Vogons, the Invaders

Pedro Henrique dos Santos Kutni
Instituto Federal Catarinense
p.henriquekutni@hotmail.com

Gustavo Danielevig Pereira Dias
Instituto Federal Catarinense
gustavo_dpdias@hotmail.com

Josef Mathaus Bischof
Instituto Federal Catarinense
theobischof@hotmail.com

Vinícius Feldhaus Gobetti
Instituto Federal Catarinense
vinigs2.gobetti@gmail.com

Adriano Pessini
Instituto Federal Catarinense
adriano.pessini@ifc.edu.br

Luiz Ricardo Uriarte
Instituto Federal Catarinense
luiz.uriarte@ifc.edu.br

ABSTRACT

Game development is a recognized and documented approach to improving computer programming learning. This article describes the redesign of the popular game Space Invaders, using Python as the programming language and libraries Arcade, Random, Os and Math. The game developed has modifications in various aspects, like textures, jogability, scores and story. The motivation for the remake was the desire to bring a nostalgic experience for the player, combined with learning about game development. Considering that it was built by people with no previous experience in game development, the proposed objectives were achieved, with the game having been completed. The Arcade library proved to be easy to use and productive. As future work, an online ranking will be developed, in addition to generating bonuses for the player and the use of design patterns.

KEYWORDS

Python, Arcade, Desenvolvimento De Jogos

1 Introdução

Há bastante tempo os jogos são uma diversão e um entretenimento para as pessoas do mundo, independente da faixa etária, pois existem jogos de estilos variados para os mais diversos gostos pessoais, destacando-se, entre eles, os jogos eletrônicos. Os jogos eletrônicos se tornaram populares a partir da década de 70/80, e com isso muitos jogos do gênero arcade também ficaram muito conhecidos, como, por exemplo: Ms. Pacman, Donkey Kong e Aero Fighters [1].

O desenvolvimento de jogos eletrônicos sempre contou com programadores para desenvolver os códigos que dão a vida para o jogo. Com o avanço das linguagens de programação e engines de desenvolvimento de jogos, o seu desenvolvimento se tornou mais simples e permitiu que fosse possível a criação de um jogo feito por amadores da área da programação [2].

Com o mercado mundial de jogos digitais se tornando um dos mais lucrativos dos últimos tempos [4], houve uma busca para aplicar algum redesign em jogos populares do século passado, alterando elementos gráficos ou sua jogabilidade. Inspirando-se no jogo "Space Invaders", foi aplicado esse intuito de redesign e houve o começo da criação do jogo "Vogons, the invaders".

O presente trabalho está dividido em 4 seções: na primeira é metodologia usada no desenvolvimento do jogo; a segunda apresenta o desenvolvimento do jogo; a terceira seção apresenta os resultados obtidos com esse desenvolvimento e na quarta seção apresenta-se as considerações finais do trabalho.

2 Metodologia

Na primeira etapa houve a pesquisa de maneiras viáveis para se desenvolver um jogo, sendo que ao longo do estudo foi escolhido o Python como linguagem de programação por conta do seu grande crescimento nos últimos anos e sua simplicidade para leigos na área
[3].

Na segunda etapa, inspirado no redesign do jogo em "Space Invaders", foi elaborado o projeto do jogo onde foram estabelecidas informações relevantes como: design do jogo, temática, personagens, enredo, entre outros. Também foi decidido pelo uso da biblioteca de criação de jogos Arcade, pois a mesma tem uma ótima usabilidade, funções prontas e com todos os atributos para o desenvolvimento de um jogo.

A terceira etapa se baseia no estudo do Python juntamente com as bibliotecas Arcade, Random, Os e Math, onde buscou-se o aprendizado e aperfeiçoamento nos mesmos.

A quarta etapa foi destinada à criação da arte do jogo através do AsePrite (editor de Pixel Art) e Photoshop (editor de imagens) e a codificação do jogo foi realizada usando o editor de textos

3 Desenvolvimento

O código do jogo foi feito inteiramente orientado à objeto, com o desenvolvimento das classes demonstradas na Figura 1 que contém toda a lógica de sua funcionalidade, sendo elas: Jogo, Tiro, Nave, Base, Nave_Inimiga e Boss.

Estas classes são especializações da classe Sprite do Arcade que provê recursos de manipulação dos sprites.

Na Figura 1 pode ser observada a imagem das naves utilizadas no jogo, sendo nave inimiga (em 1), Boss (em 2) e a nave do jogador (em 3). As imagens foram elaboradas pelos autores, seguindo as cores e a temática do jogo.

Conforme pode ser observado na Figura 2, a execução de um jogo respeita duas etapas bem definidas: a inicialização e a lógica de jogo, também conhecida como game looping.

Figura 1 – Naves do jogo. *Fonte: Autores, 2019.*

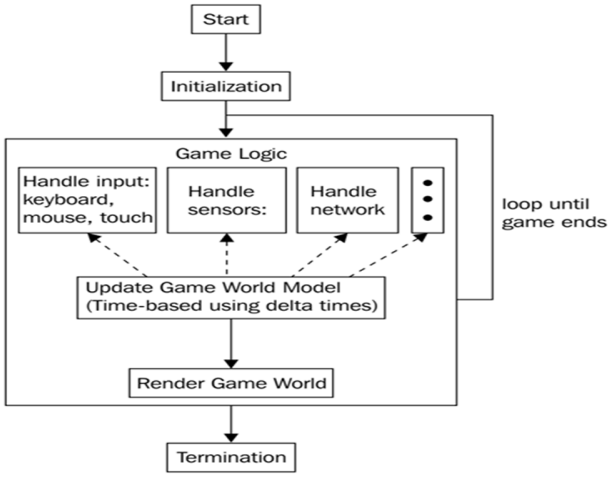

Figura 2 - Processo de inicialização de um jogo. *Fonte: Nair e Oehlke, 2015. [5].*

Na etapa de inicialização, o programador deve realizar o carregamento das texturas, animações e áudios, o posicionamento do jogador e do jogo e a definição do estado inicial do jogo. No Arcade, a inicialização é realizada por meio da implementação de um método chamado setup(), conforme pode ser observado na Figura 3.

```
87
88      def setup(self):
89          """
90          Executado de maneira manual assim que a classe é chamada
91          """
92          # Adcionar os sprites a lista de sprites
93          self.lista_nave = arcade.SpriteList() #sprite da nave
94          self.lista_inimigo = arcade.SpriteList() #sprite da nave inimiga
95          self.lista_tiro = arcade.SpriteList() #sprite do tiro
96          self.lista_boss = arcade.SpriteList() #sprite da nave inimiga
97          self.tiro_list_boss = arcade.SpriteList() #sprite do tiro do boss
98
99
100         #definindo a imagem de fundo
101         self.background = arcade.load_texture(DIRETORIO_ATUAL +
102                                               "/img/fundo.jpg")
103
104         # Definindo a nave
105         self.player_sprite = Nave(DIRETORIO_ATUAL + "/img/nave.png",
106                                   TAMANHO_DA_NAVE,
107                                   arcade.key.LEFT, arcade.key.RIGHT)
108                                   #definindo a imagem da nave
109         #imagem da nave sendo adcionada na lista de sprites
110         self.lista_nave.append(self.player_sprite)
111
```

Figura 3 - Carregamento de textura utilizando Arcade. *Fonte: Autores, 2019.*

A lógica do jogo (também chamada de game looping) é programada por meio da implementação do método update() que, quando chamada, é executada automaticamente. Neste projeto, este método foi usado para funções como: a movimentação do jogador, controlador da pontuação, spawn da nave inimiga, entre outros.

A Figura 4 mostra um trecho da codificação do jogo, essa parte do código é responsável por fazer a geração aleatória dos inimigos no topo da tela de jogo, porém limitando a posição no eixo Y em 400 pixels para garantir uma distância de segurança para o jogador. Para fazer a geração aleatória foi usada a biblioteca Random, onde a mesma gera números aleatórios entre aqueles definidos na codificação que são usados no posicionamento dos inimigos nos

```
348
349     while self.QUANTIDADE_DE_INIMIGOS > self.contador_inimigo:
350         #gera a quantidade de inimigos definidos
351         nave_inimiga = Nave_inimiga(DIRETORIO_ATUAL +
352                                     "/img/nave_inimiga.png",
353                                     TAMANHO_DA_NAVE_INIMIGA)
354         #definindo a imagem da nave inimiga
355
356         sobreposicao = [1] #lista para ser comparada com lista
357         while len(sobreposicao) !=0:
358             #posição dos inimigos(Vão ser adcionados aleatoriamente)
359             nave_inimiga.center_x = random.randrange(SCREEN_WIDTH)
360             nave_inimiga.center_y = random.randrange(480,
361             SCREEN_HEIGHT)
362             #verifica para não nascer duas naves juntas
363             sobreposicao = arcade.check_for_collision_with_list(nave_inimiga,
364                                                                 self.lista_inimigo)
365         #imagem da nave inimiga sendo adcionada na lista de sprites
366         self.lista_inimigo.append(nave_inimiga)
367         #contador para não gerar mais do que foi definido
368         self.contador_inimigo += 1
369
```

Figura 4 - Geração de inimigos na tela. *Fonte: Autores, 2019.*

Na Figura 5 mostra-se como é feito a mira do tiro do boss, cujo propósito é sempre seguir o jogador. Para se fazer essa "mira guiada" foram utilizados conhecimentos matemáticos, no caso o arco tangente do coeficiente dos parâmetros sendo eles o eixo x e y (ângulo teta).

```
415     for boss in self.lista_boss:
416
417         start_x = boss.center_x #recebe posição do boss (eixo x)
418         start_y = boss.center_y #recebe posição do boss (eixo y)
419
420         dest_x = self.player_sprite.center_x #recebe posição no player (eixo x)
421         dest_y = self.player_sprite.center_y #recebe posição no player (eixo y)
422
423         x_diff = dest_x - start_x #Calcula a diferença (eixo x)
424         y_diff = dest_y - start_y #Calcula a diferença (eixo y)
425
426         #calculo do angulo
427         angle = math.atan2(y_diff, x_diff)
428         boss.angle = math.degrees(angle)-90
```

Figura 5 - Implementação da "mira guiada". *Fonte: Autores, 2019.*

A movimentação da nave inimiga é feita na sua própria classe, como pode ser observado na figura 6. A movimentação da nave inimiga é feita somente no eixo Y, com uma velocidade de -1 pixel por frame.

```
12      def update(self):
13          self.center_y -= 1
```

Figura 6- Codificação do movimento da nave inimiga. *Fonte: Autores(2019)*

Já a movimentação do jogador é feita em dois momentos para separar as atribuições e responsabilidades de acordo com os princípio da orientação a objetos: primeiramente na classe Jogo (Figura 7) e logo após na classe Nave (figura 8), pois

das teclas de movimentação pressionadas pelo jogador é realizada na segunda, enquanto que a primeira agrupa as funcionalidades relacionadas à lógica do jogo que envolvem a nave do jogador, como por exemplo a verificação de colisão, entre outros.

```
287     def on_key_press(self, key, modifiers):
288
289         if self.vida == True:
290             for player in self.lista_nave:
291                 verificador = player.on_key_press(key, modifiers)
292                 if verificador is not None:
293                     self.lista_tiro.append(verificador)
```

Figura 7 - Movimentação do player na classe Jogo. *Fonte: Autores, 2019.*

```
20      def on_key_press(self, key, modifiers):
21
22          if key == self.movimentacao_esquerda:
23              self.change_x = -MOVIMENTACAO_DA_NAVE
24
25          if key == self.movimentacao_direita:
26              self.change_x = MOVIMENTACAO_DA_NAVE
```

Figura 8 - Movimentação do player na classe Nave. *Fonte: Autores, 2019.*

4 Resultados obtidos

Com o fim da codificação os resultados obtidos foram além do esperado, levando em consideração que foi construído por pessoas sem experiência no desenvolvimento de jogos. Confira abaixo as telas do jogo "Vogons, the invaders" em execução:

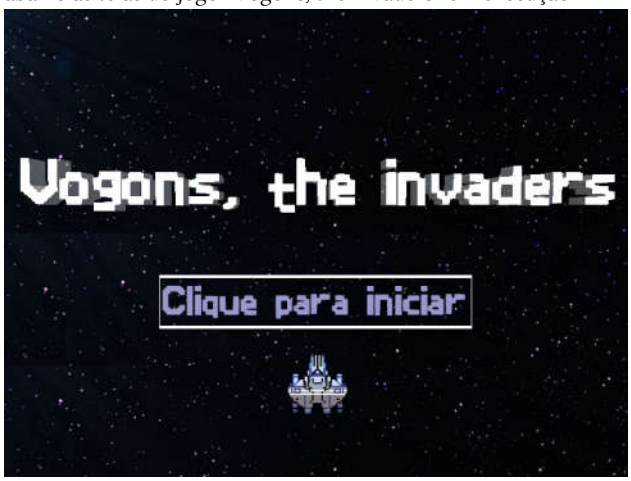

Figura 9 - Tela inicial do jogo. *Fonte: Nair e Oehlke, 2015.*

A tela de jogo é exibida na Figura 10, onde podem ser observados o tempo de jogo (em 1), a quantidade de disparos disponíveis para o jogador (em 2), a quantidade de hordas de inimigos que já foram geradas, o inimigo da classe Nave_Inimiga (em 4), o boss com o disparo guiado na direção do jogador (em 5), a nave do jogador (em
6), a imagem da base com o respectivo percentual de durabilidade restante (em 7, que se chegar a zero causa o encerramento por perda do jogo), a pontuação alcançada pelo jogador (em 8) e o

Figura 10 - Jogo em execução. *Fonte: Autores, 2019.*

5 Considerações finais

Diante do exposto, o presente trabalho mostrou o desenvolvimento de um jogo usando a linguagem de programação Python. O objetivo deste trabalho foi o enriquecimento de conhecimento pessoal na área da informática, mais precisamente em programação. A utilização da biblioteca Arcade proporcionou facilidade no desenvolvimento do jogo, sendo produtiva e trazendo diversão no decorrer do desenvolvimento do mesmo. Recomenda-se a utilização desta biblioteca para os iniciantes na área de programação, uma vez que é de fácil compreensão e acessível a todos. Vale observar que, apesar de ser de fácil utilização, o desenvolvimento de sistemas não deve se limitar a uma única biblioteca, visto o constante desenvolvimento da área de informática, principalmente no que diz respeito às ferramentas para produção de jogos. Como trabalhos futuros, estão previstas implementações de um ranking online dos jogadores, a geração de bônus para os jogadores que aumentem o número de vidas, velocidade, quantidade de disparos, entre outros. Outra possibilidade, como recomendação para trabalhos futuros, e a utilização de padrões de projeto durante o desenvolvimento.

REFERÊNCIAS

[1] Batista, M. L. S.; Quintão, P. L.; Lima, S. M. B.. Campos, L. C. D.,Batista, T. J. S. Revista Eletrônica da Faculdade Metodista Granbery, n 3,jul/dez 2007.
[2] Tiobe (2019) "TIOBE Index for October 2019", https://www.tiobe.com/tiobeindex/, Outubro 2019.
[3] Valente, J. A. e Almeida F. J. Visão analítica da informática na educação no Brasil: A questão da formação do professor. Revista Brasileira de Informática na Educação – Número 1 – 1997.
[4] Ximenes, M., Souza, L. J., Malcher, F., Marinho, L., Neves, A. eCampos, F. Redesign de Jogos Clássicos VII Brazilian Symposium on Computer Games and Digital Entertainment November, 10-12, 2008
[5] Nair, S. B. e Oehlke, A. Learning LibGDX Game Development. Second Edition-Packt Publishing, 2015