teclado, que realizam a temporização do jogo, etc. Utilizou-se também as bibliotecas gráficas do Java 8 para desenvolver a interface.

2. Jogabilidade:

2.1. Modo single player:

Neste modo, o jogador assume o papel do caçador, controlando, por meio do mouse, a mira de uma arma. A cada round existem 4 ondas de patos, com 3 patos cada. O jogador possui 4 tiros durante cada onda de patos e a onda termina quando o atirador gasta os 4 tiros ou quando os 3 patos são acertados. Para avançar ao próximo round deve-se acertar no mínimo 8 dos 12 patos que surgem a cada round. Caso contrário, resulta no fim do jogo (*game over)*. A pontuação segue a seguinte regra: a cada pato morto, o atirador ganha 500 pontos, caso tenha acertado o pato em uma região próxima das extremidades da hitbox do pato e 1000 pontos, caso tenha acertado na região central do pato. No final da partida, se houver *game over*, é disponibilizada a pontuação final do jogador.

2.2. Modo multiplayer:

O modo de jogo possui dois jogadores, um que controla o caçador e o outro o pato. O caçador não possui limite de disparos, controlando a mira da arma através do mouse. O Pato utiliza as setas do teclado para definir uma direção de movimento e a tecla espaço para realizar uma batida de asas, subindo a altura do pássaro. Para ganhar pontos, o caçador deve acertar a maior quantidade de tiros nos patos do adversário, em contrapartida, o jogador controlando o pato deve se esquivar dos tiros do outro jogador, ganhando pontos para cada tiro errado do caçador. Quando o caçador acerta um pato ele recebe 500 ou 1000 pontos variando da região do pato atingida, já jogador controlando o pato recebe 200 pontos para cada tiro que o caçador erra. A partida dura 60 segundos, o jogador com a maior pontuação no final do tempo é o vencedor.

2.3 Controles:

Caçador: Mouse para mirar e botão esquerdo do mouse para realizar o disparo.

Pato: Setas direcionais esquerda e direita para a direção do pato e barra de espaço para bater as asas.

3. Detalhes de implementação:

3.1. Classes principais:

Passaro: Essa classe implementa o comportamento de cada um dos pássaros presentes no jogo, tendo atributos como a posição do pato na tela, a velocidade em cada eixo, o valor da constante gravitacional que atua na física do jogo e outras variáveis que controlam o estado do pato. O construtor recebe como parâmetro o estado do jogo, sendo 1 se for o modo single player ou 2 para o modo multiplayer, a cor do pato, para selecionar o *sprite* correto, e os limites da tela. No caso de single player, um método chamado vooIndefinido() controla o movimento do pato, sorteando valores aleatórios para mudança de direção e para a velocidade no eixo x, para a quantidade de batidas de asas (*flap()* - inverte a velocidade para negativa no eixo y), isto é, quantas vezes o pato baterá as asas para subir ou não no eixo y, além de sortear a posição no eixo x em que o pato irá surgir e de sortear um tempo para que não haja batidas de asas, ou seja, para que o pato desça de posição.

Há também o método update(), que atualiza a posição x e y do pato de acordo com o tempo e as velocidades nos eixos, garantindo que os limites da janela sejam respeitados pelo movimento do pato.

Outro método importante é o desenhar(), o qual define quais os *sprites* devem ser desenhados na tela de acordo com o estado atual do pato, podendo estar para a direita ou esquerda, voando para cima, para baixo ou para a frente, morto ou não. Este método utiliza um métodos de outra classe (Tela) que irá realizar os corte em x e y (passados por parâmetro) no *sprite* definido e o desenhar na tela por meio da classe Graphics2d do Java. Semelhante a esse método, existe o desenhaPontuacao(), que escreve na posição do pato na tela o número de pontos ganhos pelo usuário, caso o estado do pato esteja morto.

O método hitbox() define as posições da tela em que um clique deve matar o pássaro e incrementa a pontuação atual do usuário, caso acerte o pato. Para iso, define-se que um clique em qualquer posição interna ao retângulo no qual o pato está inserido define um acerto, diferenciando a pontuação de acordo com a região do pato que foi acertada. Muda-se, também, o estado do pato entre morto e vivo.

Por fim, há o método reiniciar(), que é chamada quando um pato passa para o estado morto e irá surgir novamente na tela. Redefine-se todos os atributos do pato para os iniciais, com exceção da velocidade do *flap*, que é incrementada para que a dificuldade do jogo aumente a cada rodada, e do limiar do flap, para que o pássaro bata as asas mais vezes em um mesmo instante de tempo, ou seja, aumente a velocidade no eixo y para cima.

Tela: Essa classe realiza a impressão de todas as imagens na tela, utilizando um objeto do tipo Graphics2D, já implementado na API do JAVA, para desenhar as imagens escolhidas na tela. Para isto, utiliza-se o método imagem(), que recebe por parâmetro o nome do arquivo de imagem, a posição em x e y para realizar um corte e as dimensões da parte da imagem que se quer imprimir. No caso do jogo, utilizou-se *sprites* de diferentes estados dos patos, dos estados do jogo e dos elementos de contagem para desenhar na tela.

Duck: É a classe que organiza a lógica do jogo, implementando a interface jogo. Possui o atributo *game_state* que define em que estado o jogo está no momento. O

estado 0 é o menu principal, o estado 1 é o modo single player, o estado 2 é o multiplayer, o estado 3 é a tela de créditos, o 4 é a tela de espera entre rodadas, 5 é a tela de fim de jogo para o single player e 6 é a tela de fim de jogo para o multiplayer. Para cada uma delas, o método desenhar() define o que deve ser mostrado na tela e gerencia todos os atributos das classes auxiliares para que a execução do jogo ocorra. Nessa classe é onde se instancia os objetos do tipo Passaro.

Os métodos mouse() e tecla() gerenciam as interações do jogo com o usuário, definindo a resposta para as teclas do teclado ou cliques de mouse em regiões específicas da tela. No caso do mouse(), como é a interação principal do usuário, podendo escolher o início do jogo, o modo ou os créditos, define-se qual deve ser a região em que ocorreu o click para que haja determinada resposta, dependendo do estado atual do jogo. Por fim, o método tique() realiza a atualização dos atributos que variam com o tempo, no caso, a posição e velocidade dos patos.

Motor: A classe motor executa a parte do jogo relacionada à interface gráfica, definindo os atributos da janela, instanciando o *frame* e o painel em que serão desenhadas as imagens, determinando as bordas da tela, criando os objetos das classes gráficas do Java que gerenciam as imagens, instanciando o cursor e implementando os métodos de captura das teclas e do mouse. Portanto, esta classe é responsável por executar o laço de repetição principal do jogo, utilizando um contador de tempo para realizar a atualização da tela do jogo, chamando os métodos que desenham na tela a cada iteração do loop, além de passar por parâmetro o tempo atual do jogo, o qual será responsável por atualizar a posição dos patos na imagem.

3.2. Classes Auxiliares:

ScoreNumber: Essa classe é responsável por guardar os dados do usuário, como a pontuação total e o número de patos que o jogador matou em cada rodada. Os métodos dessa classe são utilizados para escrever números (a partir de um *sprite* com todos os algarismos) que representam a pontuação total e a pontuação por acerto.

Random: A classe gera números aleatórios em intervalos definidos por parâmetro, para serem usados em variáveis do jogo.

Balas: A classe controla a impressão das balas disponíveis para o caçador e administra a quantidade de balas restantes em cada onda de patos no modo single player.

Jogo: Interface que define alguns dos métodos a serem implementados na classe Duck.

3.3. Principais Bibliotecas:

java.awt: Utilizada para auxiliar no desenvolvimento da interface gráfica, por meio das classes gráficas para desenhar as imagens na tela, da classe para desenhar o cursor, para determinar as dimensões da janela e das classes de eventos, que capturam as entradas do mouse e do teclado.

javax.swing: Utilizada para criar a tela do jogo, isto é, o *frame* e o *panel* onde serão inseridas as imagens (*sprites*).

javax.imageio.ImageIO e java.io.File: Utilizada para tratar leitura dos arquivos das imagens.

4. Interface Gráfica e Execução:

A interface inicializa com a tela de menu principal, no qual o usuário pode escolher qual modo de jogo irá jogar ou se deseja visualizar os créditos do jogo. A escolha de uma dessa 3 opções é feita por meio do clique do mouse no retângulo correspondente.

Figura 1 - Menu principal do jogo

Caso seja selecionada a opção de single player, inicia-se a primeira rodada em que o jogador deve cumprir o objetivo do jogo, nesta tela são mostrados o número de balas restantes, o número de patos restantes, o número de patos acertados e a pontuação, além dos 3 patos que movem de maneira aleatória. Caso o jogador consiga cumprir o objetivo, as próximas rodadas são realizada, caso contrário o jogo mostra a tela de *game over*, mostra a pontuação total do jogador e uma opção para voltar ao menu (tela inicial).

Figura 2 - Modo Single Player.

Caso seja selecionada a opção de multiplayer, é iniciado a tela de multiplayer com o pato que será controlado pelo segundo jogador, com um contador de 60 segundos, com a pontuação do jogador que controla o pato e do jogador que controla o caçador. Quando o contador zera, é mostrada a tela de *game over* com o vencedor e a pontuação de cada jogador, além da opção de retornar ao menu.

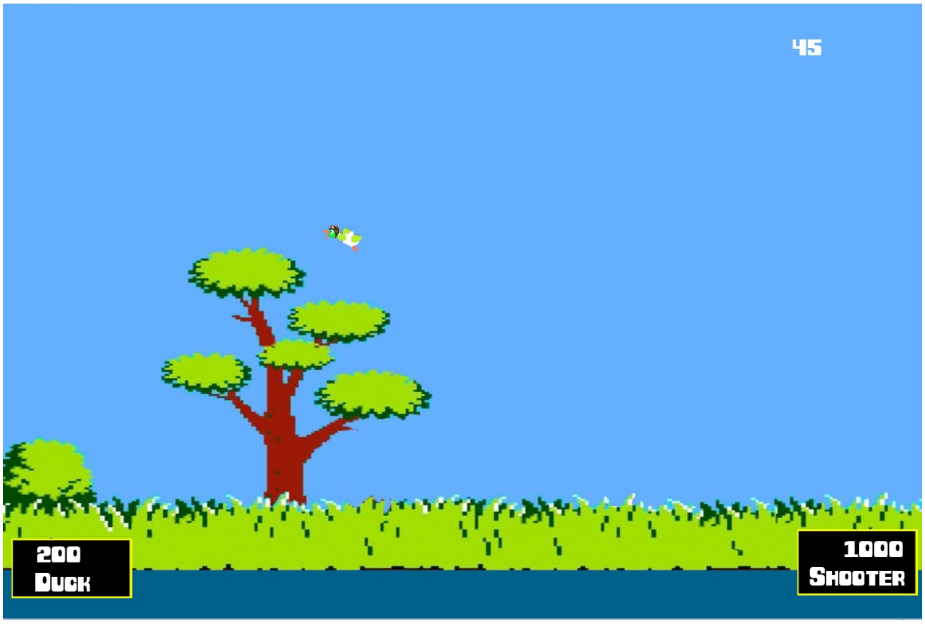

Figura 3 - Modo Multiplayer

Figura 4 - Game Over - Multiplayer.

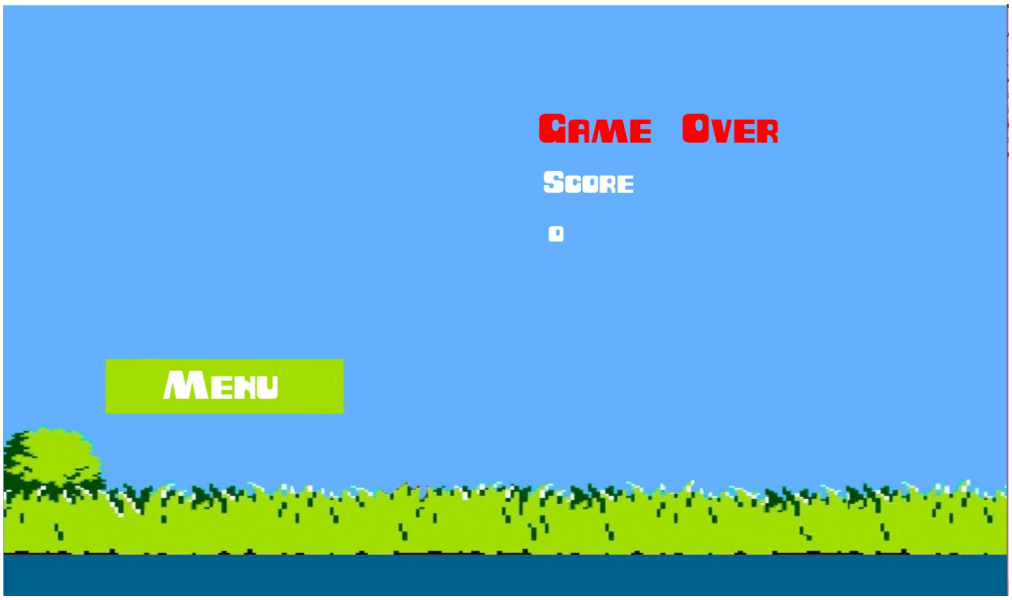

Figura 6 - Game Over - Single Player

O Trabalho foi dividido igualmente entre os membros do grupo:

Ewerton Patrick Silva do Amaral => 25%

Vinicius Torres Dutra Maia da Costa => 25%
Gabriel Citroni Uliana => 25%
Caio Abreu de Oliveira Ribeiro => 25%

Figura 6 - Créditos.

5. Referências Bibliográficas:

https://stackoverflow.com/
https://docs.oracle.com/javase/8/docs/api/