Programação Concorrente

Trabalho Prático

Choque de Glutoes

Realizado por: Grupo 20

André Araújo a87987

Carlos Ferreira a87953

Daniel Ribeiro a87994

Servidor

O servidor encontra-se dividido em 7 classes auxiliar, conversores, login_manager, server, criaturas, jogadores e estado.

Auxiliares: A classe auxiliar é composta por um conjunto de operações auxiliares tais como multiplicaVector/2 que multiplica um vetor por número, normalizaVector/1, adicionaPares/2 que adiciona dois vetores, distancia/2 que indica a distância entre dois vetores, posiciona/2 que indica uma posição sem nenhum obstáculo para o raio dado e geraObstaculo/2 que adiciona um certo número de obstáculos a uma lista de obstáculos.

Conversores: A classe conversores tem as funções formatarPontuacoes/1 que formata as pontuações para depois enviar para os clientes, formataTecla/1 que formata a tecla enviada pelos clientes para indicar quais operações devem ser realizadas e a função formatState/1. Esta é responsável transformar o estado para enviar aos clientes para eles utilizarem essa informação para apresentar o jogo, e as suas funções auxiliares tais como, jogadores_para_string/1 transforma uma lista de jogadores em uma string, jogador_para_string/1 que auxilia a jogadores_para_string/1 e transforma um jogador em uma string, criaturas_para_string/1, criatura_para_string/1, obstaculos_para_string/1 e obstaculo_para_string/1.

Login_manager: A classe login_manager é a classe responsável por criar uma conta, fechar uma conta e efetuar o login e logout de um jogador. Esta classe contem as seguintes funções start_Login_Manager/0 responsável por iniciar o login_manager, create_account/2, close_account/2, login/2, logout/1. Contendo este um loop que guarda as informações das contas tais como o nome, palavra passe e se esta encontra-se online, esta recebe os comandos das outras funções e atualiza o Map onde estas informações são guardadas com eles.

Server: A classe server é responsável por iniciar o servidor fazer conexão entre o cliente e o servidor indicando qual é a operação que o cliente pediu. A função start/0 é responsável por iniciar o servidor iniciando um processo com estado e

outro com o login_manager, criar um socket e de o enviar para a função acceptor/1, esta aceita uma requisição de conexão criando outro processo acceptor para podermos ter mais de um cliente e inicia o authenticator/1 com essa conexão.

O authenticator/1 é responsável por ler uma mensagem do socket e vai tentar efetuar a operação descrita caso ela seja de login, create_account ou close_account. Caso seja um login bem-sucedido vai para a função user/2. Esta espera o utilizador ter uma vaga para entrar no jogo quando consegue chama o cicloJogo/3 este recebe e lida com as mensagens vinda do utilizador.

Criaturas: A classe criaturas alberga as funções relativas as criaturas tais como novaCriatura/2 que gera uma nova criatura, atualizaListaCriaturas/2, que com o auxílio da função atualizaCriatura/2 que move a criatura e lida com as colisões com as paredes e das funções verificaColisaoObstaculos/2 e da verificaColisaoObstaculo/2 que resolvem a colisão com os obstáculos, lida com o movimento da criatura. Nesta classe ainda temos verificaColisoesCriaturaLista/2 que com o auxílio da verificaColisaoCriatura/2 verifica a colisão entre um jogador e uma lista criaturas.

Jogadores: Cada objeto jogador contém os seus essenciais sendo alguns deles constantes como o RaioMax, RaioMin, EnergiaMax, Arrasto, AcelaracaoLinear, AcelaracaoAngular, GastoEnergia e GanhoEnergia. Também contém as variáveis como a EnergiaAtual, Velocidade, Raio, Direcao, Agilidade, Pontuacao e Posicao.

A classe jogadores contem as funções novoJogador/1 que cria um novo jogador, acelerarFrente/1 que ativa o propulsor que anda para que acelere para a frente, viraDireita/1 e viraEsquerda/1 que ativam os propulsores que viram o jogador e a função atualizaJogadores/4.

Esta atualiza o jogador com o auxílio das funções movimentaJogador/1 que move e atualiza parâmetros do jogador, verificaColisaoObstaculos/2 que com o auxílio da verificaColisaoObstaculo/2 sinaliza os jogadores que colidiram com o obstáculo e perdem e atualiza os que não perdem, verificaColisaoJogadores2/3 que auxilia verificaColisaoJogadoresL/2 a verificar se os jogadores colidem e atualiza o seu estado em caso positivo, atualizaColisaoVerdes/2 que caso um jogador colida com uma criatura verde atualiza o seu estado recebendo bónus decrescentes dependendo do seu tamanho e atualizaColisaoVermelhos/2 que quando um jogador colide com uma criatura vermelha atualiza o seu estado.

Estado: Esta classe e constituída pelas funções start_state/0 que inicia os processos com o gameManager/2, o refresh/1, adicionarVerdes/1,

adicionarReds/1 e executa a função estado/2. O refresh/1 envia de tanto em tanto tempo uma mensagem ao gameManager/2 com o comando refesh para este atualizar o estado e envia-lo aos jogadores, a adicionarVerdes/1 envia de tanto em tanto tempo uma mensagem ao gameManager/2 para este ver se é possível adicionar criaturas verdes e a adicionarReds /1 envia de tanto em tanto tempo uma mensagem ao gameManager/2 para este ver se é possível adicionar criaturas vermelhas.

O processo estado/2 recebe do servidor os pedidos de entrar, entrando no jogo e na lista de jogadores atuais e se esta não tiver cheia e caso contrário vai para lista de espera, e sair do jogo deixando entrar outro que se encontra na lista de espera.

As funções novoEstado/0 que cria um novo estado para iniciar um jogo, adicionaJogador/2 que adiciona um jogador ao estado, removeJogador/2 que remove um jogador do estado e atualizaMelhoresPontos/2 que atualiza as melhores pontuações dos jogadores auxiliam o gameManager/2.

O gameManager/2 gere o jogo recebendo varias mensagens de outros processos e realizando as operações, que estes pedem dentre as operações mais essenciais para o funcionamento do jogo vem das mensagens keyPressed que com o formataTecla/1 na conversores e da função updateTeclas/3 atualiza a velocidade e direção de um jogador, a refresh que realiza o update/1 atualizando o estado e atualiza as melhores pontuações dos jogadores com a função atualizaMelhoresPontos/2 e envia para os utilizadores o estado de jogo atulizado.

A função updateTecla/2 indica que um jogador pressionou uma tecla e diz qual das funções acelerarFrente/1, viraDireita/1 ou viraEsquerda/1 devemos usar para atualizar o jogador auxiliando a função descobreJogador/3 que descobre o jogador que clicou na tecla. Estas duas funções auxiliam o updateTeclas/3.

A função update atualiza o estado para isso, começa por verificar as colisões entre os jogadores e as criaturas com verificaColisoesCriaturaLista/2, de seguida retira as criaturas que colidiram com jogadores e atualiza as restantes criaturas na função atualizaListaCriaturas/2. De seguida a atualizamos os jogadores na função atualizaJogadores/4 para concluir filtramos os jogadores que perderam na função filtrar/2 que devolve a Lista dos jogadores que ainda continuam e os dados dos jogadores que perderam para indicar que já perderam.

Cliente

De forma a desenvolvermos a nossa interface gráfica em Java utilizamos tal como sugerido o Processing, e para a criação de botões e janelas usamos mais concretamente o g4p que é uma livraria de Processing que possui um builder onde podemos criar e personalizar a nossa janela e o código é nos gerado automaticamente o que facilitou o processo da construção da interface gráfica.

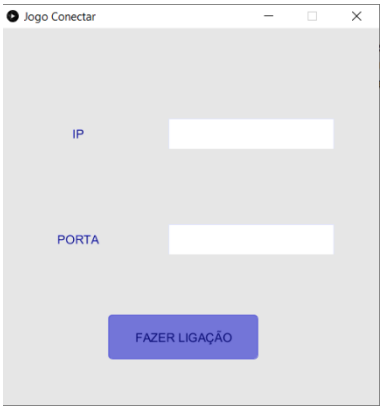
O nosso menu disponibiliza 4 opções:

- **Registo de conta:** escolher nome e palavra-passe pretendida, nenhum dos campos pode ser deixado em branco nem o nome da conta pode já estar a ser utilizado.

- **Cancelar o registo:** digitamos o nome da conta e a palavra-passe correspondente de forma a eliminar o nosso registo.

- **Opções:** onde podemos mudar a porta e o endereço IP a onde nos vamos conectar.

- **Login:** digitamos o nome e a palavra-passe correspondente, caso haja vaga entrámos em jogo, no caso de 3 pessoas já estarem numa partida esperámos numa lista de espera até alguém perder ou abandonar a partida.



Cada botão do menu é mediado pelo seu handler, onde comunicámos com a socket as nossas ações.

Para a parte de desenhar a tela, temos duas **threads**, uma que lê e atualiza o estado corrente do jogo e a outra thread atualiza as pontuações globais a cada 5segundos sendo o ciclo de vida desta última desde quando se abre a janela pontuações globais até ao fecho dessa mesma janela.

Classes Java:

Conector: classe que nos permite conectar com o servidor via Sockets TCP, nesta alocámos a Socket , o BufferedReader e o PrintWriter. Temos 4 métodos nesta classe, o connect que recebe um endereço ip e uma porta e cria uma

Socket, o nosso BufferedReader será então criado a partir da inputstream da socket , e o PrintWriter criado a partir da outputstream da socket.

O método read vai ler a informação da stream a partir do nosso BufferedReader, o método write escreve para a stream a partir do nosso PrintWriter, o método disconnect fecha a nossa socket.

Criatura: classe onde alocámos informação da criatura que apenas interessa para a apresentação na tela de jogo, como a posição, a direção, o tipo da criatura e o tamanho desta.

Temos um método draw onde dependendo do tipo da criatura a sua cor vai variar, ou seja, tipo = 0 então sabemos que é uma criatura verde, caso contrário sabemos que é vermelha, então desenhámos um círculo com a cor e tamanho respetivo.

Apartir do ângulo da direção e de métodos disponibilizados pelo Processing tal como o translate e rotate, desenhámos um triângulo para sabermos a direção para onde a criatura se está a movimentar.

Obstáculo: classe onde alocámos a posição e tamanho do obstáculo e temos um método draw que desenha um círculo preto com o tamanho respetivo do obstáculo.

Jogador: classe que aloca o nome, a pontuação do jogador, a sua posição, direção, tamanho, energia, agilidade e o nome do jogador do cliente que nos ajuda a saber se este jogador é o jogador do cliente ou é um adversário.

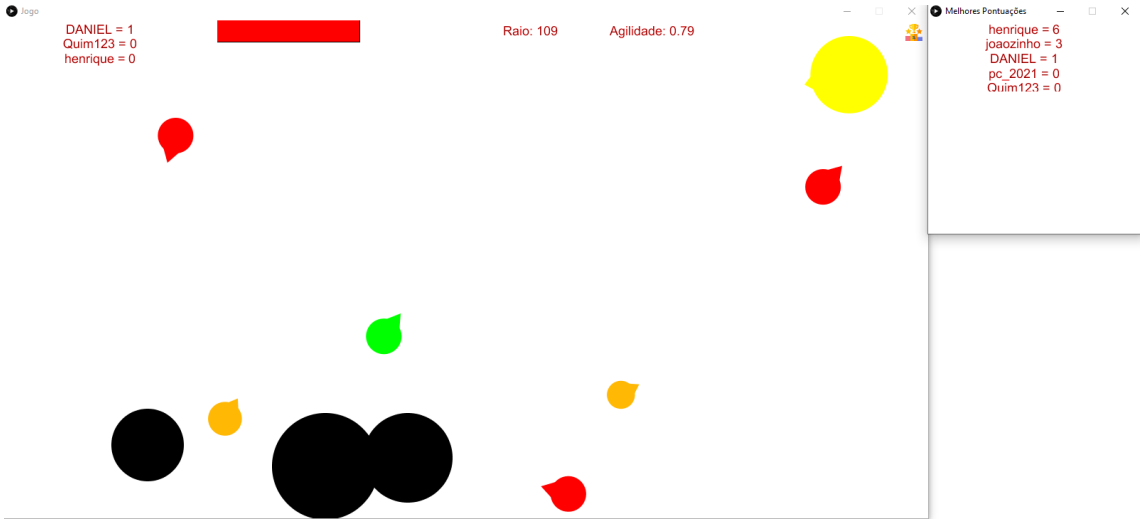
Temos um método draw onde desenhamos o jogador do cliente como um círculo amarelo com o respetivo tamanho do jogador e desenhamos também um triângulo para sabermos a direção. Caso o jogador não seja o do cliente mudámos a cor para um amarelo mais escuro.

Jogo: classe onde alocámos toda a informação que queremos que seja representada na tela, desde um array com as criaturas, outro array com os obstáculos e outro array com os jogadores. Temos também um HashMap para as pontuações atuais do jogo.

Esta classe para além do método de criação, tem um método de update onde utilizamos um **ReentrantLock** garantindo que temos **exclusão mútua**.

Temos um método draw onde apenas percorremos os arrays e chamamos as funções draw das outras classes.

Resultados:


Jogo
DANIEL = 1
Quim123 = 0
henrique = 0
Raio: 109
Agilidade: 0.79
Melhores Pontuações
henrique = 6
joaozinho = 3
DANIEL = 1
pc_2021 = 0
Quim123 = 0


Jogo
PERDEU