CONGRESSO CATARINENSE DE CIÊNCIA DA COMPUTAÇÃO
2019. Ano 6, Vol. 6. ISSN: 2319-0655

JOGO DIGITAL – SPACE FX

José Oscar dos Santos Junior[1]
[1]Instituto Federal Catarinense – IFC- Rio do Sul
Jose_santos_junior@outlook.com,

Abstract. *This work has as main objective, the development of an electronic space ship game called Space FX, created in javaFx language, the main objective of the game is to capture small ships along the way, where they accumulate points.*

Keywords: *electronic game; javaFX; ship game*

Resumo. *Este trabalho tem como objetivo principal, o desenvolvimento de um jogo eletrônico de espaço nave chamado de Space FX, criado na linguagem javaFx, o jogo tem como objetivo principal capturar pequenas naves pelo caminho, onde vai acumulando pontos.*

Palavras-chave: *jogo eletrônico; javaFX; jogo nave.*

1. Introdução

Assim que os primeiros computadores receberam formas gráficas como saída, os jogos digitais começaram a ser criados.

Segundo Conti (2015), 1952, aparentemente nesse ano surgiu o primeiro jogo, 0x0, escrito por Alexandre S. Douglas e deve ter sido executado no computador EDSAC.

A partir da década 60, os jogos evoluíram acompanhando a evolução dos computadores, até se tornarem o que são hoje, jogos realistas, simulando perfeitamente a vida real, tendo campeonatos mundiais, com públicos muito alto.

Este trabalho irá mostrar o que acontece por traz de um jogo, o que é necessário para a criação do mesmo, nesse caso foi criado o jogo chamando Space FX, nele uma nave navega pelo espaço capturando pequenas naves, e desviando dos asteroides, cada nave lhe concede um ponto, os três maiores placares ficam na tabela score.

2. Programação orientada a objetos

Segundo Santos (2011), Programação Orientada a Objetos ou, abreviadamente POO, é um paradigma de programação de computadores onde se usam classes e objetos, para representar e processar dados usando programas de computadores.

Para facilitar o entendimento, segundo Macoratti (2010?), o termo orientação a objetos significa organizar o mundo real como uma coleção de objetos que incorporam estrutura de dados e um conjunto de operações que manipulam estes dados.

3. Java FX

No site do Java (ORACLE) define que Java FX, é uma tecnologia de software que, ao ser combinada com Java, permite a criação e implantação de aplicações de aparência moderna e conteúdo rico de áudio e vídeo.

Ainda no site do Java destacamos a diferença do Java FX.

- Permite que os desenvolvedores integrem gráficos de vetor, recursos Web de animação, áudio e vídeo em uma aplicação rica, interativa e imersiva.
- Estende a tecnologia Java permitindo o uso de qualquer biblioteca Java em uma aplicação JavaFX.
- Permite um fluxo de trabalho eficiente de designer para desenvolvedor, no qual os designers podem trabalhar com suas ferramentas preferidas, em colaboração com desenvolvedores

3.1 Jogo em Java FX

O jogo conta com duas classes principais, a classe TelaInicial, responsável pela tela inicial do jogo, e a classe GameView que é a classe do jogo rodando.

Na imagem a baixo está o método que cria a escolha das naves.

```
196     private void createPickShip() {
197         playScene = new SpaceScene();
198         mainPane.getChildren().add(playScene);
199 
200         try {
201             tf.setFont(Font.loadFont(new FileInputStream(new File("src/Imagem/kenvector_future.ttf")), 13));
202         } catch (FileNotFoundException ex) {
203             Logger.getLogger(TelaInicial.class.getName()).log(Level.SEVERE, null, ex);
204         }
205         tf.setLayoutX(102);
206         tf.setLayoutY(305);
207         playScene.getPane().getChildren().add(tf);
208         infoLabel infLabel = new infoLabel("Escolha sua Nave");
209         infLabel.setLayoutX(25);
210         infLabel.setLayoutY(25);
211 
212         infoLabel inf = new infoLabel("SEU NOME");
213         inf.setLayoutX(20);
214         inf.setLayoutY(250);
215         playScene.getPane().getChildren().add(inf);
216         playScene.getPane().getChildren().add(infLabel);
217         playScene.getPane().getChildren().add(pickerShipToPlay());
218         playScene.getPane().getChildren().add(buttonStarGame());
219     }
```

Figura 1- Método Escolha da Nave.

O método acima insere, um campo de texto, e os label que trazem as mensagens para escolher a nave e inserir o nome, e por fim nas linhas 215 até a linha 218 é setado no painel. Como podemos ver na linha 217 é chamada a função onde é responsável por escolher a imagem da nave a ser escolhida, como podemos ver na figura 2.

```
199     private HBox pickerShipToPlay(){
200         HBox hbox = new HBox();
201 
202         hbox.setSpacing(20);
203         listShip = new ArrayList<>();
204         for (SHIP ship : SHIP.values()) {
205             ShipPick pick = new ShipPick(ship);
206             hbox.getChildren().add(pick);
207             listShip.add(pick);
208             pick.setOnMouseClicked((MouseEvent event) -> {
209                 for (ShipPick shipPick : listShip) {
210                     shipPick.setCircleTrueOrFalse(false);
211                 }
212                 pick.setCircleTrueOrFalse(true);
213                 shipChoose = pick.getShip();
214             });
215         }
216         hbox.setLayoutX(300-(118*2));
217         hbox.setLayoutY(100);
218         return hbox;
219     }
```

Figura 2- Método Seleciona Imagem da nave.

Figura 2 mostra a forma na qual é organizado a caixa onde é colocado as imagens da nave, ele verifica onde qual nave foi selecionada, e chama um enum chamado SHIP para retornar a imagem da nave.

Outra função importante, responsável por carregar e mostrar a tabela de pontuação, como pode ver na figura 3.

```
234     private void listaScore(){
235         scoreScene = new SpaceScene();
236         mainPane.getChildren().add(scoreScene);
237         GameView gView = new GameView();
238         gView.lerScore();
239          try {
240            gView.scoreNome = gson.lerNome();
241            gView. scoreVetor = gson.lerScore();
242         } catch (IOException ex) {
243             Logger.getLogger(GameView.class.getName()).log(Level.SEVERE, null, ex);
244         }
245 
246         infoLabel infLabel = new infoLabel("TOP PLAYERS");
247         infLabel.setLayoutX(25);
248         infLabel.setLayoutY(25);
249         scoreScene.getPane().getChildren().add(infLabel);
250         infoLabel vetInf = new infoLabel("");
251 
252         for (int i = 1; i < gView.scoreNome.length+1; i++){
253             vetInf = new infoLabel(i+"ª "+gView.scoreNome[i-1] + ": "+Integer.toString(gView.scoreVetor[i-1]));
254             vetInf.setLayoutX(25);
255             vetInf.setLayoutY(i*100);
256             scoreScene.getPane().getChildren().add(vetInf);
257         }
258 
259     }
```

Figura 3- método que insere a lista dos top 3 na cena score.

Na figura 3, o método chama as funções para ler o json onde está o nome e os pontos dos 3 maiores pontuadores, onde é lido e passado para o vetor na classe gameView, e exibido na aba score, por meio do laço de repetição.

Já na classe GameView, onde o jogo roda, um método essência para a mecânica do jogo é a criação da vida do personagem, na figura 4 podemos ver a função responsável por essa função.

```
205     public void CreateLife(LIFE lifeShip){
206         playerLife = 2;
207         lifeS = new ImageView(lifeScore);
208         setPosition(lifeS);
209         gamePane.getChildren().add(iVpoint);
210         gamePane.getChildren().add(lifeS);
211         score = new ScoreLabel("Points : 00");
212         score.setLayoutX(870);
213         score.setLayoutY(20);
214         gamePane.getChildren().add(score);
215         playerLifeArray = new ImageView[5];
216 
217         for (int i = 0; i < playerLifeArray.length; i++) {
218             playerLifeArray[i] = new ImageView(lifeShip.getUrlLife());
219             playerLifeArray[i].setLayoutX(850+(i*50));
220             playerLifeArray[i].setLayoutY(80);
221             gamePane.getChildren().add(playerLifeArray[i]);
222 
223         }
```

Figura 4- método criar vida, nele é criada as 3 vidas do personagem.

Esse método é responsável por criar as 3 vidas, o laço na linha 217 é responsável por setar a imagem da vida na tela e no local certo. Os pontos também são lançados no painel nesse local, como podemos ver na linha 214.

Na figura 4 foi apresentado o método que cria a vida, já na figura 5 está o método que retira a vida do personagem.

```
225     private void removeLife(){
226         gamePane.getChildren().remove(playerLifeArray[playerLife]);
227         playerLife--;
228         if(playerLife < 0){
229             gson.verificandoVetor(scoreVetor, scoreNome, nomePlayer, points);
230             gameStage.close();
231             gameTimer.stop();
232             menuGame.show();
233         }
```

Figura 5- remover a vida do personagem.

O método acima retira a vida do personagem até chegar em um, com a condição da linha 228 é verificada se a vida está em zero, se for verdadeira a afirmação é salvada os pontos do personagem chamando a função na classe gson, e fechado o jogo e por fim chamando a tela inicial.

Na imagem a baixo vamos ver duas funções, esses métodos funcionam de forma conjunta, sendo o método para calcular a colisão, e o método que verifica se a colisão foi detectada.

```
236     private double calculatorDistance(double x1, double x2, double y1, double y2){
237         return Math.sqrt(Math.pow(x1-x2, 2)+ Math.pow(y1-y2, 2));
238     }
239     private void collisionDetected(){
240         if(shipDetector + lifeDetector > calculatorDistance(ship.getLayoutX()+55, iVpoint.getLayoutX()+30,
241                 ship.getLayoutY()+45, iVpoint.getLayoutY()+30)){
242             setPosition(iVpoint);
243             points++;
244             String textToset = "POINTS: ";
245             if(points < 10){
246                 textToset += "0";
247             }
248
249             score.setText(textToset + points);
250         }
            for (int i = 0; i < asteroidArray.length; i++) {
252            if(shipDetector + asteroidDedetector > calculatorDistance(ship.getLayoutX() +49,
                    asteroidArray[i].getLayoutX() +20, ship.getLayoutY() +45,  asteroidArray[i].getLayoutY()+25)){
254                removeLife();
255                setPosition(asteroidArray[i]);
256            }
257
258         }
259     }
```

Figura 6- função calcula a distância, e função vê se objeto colidiu com a nave.

Os o método collisionDetected é responsável por verificar se a colisão existiu, a primeira colisão verificada é dos pontos que vão caindo pelo mapa, e a segunda é dos asteroides, se confirmada a colisão é retirada uma vida do personagem.

4. Resultados e Discussões

Por fim foi concluído o jogo de nave, usando a IDE Netbeans e escrito em JavaFX, usando várias classes do Java, mas sendo a principal usada foi a Scene, com ela foi montado toda a parte visual gráfica do jogo, sendo assim a tela inicial do jogo ficou dessa maneira.

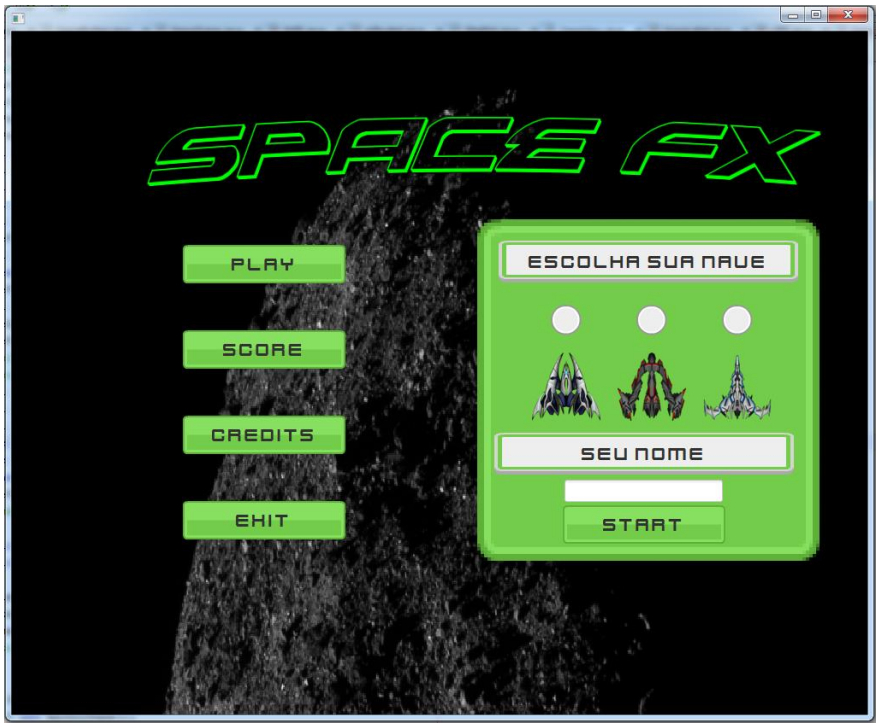

Figura 7- Tela Inicial do game.

Já na tela principal e clicando no play, aparece o painel onde tem que escolher a sua nave e digitar seu nome, clicando no start o jogo é iniciado. Como na figura 8 que está abaixo.

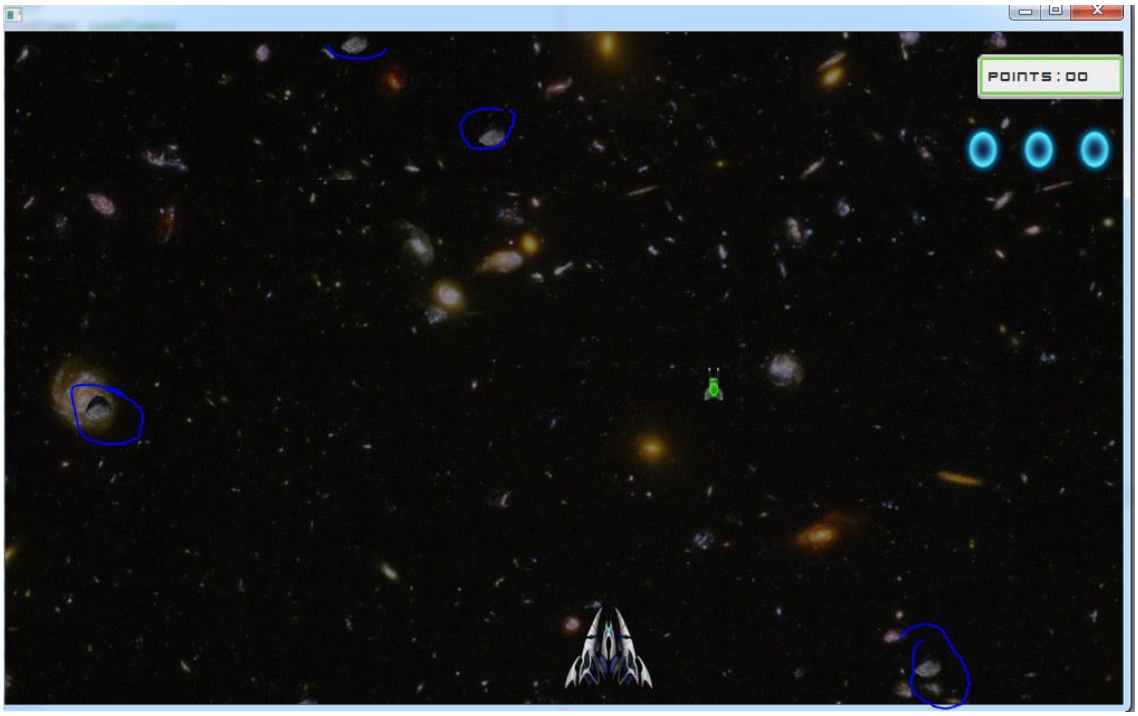

Figura 8- Cena do jogo, círculos azul auxiliam na visualizações dos asteroides.

Cena do jogo rodando, o jogo tem como objetivo esquivar dos asteroides e pegar as pequenas naves verde, cada nave vale um ponto, as três maiores pontuações ficam salvo na tabela score, na tela inicial.

4. Conclusão

Por fim, pode-se concluir com este trabalho, que o Java, tendo o foco em JavaFx é uma linguagem bem completa para a utilização em aplicações gráficas, várias bibliotecas uteis para a criação gráfica com design moderno. Tal feito foi usado a IDE Netbeans. Por sua vez o projeto ajudou a compreender melhor o significado de orientação a objetos e as ferramentas usadas para a criação de um jogo, nesse caso o desenvolvimento do jogo Space FX.

5. Referências

SANTOS, Rafael. **Introdução à Programação Orientada a Objetos usando Java**. 2011. Disponível em: <https://ramonrdm.files.wordpress.com/2011/09/java-orientado-a-objetos.pdf>. Acesso em 04 de dez. 2019.

MACORATTI, José Carlos. **O que significa Orientação a objetos ?** Disponível em: <http://www.macoratti.net/oo_conc2.htm>. Acesso em: 04 dez. 2019.

ORACLE. **Overview (JavaFX 8).** Disponível em: <https://docs.oracle.com/javase/8/javafx/api/overview-summary.html>. Acesso em: 04 dez. 2019.