PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

PIRATAS VS MARINHA:
UM JOGO DE PERSEGUIÇÃO UTILIZANDO TEORIA DOS GRAFOS E PYGAME

DÉBORA BARBOSA BARRETO

GOIÂNIA
2021

DÉBORA BARBOSA BARRETO

PIRATAS VS MARINHA:
UM JOGO DE PERSEGUIÇÃO UTILIZANDO TEORIA DOS GRAFOS E PYGAME

Trabalho de Conclusão de Curso apresentado à Escola Politécnica, da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Me. Fernando G. Abadia

GOIÂNIA
2021

DÉBORA BARBOSA BARRETO

PIRATAS VS MARINHA:
UM JOGO DE PERSEGUIÇÃO UTILIZANDO TEORIA DOS GRAFOS E PYGAME

Este trabalho de Conclusão de Curso, julgado adequado para a obtenção do título de Bacharel em Ciência da Computação, e aprovado em sua forma final pela Escola Politécnica da Pontifícia Universidade Católica de Goiás, em __/__/____.

_____
Profa. Ma. Ludmilla Reis Pinheiro dos Santos
Coordenador(a) de Trabalho de Conclusão de Curso

Banca examinadora:

_____
Prof. Me. Fernando Gonçalves Abadia

_____
Prof. Dra. Carmen Cecília Centeno

_____
Prof. Me. Max Gontijo de Oliveira

GOIÂNIA
2021

Dedico essa monografia a Deus por sempre estar comigo. A minha avó Marlene (*in memorian*), pois ela sempre apoiou meus sonhos, e também ao meu avô Olintho (*in memorian*). Aos meus pais, Adriana e André, por se esforçarem tanto para eu ter uma boa educação e formação. A minha irmã Bárbara por sempre me incentivar e me apoiar em tudo e ao meu irmão Gabriel. Ao meu amigo Guilherme. E ao Little Mix, grupo que sou fã e admiro muito.

AGRADECIMENTOS

Quero agradecer ao meu orientador, Prof. Me. Fernando G. Abadia, que esteve nesse longo percurso comigo. Sempre me ajudando e sanando minhas dúvidas. Agradeço também à Profª. Ms. Carmen Cecilia Centeno por ter me apoiado durante a graduação.

RESUMO

O presente projeto almeja, principalmente, abordar problemas e sugerir soluções de caráter lúdico relacionadas com a Teoria dos Grafos. Considerada relativamente recente, a aludida Teoria, que surgiu no século XVIII e entrou nos programas do ensino secundário a partir do século XX, pode ser apontada como uma das aplicações mais interessantes dentre as teorias matemáticas voltadas para aplicações lúdicas. Além disso, uma das funções da Teoria dos Grafos consiste justamente na solução de problemas e na compreensão de jogos – razão pela qual foi adotada para o desenvolvimento do presente estudo. Em linhas gerais, neste trabalho, que aborda especialmente jogos de perseguição, foi desenvolvido um jogo em Pygame. Para que isso fosse possível, partiu-se da aplicação da teoria desenvolvida a partir de grafos, que demonstraram como o jogo de perseguição funciona na prática.

Palavras-chave: Grafos; Jogos de perseguição; Python; Pygame.

ABSTRACT

This project aims, mainly, to address problems and suggest solutions of a playful nature related to Graph Theory. Considered relatively recent, the aforementioned Theory, which emerged in the 18th century and entered secondary education programs in the 20th century, can be identified as one of the most interesting applications among mathematical theories aimed at recreational applications. Furthermore, one of the functions of Graph Theory consists precisely in solving problems and understanding games – which is why it was adopted for the development of this study. In general terms, in this work, which deals especially with chase games, a game in Pygame was developed. To make this possible, we started with the application of the theory developed from graphs, which demonstrated how the chase game works in practice.

Graphs; Pursuit Games; Python; Pygame.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Uma forma simples de definir grafos, é interpretá-los como uma estrutura que contém vértices que são ligados por arestas. Os vértices são representados por pontos e as arestas são representadas por segmento de retas, podendo ser curvos ou não. (ROSEN, 2009)

Apesar de todo conhecimento e teoremas atuais nessa área, a teoria dos grafos foi redescoberta muitas vezes, sendo esta estudada em diversas áreas separadas e, constantemente, os problemas estudados, de certa forma, apresentavam algumas características semelhantes. (NETTO, 2012)

O pioneiro nos estudos da teoria dos grafos foi Euler, em 1736. Ele propôs uma solução para o *problema da ponte de Königsberg*, a resposta desse problema foi considerada o primeiro teorema na teoria dos grafos. A solução foi proposta há mais de 200 anos, mas nos anos seguintes, não tiveram muitos trabalhos e estudos nessa área. (SZWARCFITER, 2018)

Os estudos de teoria dos grafos voltaram a aparecer em meados do século XIX, sendo que foram três estudos isolados que contribuíram para despertar o interesse pela área. As três pesquisas e soluções foram do *problema das quatro cores*, *problema do ciclo Hamiltoniano* e *teoria das árvores*.

De acordo com Szwarcfiter (2018), durante o século XX, o interesse pela teoria dos grafos só cresceu e ainda existem várias regiões inexploradas nessa área. Entretanto, segundo Rosen (2009). atualmente, os grafos já têm muitas utilidades como, por exemplo, grafos para mostrar relacionamentos, grafos de superposição de nichos em ecologia, grafos de influência, grafos de coloração e vários outros.

Um dos problemas em teoria dos grafos diz respeito a jogos de perseguição, como, por exemplo, Polícia e Ladrão, Interpol e Pandemic. Segundo Neufeld e Nowakowski (1998), nesses jogos, um jogador tenta capturar o outro enquanto eles se movimentam por um grafo não direcionado. Existem muitas variações de jogo, e em alguns podem existir mais de um perseguidor.

O Interpol é um jogo de tabuleiro e também utiliza o conceito de jogo de perseguição. Nesse jogo, o Mister X foge pelas ruas de Londre utilizando algum meio de transporte e ele é visto apenas algumas vezes em sua localização exata. Existem outros jogadores tentando descobrir onde ele está para poder pegá-lo.

No jogo de polícia e ladrão, um vértice é escolhido para o perseguidor e um vértice para quem está sendo perseguido. Os dois jogadores vão se revezando durante o jogo, cada jogador precisa se mover pelos vértices ou então eles podem permanecer no mesmo lugar. O perseguidor ganha se conseguir capturar quem está perseguindo, ocupando o mesmo vértice de quem foi procurado. (NEUFELD, NOWAKOWSKI, 1998)

Existem algumas variações no jogo e uma versão importante é a ativa do jogo. Nessa versão, o jogador perseguido deve se mover sempre que for sua vez de jogar. Porém, na versão passiva, esse jogador pode ficar na mesma posição se quiser. O jogador que está perseguindo deve se mover em ambas as versões. Os jogos de perseguição têm várias variações, mas, sempre, os jogadores devem jogar de acordo com regras determinadas. (AIGNER, FROMME, 1984)

Neste projeto, foram estudados alguns conceitos teoria dos grafos para poder compreender como os jogos de perseguição funcionam. Além disso, foi implementado um jogo de perseguição para melhor visualização e compreensão da teoria.

No jogo implementado para esse trabalho, existem dois jogadores, um navio da marinha perseguindo um navio pirata. Cada jogador deve estar em um vértice. O objetivo da tripulação de piratas é fugir da marinha e continuar a aventura deles, mas para alcançar esse objetivo, eles precisam pensar em uma estratégia para fugir da marinha. E o objetivo da marinha é capturar a tripulação dos piratas.

A marinha vence se capturar os piratas, se isso acontecer o jogo finaliza. Os piratas só vencem se a marinha não capturar o navio deles, porém esse é um jogo onde a marinha tem uma estratégia de vitória, portanto sempre vence.

No capítulo 2 será visto os conceitos de teoria dos grafos para entender como funciona os jogos de perseguição utilizando teoria dos grafos. Esses conceitos são desde o que é um grafo até o conceito de homomorfismo. Outros conceitos que foi abordado no capítulo 2 são árvores, ciclos, grafos conexos, coloração e produto de grafos. No capítulo 2 também foi abordado os jogos de perseguição.

No capítulo 3 mostra quais métodos e quais materiais e IDEs foram utilizadas para a escrita e desenvolvimento desse projeto. No capítulo 4 apresenta como foi o desenvolvimento do jogo, mostrando os conceitos básicos de Pygame, os pilares importantes na criação de um jogo e qual foi a lógica utilizada durante o

desenvolvimento do jogo. E o capítulo 5 é mostrado qual foi o resultado desse jogo desenvolvido.

2. FUNDAMENTAÇÃO TEÓRICA

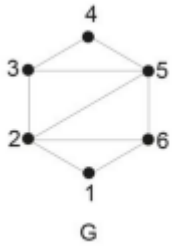
Nesse capítulo será abordado as principais definições e teoremas sobre a teoria dos grafos. Esses conceitos serão usados como base para a teoria de jogos de perseguição utilizando grafos. Além disso, serão abordados conceitos sobre jogos de perseguição utilizando a teoria dos grafos.

2.1. Estudo de Grafos

Seja G um grafo que contém dois conjuntos. Um é o conjunto de vértices V, um conjunto não vazio. E o outro conjunto é o de arestas E, esse conjunto pode ser vazio ou não. Quando um grafo possui um conjunto finito de vértices, ele é chamado de grafo finito e é chamado de grafo infinito quando possui com um conjunto de vértices infinito. (ROSEN, 2009)

Na figura 1 mostra um grafo onde nenhum dos conjuntos são vazios.

Figura 1 - Grafo com conjuntos não vazios

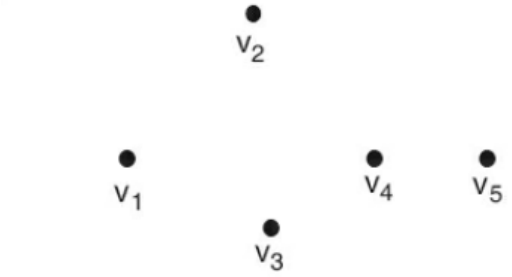


Fonte: SZWARCFITER (2018, P. 65)

De acordo com Nicoletti e Hruschka, são utilizados os termos ordem e tamanho na literatura dos grafos. Ordem indica o número de vértices de um grafo e tamanho, quando o grafo é não nulo, indica o número de arestas de um grafo.

Cada aresta em E é conectada a um par de vértices (u,v) em V, cada vértice é chamado de vértice-extremidade. Um conjunto E de arestas pode ser vazio, nesse caso o grafo é chamado de grafo nulo. Na figura 2 é apresentado um grafo nulo. (NICOLETTI; JR., 2018)

Figura 2 - Grafo Nulo

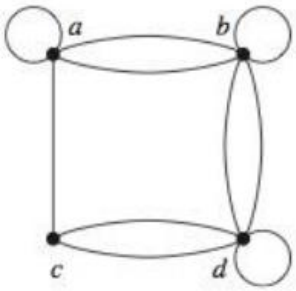


Fonte: NICOLETTI; JR. (2018, P. 40)

Os vértices extremos de uma aresta são adjacentes. Além de vértices adjacentes, existem as arestas adjacentes que são arestas que possuem um vértice em comum. Porém, se duas ou mais arestas possuem os mesmos vértices, elas são arestas paralelas. (NICOLETTI; JR., 2018)

Segundo Rosen, algumas arestas podem ser laços, isso acontece se uma aresta for usada para conectar um vértice nele mesmo. Os grafos que contém esses laços são chamados de pseudografos. A figura 3 apresenta um exemplo de grafo que contém laços.

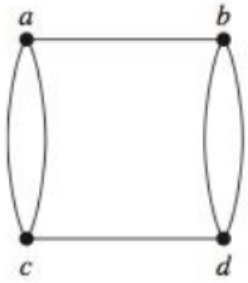
Figura 3 - Grafo com laços



Fonte: ROSEN (2009, P. 596)

Os grafos podem ser classificados como grafos simples. Um grafo é considerado simples se cada aresta se conecta a dois vértices diferentes e se duas arestas diferentes nunca conectam ao mesmo par de vértice. Um grafo pode ser um multigrafo se ele tiver arestas múltiplas que conectam os mesmos vértices. A figura 4 apresenta um multigrafo, onde os pares de vértices (a,c) e (b,d) são conectados por duas arestas. (ROSEN, 2009)

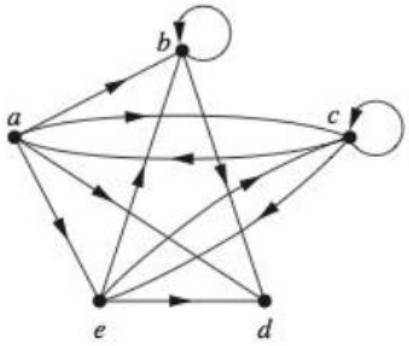
Figura 4 - Multigrafo



Fonte: ROSEN (2009, P. 596)

Um grafo pode ser não orientado se suas arestas são não orientadas, se não possuem direção. Um grafo orientado pode ser chamado de dígrafo e ele possui um conjunto não vazio de vértices e um conjunto de arestas orientadas, que são chamadas de arcos. Essas arestas são associadas a um par ordenado de vértices. Uma aresta que está associada ao par ordenado (*u, v*) se começa em u e termina em *v*. Quando um grafo possui arestas orientadas e arestas não orientadas, ele é chamado de grafo misto. A figura 5 mostra um grafo orientado, onde as arestas mostram a direção dos vértices. (ROSEN, 2009)

Figura 5 - Grafo orientado



Fonte: ROSEN (2009, P. 612)

Um grafo orientado é considerado um grafo orientado simples se ele não possui laços e nem arestas orientadas múltiplas. E pode ser um multigrafo orientado se possuir arestas orientadas múltiplas. (ROSEN, 2009)

2.1.1 CAMINHOS E CICLOS

Nos grafos aparecem algumas sequências, como exemplo: passeio, caminho e ciclos.

De acordo com Goldbarg, um passeio em um grafo é uma “sequência finita de vértices e arestas x_0, a_1, x_1, a_2,..., x_{k-1}, a_k, x_k.” Sendo que essa sequência precisa começar e terminar com vértices onde x_1-1 e x_i são vértices terminais da aresta a_i, $1 \leq i \leq k$.

Seja um passeio P, sendo que P = v_1, ..., v_k. O vértice v_1 é chamado de origem e o vértice v_k é chamado de término e ambos são chamados de extremos de P. E são chamados de vértices internos de P, os vértices v_2, ..., v_{k-1}. O comprimento de P é quantidade de arestas nessa sequência. (MOTA; 2019)

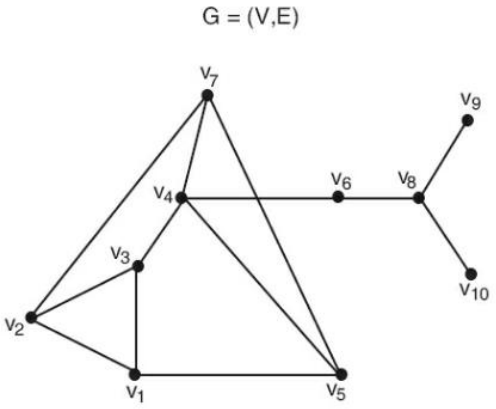
Um passeio é considerado fechado se o seu comprimento é não nulo e se o vértice de origem é o mesmo que o vértice de término. Caso contrário, é um passeio aberto. (MOTA; 2019)

Além do passeio, existem dois subgrafos que são necessários citar. São os trajetos e os caminhos. Um trajeto de um grafo é uma sequência finita de vértices e arestas e deve sempre começar e terminar por vértices. Em um trajeto, cada aresta aparece apenas uma vez e deve ser incidente ao vértice que precede e que sucede. Um caminho é um trajeto, porém não pode haver repetição de vértices em um caminho. (RANGEL; OLIVEIRA; ARAUJO, 2018)

Assim como um passeio, também existem trajetos e caminhos fechados. Um trajeto fechado é um trajeto em que o vértice final e o inicial são iguais. E um caminho fechado é um trajeto em os únicos vértices que aparecem mais de uma vez é o inicial e o final. Um caminho fechado também pode ser chamado de ciclo. (RANGEL; OLIVEIRA; ARAUJO, 2018)

Na figura 6 é apresentado um grafo que contém passeios, trajeto, caminhos e ciclos.

Figura 6 - Passeios, trajetos, caminhos e ciclos



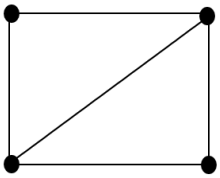
Fonte: SZWARCFITER (2018, P. 80)

De acordo com Szwarcfiter, os exemplos de passeios, trajeto, caminhos e ciclos que aparecem nesse grafo são:

- Passeios: $v_1\ v_3\ v_2\ v_1\ v_3\ v_2$; $v_7\ v_4\ v_5\ v_7\ v_4\ v_6$
- Trajetos: $v_2\ v_1\ v_5\ v_7\ v_2\ v_3\ v_4$; $v_1\ v_3\ v_2\ v_1\ v_5\ v_4$
- Caminhos: $v_1\ v_3\ v_4\ v_6\ v_8$; $v_1\ v_3\ v_2\ v_7\ v_4\ v_6$
- Ciclos: $v_5\ v_2\ v_1\ v_7\ v_5$; $v_4\ v_5\ v_1\ v_2\ v_3\ v_4$

Utilizando o conceito de caminhos, é possível definir outro tipo de grafo muito importante, os grafos conexos. De acordo com Oliveira, Rangel e Araujo, um grafo conexo é um grafo onde existe um caminho entre cada par de vértice do grafo. Se não existir esse caminho, o grafo é desconexo. Na figura 7 está um exemplo de grafo conexo.

Figura 7 - Grafo Conexo



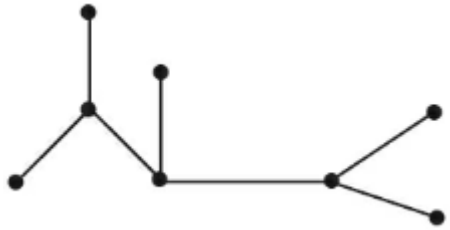
Fonte: autoria própria

2.1.2 ÁRVORES

Um grafo é considerado uma árvore se ele for um grafo acíclico e conexo. Se em uma árvore, T(*V,E*), o vértice *v* possuir grau ≤ 1 então ele é uma **folha**. Porém, se o vértice v tiver grau > 1, ele é um **vértice interior**. (SZWARCFITER, 2018)

A figura 8 apresenta um exemplo de árvore.

Figura 8 - Árvore



Fonte: NICOLETTI; JR. (2018, P. 117)

TEOREMA 1 [SZWARCFITER]: "*Um grafo G é uma árvore se e somente se existir um único caminho entre cada par de vértices de G.*"

2.2. Jogos de Perseguição Utilizando Grafos

De acordo com Isaacs (1999), os jogos de perseguição podem ter várias versões, como navio e submarino, míssil e bombardeio, tanque, jipe, polícia e ladrão, a princesa e o monstro, entre outros.

Seja um perseguidor P e um evasor E, os personagens podem se mover utilizando diversos jeitos e meios de transporte. Em versões mais simples, existem dois personagens. Porém, em versões mais complexas podem existir mais de dois personagens, para se opor as tropas inimigas. P e E podem representar suas tropas quando existem mais de dois jogadores, mas, quando são apenas, eles representam os veículos utilizados para a movimentação. (ISAACS, 1999)

Geralmente, os jogos de perseguição acabam quando o perseguidor consegue capturar o evasor. Isso acontece quando a distância entre eles se torna menor que uma quantidade positiva prescrita.

Para elaborar um jogo de perseguição e suas estratégias, é necessário pensar em duas coisas. Primeiro, é necessário pensar qual é a melhor maneira de P capturar E. Sabendo que P conhece sua localização e a localização de E, como P deve regular sua direção e posição? E a segunda coisa é, decidir o que significa melhor. Falando em termos de teoria dos jogos, deve-se escolher uma recompensa. Por exemplo, um critério realista pode ser capturar o evasor antes do tempo esperar ou então o evasor conseguir atingir seu objetivo, diga-se que pode ser bombardear o alvo determinado.

Um exemplo de jogos de perseguição, temos o Interpol da Grow mostrado na **Erro! Fonte de referência não encontrada.**, onde um ladrão muito procurado, chamado Mister-X, está fugindo e os detetives devem capturá-lo. De acordo com o site Ludopedia, nesse jogo, o Mister-X mostra sua localização apenas nas movimentações 3, 8, 13 e 18. E os detetives sempre mostram sua localização.

Figura 9 - Jogo Interpol da Grow

Fonte: ludopedia.com.br (2021)

No início do jogo Interpol, cada jogador pega uma ficha de início de partida e o número da ficha é o ponto de partida do jogador. Para os jogadores se movimentarem é necessário utilizar bilhetes de táxi, metrô e ônibus. Entretanto, quando os detetives usam um bilhete, esse mesmo deve ser transferido para o Mister-X. Então, em algum momento do jogo, pode acontecer dos detetives ficarem sem bilhetes e o Mister-X continuar com vários bilhetes. Por isso é necessário que os detetives pensem em uma estratégia para continuarem se movimentando durante o jogo.

No jogo é possível usar movimentos especiais. O Mister-X pode jogar duas vezes seguidas utilizando a ficha de movimento duplo. Ele também possui bilhetes negros e quando ele os usa, os detetives não sabem qual meio de transporte o Mister-X utilizou. Os detetives podem utilizar um helicóptero em três rodadas, porém só pode

ser utilizado um helicóptero em uma rodada. Eles podem bloquear estações para o Mister-X não passar por ela.

Os detetives ganham se um detetive ocupar a mesma estação que o Mister-X e ele ganha se os bilhetes dos detetives acabarem ou se ele fugir dos detetives por 24 rodadas e eles não conseguirem capturar o Mister-X.

Outro jogo de perseguição, porém que não utiliza grafos, é o Subway Surfers. Nesse jogo, um jogador está sendo perseguido por um inspetor. Existem três pistas e sempre que o jogador encontra um obstáculo é necessário mudar de pista para não ser capturado pelo inspetor. Esse jogo só acaba quando o personagem principal esbarra em um obstáculo e é capturado. Entretanto, é possível que ele quebre recordes de corrida.

2.2.1 JOGOS DE PERSEGUIÇÃO EM GRAFOS

Polícia e ladrão é um jogo de perseguição aplicado em um grafo não direcionado finito e conexo, seja esse grafo G = (V,E). O jogo pode ter 2 ou mais jogadores, mas sempre será dois conjuntos, o de policiais e o do ladrão. (NEUFELD, NOWAKOWSKI, 1998)

O jogador 1, o policial, começa escolhendo um vértice. Em seguida, o jogador 2, o ladrão, escolhe um vértice. Durante o jogo, eles vão revezando as jogadas. A cada jogada, um jogador se desliza pelas arestas para vértices adjacentes.

Esse jogo tem muitas versões possíveis. Duas versões são a passiva e a ativa. A versão passiva permite que o ladrão não seja obrigado a se mover na sua vez de jogar. E na versão ativa, o ladrão é obrigado a se movimentar quando for sua vez de jogar. O policial deve se mover na sua rodada nas duas versões.

O policial vence se ele conseguir ocupar o mesmo vértice que o ladrão está. E o ladrão ganha se conseguir evitar essa situação.

Em cada grafo G existe uma estratégia vencedora para algum jogador. Por exemplo, se a estratégia for para o policial então ele deve conseguir pegar o ladrão. O grafo é dito *cop-win* quando a estratégia vencedora é do policial e é dito *robber-win* quando o ladrão tem a estratégia vencedora. (AIGNER, FROMME, 1984)

Um grafo *cop-win* óbvio são as árvores. Segundo Aigner e Fromme, o vértice ocupado pelo policial divide a árvore em 2 componentes. E sempre que o policial se move pelo caminho único em direção ao ladrão, pelo menos um vértice é reduzido.

Na figura 10 mostra um exemplo de um grafo com estratégia *cop-win* óbvia. O grafo é acíclico e é conexo, portanto, é uma árvore.

Figura 10 - Grafo cop-win

Fonte: SZWARCFITER (2018, P. 71)

Por exemplo, considere que nessa árvore na figura 11 o policial é o vermelho e o ladrão é o verde.

Figura 11 - Jogo de perseguição em árvore

Fonte: autoria própria

O policial irá se mover e a única opção dele é para a direita. Depois é a vez do ladrão jogar, ele pode continuar no mesmo vértice ou se mover. Suponha que o ladrão se mova para a direita e durante as próximas rodadas, os dois jogadores vão se movendo para a direita. Em um momento, as posições dos jogadores ficarão como na figura 12.

Figura 12 - Final do jogo de perseguição em árvore

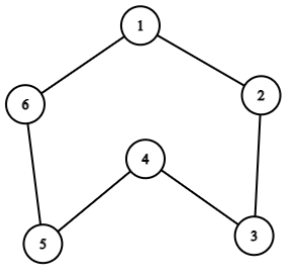
Fonte: autoria própria

Na figura x, o ladrão não tem nenhuma opção de movimento então será obrigado a ficar no mesmo vértice. Na rodada que o policial joga, ele irá para a direita e ocupará o mesmo vértice que o ladrão. Sendo assim, o policial vence.

Nas árvores, em alguma rodada, não terá mais uma saída para o ladrão então o policial sempre conseguirá ocupar o mesmo vértice que o ladrão em algum momento.

E um grafo com ciclo de comprimento de pelo menos 4 é um grafo *robber-win* óbvio, pois o ladrão pode sempre se posicionar a uma distância 2 do policial. Na figura 13 é mostrado um grafo com estratégia *robber-win* óbvio pois possui 6 vértices e é um ciclo.

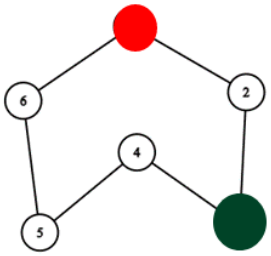
Figura 13 - Ciclo



Fonte: autoria própria

Na figura 14, o policial é o vermelho que ocupa o vértice 1 e o ladrão é o verde que ocupa o vértice 3.

Figura 14 - Jogo de perseguição em ciclo

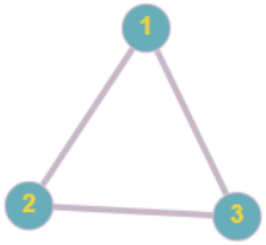


Fonte: autoria própria

O policial começa jogando. Suponha que ele se mova para o vértice 2 e, após essa jogada, o ladrão se mova para o vértice 4. A cada rodada, os jogadores se movimentam para o próximo vértice, porém, a distância entre eles continua a mesma. Isso gera um *looping* e o policial nunca irá ocupar o mesmo vértice que o ladrão. Sendo assim, o grafo é *robber-win*.

Entretanto, em um grafo que contém um ciclo com 3 vértices, o policial irá ganhar. Na figura 15 temos esse grafo.

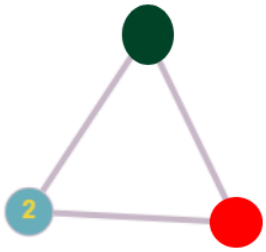
Figura 15 - Ciclo com 3 vértices



Fonte: autoria própria

Suponha que o policial seja o vermelho e olha o vértice 3 e o ladrão seja o verde e ocupa o vértice 1 como na figura 16. O policial começa se movendo para o vértice 1 e já captura o ladrão. Sendo assim, nesse grafo é impossível o ladrão ganhar.

Figura 16 - Jogo de perseguição em ciclo com 3 vértices



Fonte: autoria própria

O corolário 1 utiliza conceitos já apresentados nessa seção, que uma árvore é um grafo *cop-win* e que um grafo com um ciclo maior ou igual a 4 é um grafo *robber-win*.

Corolário 1 [Nowakowski & Winkler]: Se um grafo G é um retrato de um produto finito de caminhos, G pode ser considerado *cop-win*; Se o retrato de G for um ciclo com n≥4, sendo n o número de vértices do grafo, então é um grafo *robber-win*.

3. MATERIAIS E MÉTODOS

Diante a necessidade de entender a lógica e os teoremas dos jogos de perseguição, antes de iniciar a implementação do jogo, foi necessário realizar pesquisas bibliográficas. Portanto, esse trabalho teve cunho tecnológico e cunho ciêntifico.

A primeira etapa do trabalho foi reunir bibliografias para poder aplicar na prática, toda a teoria estudada. Foram utilizados diversos livros sobre grafos para poder escrever sobre a teoria dos grafos, os livros mais utilizados foram matemática discreta e Suas Aplicações do Rosen, Teoria Computacional dos Grafos do Szwarcfiter e Fundamentos da Teoria dos Grafos para Computação da Nicoletti e do Hruschka. Além desses livros, foram utilizados vários livros e apostilas de universidades de todo o Brasil.

Também foi necessário reunir estudos sobre jogos de perseguição utilizando a teoria dos grafos para poder realizar esse trabalho. Um livro utilizado foi Differential Games de Isaacs, foi utilizado o conceito básico de jogos de perseguição. A pesquisa mais utilizada nesse trabalho foi a de Neufeld e Nowakowski sobre jogos de perseguição utilizando grafos.

E a última bibliografia reunida foi sobre Pygame. O livro utilizado nessa pesquisa foi Introdução ao Desenvolvimento de Jogos com Python com Pygame de Kinsley e McGugan. O site do próprio Pygame e o curso Como criar um jogo 2D do canal Uniday Studio também foram utilizados.

A segunda parte do trabalho foi desenvolver um jogo utilizando as ideias e regras da pesquisa realizada. Para a realização do projeto foi utilizado o notebook Samsung da linha Essentials NP350XAA-KF3BR. O sistema operacional utilizado nessa máquina é o Windows 10. A tabela 1 mostra informações adicionais sobre o notebook.

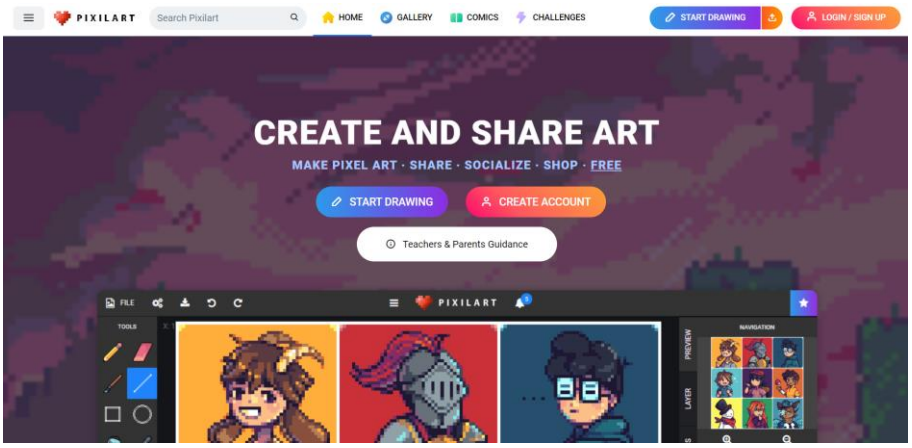
Tabela 1 - Configurações do Computador

RAM	4GB
Tipo do processador	Intel Core i3

Fonte: autoria própria

Para desenhar os personagens, foi utilizado o site pixilart.com, conforme mostrado na 17, que é uma plataforma utilizada para fazer artes em pixel. É uma ferramenta gratuita e feita para todas as idades.

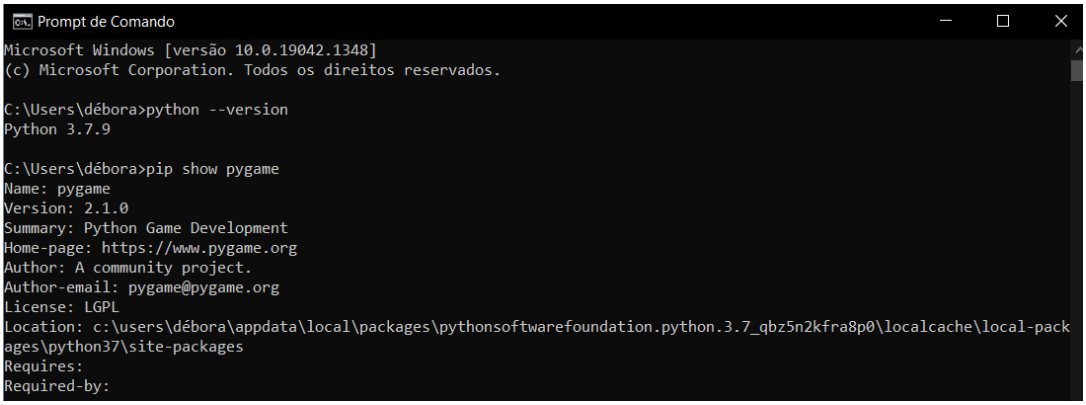
Figura 17 - Pixilart



Fonte: autoria própria

Para o desenvolvimento do jogo, foi utilizada a linguagem Python e também foi necessário usar a biblioteca Pygame, de acordo com a figura 18. Foram instaladas a versão 3.7.9 do Python e a versão 2.1.0 do Pygame.

Figura 18 - Versão Python e Pygame

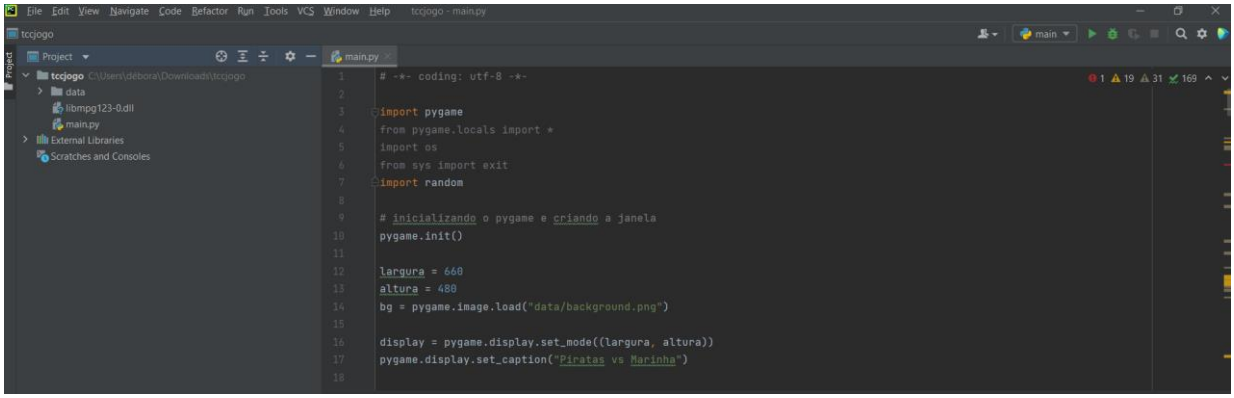


Fonte: autoria própria

Foi utilizada a IDE Pycharm Community, ferramenta criada pela empresa JetBrains, para poder codificar, compilar e realizar testes do jogo. A ferramenta oferece suporte para Python, JavaScript, CoffeeScript, TypeScript, CSS e vários outras linguagens. Também oferece a instalação de plugins e pacotes, sendo um deles o Pygame.

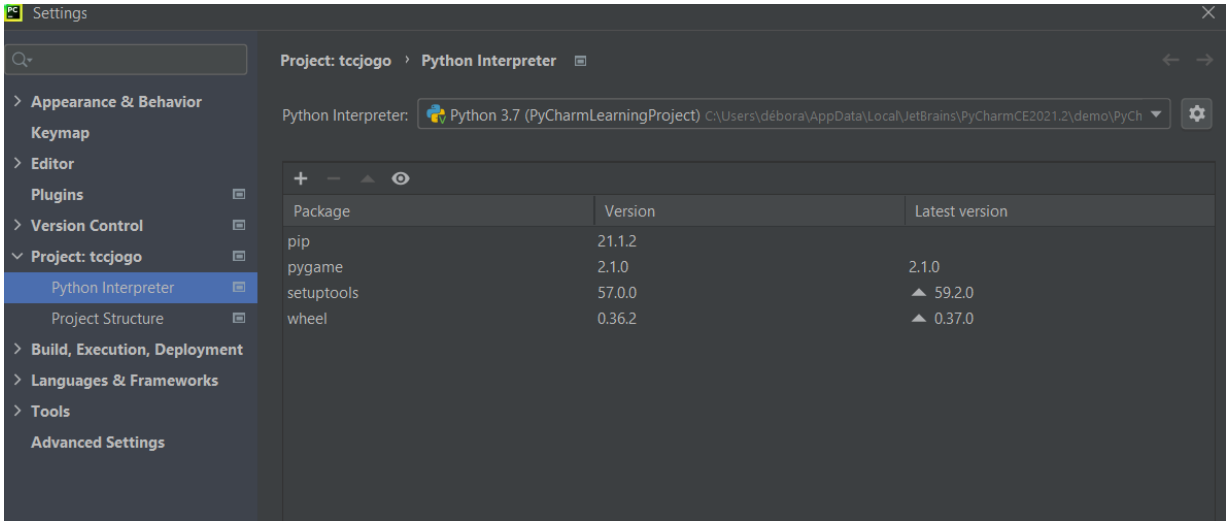
Na figura 19 é mostrado o ambiente de desenvolvimento utilizado para a codificação do jogo e a figura 20 mostra os pacotes instalados no Pycharm.

Figura 19 - Pycharm



Fonte: autoria própria

Figura 20 - Pacotes



Fonte: autoria própria

4. DESENVOLVIMENTO

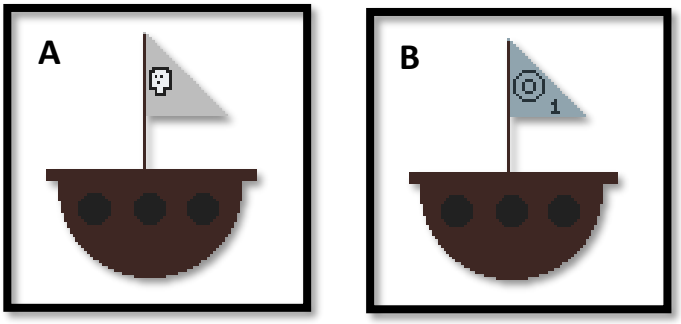
O jogo desenvolvido tem dois personagens, o navio pirata e o navio da marinha. É um jogo *cop-win*, semelhante ao jogo de polícia e ladrão, sendo que os piratas fazem o papel de ladrão e a marinha faz o papel da polícia. Como foi mencionado no capítulo 2, as árvores são grafos *cop-wins* óbvios, então foi desenvolvido um mapa com as características de uma árvore.

4.1 O Jogo

Segundo Guilherme Teres, desenvolvedor de jogos e responsável pelo canal Uniday Studio no YouTube, existem cinco pilares para criar qualquer jogo. São eles: janelas, inputs, desenho, som e lógica.

Usando esse conceito dos cinco pilares de um jogo, durante o desenvolvimento do jogo, o primeiro passo foi criar e desenhar os jogadores do jogo. Foram criados dois jogadores, um navio da marinha e um navio dos piratas, usando a ferramenta pixilart. A figura 21 representam as imagens dos personagens.

Figura 21 - Navio Pirata (A) e Navio da Marinha (B)



Fonte: autoria própria

Após a criação dos personagens, a próxima etapa foi escolher a música do jogo. A música foi escolhida no site OpenGameArt, é um site totalmente gratuito e feito para compartilhar recursos como músicas e desenhos para desenvolvedores de jogos. Sendo assim, é autorizado usar as artes compartilhadas. Foi escolhida uma música de batalha para representar a rivalidade e perseguição entre os dois jogadores. A

música utilizada no jogo se chama battle ThemeA e foi criada por Alex Smith, que utiliza o nome de usuário cynicmusic no site.

Para poder desenvolver os cinco pilares de um jogo, é necessário utilizar os módulos do Pygame. O pacote possui diversos módulos e eles podem ser utilizados independente um do outro. Os módulos desse pacote estão na tabela 2. (KYNSLEY; MCGUGAN, 2019)

Tabela 2 - Módulos do Pygame

Nome do módulo	**Função do módulo**
pygame.cdrom	Acessar e controlar drivers de CD
pygame.cursors	Carregar imagens de cursores
pygame.display	Acessar o display
pygame.draw	Desenhar formas, linhas e pontos
pygame.event	Administrar eventos externos
pygame.font	Utiliza fontes do sistema
pygame.image	Carregar e salvar uma imagem
pygame.joystick	Utilizar joysticks e dispositivos semelhantes
pygame.key	Ler pressionamento de teclas do teclado
pygame.mixer	Carregar e reproduzir sons
pygame.mouse	Administrar o mouse
pygame.movie	Reproduzir arquivos de filmes
pygame.music	Trabalhar com música e com streaming de áudio
pygame.overlay	Acessar funções sofisticadas de vídeo overlays. Contém funções Pygame de alto nível
pygame.rect	Administrar áreas retangulares
pygame.sndarray	Manipular dados de sons
pygame.sprite	Administrar imagens em movimento
pygame.surface	Administrar imagens e tela
pygame.sufarray	Manipular dados de pixels de imagens
pygame.time	Administrar tempo e taxa de frames
pygame.transform	Redimensionar e mover imagens

Fonte: KINSLEY; MCGUGAN (2019, P. 1479)

Com os personagens prontos e a música do jogo escolhida, é necessário continuar desenvolvendo os cinco pilares de um jogo. O próximo passo foi criar uma tela. O Python e o Pygame foram devidamente instalados no computador para poder realizar a implementação do jogo.

4.1.1 TELA DO JOGO

Com Python e Pygame instalados, foi possível iniciar a implementação do jogo. O primeiro passo era criar uma tela e colocar alguns inputs. Para criar uma tela é necessário usar o pacote pygame.display. A figura 22 mostra o código de como a tela foi feita. Para poder usar o pacote, foi necessário importar o pygame e após isso, foi feita a inicialização do Pygame. Logo abaixo foi criada a tela e foi determinado o tamanho e largura.

Figura 22 - Código da tela

```
# -*- coding: utf-8 -*-

import pygame
from pygame.locals import *
from sys import exit
import random



# inicializando o pygame e criando a janela
pygame.init()

largura = 660
altura = 480

display = pygame.display.set_mode((largura, altura))
pygame.display.set_caption("Piratas vs Marinha")
```

Fonte: autoria própria

Após isso, foi iniciado o desenho e criação do grafo na tela. Para desenhar o grafo, foi usado o pacote pygame.draw. Esse pacote possui diversas formas para desenhar em superfícies. Os módulos para desenhar as formas estão na tabela 3.

Tabela 3 - Módulo draw

Nome do módulo	**Função**
pygame.draw.rect	Desenhar um retângulo
pygame.draw.polygon	Desenhar um polígono
pygame.draw.circle	Desenhar um círculo
pygame.draw.ellipse	Desenhar uma elipse
pygame.draw.arc	Desenhar um arco
pygame.draw.line	Desenhar uma reta linha
pygame.draw.lines	Desenhar vários segmentos de retas próximos
pygame.draw.aaline	Desenhar uma linha reta suavizada
pygame.draw.aalines	Desenhar vários segmentos de retas suavizadas próximos

Fonte: KINSLEY; MCGUGAN (2019, P. 2614)

Os módulos utilizados para desenhar o grafo que representa o mapa do jogo foram pygame.draw.line para representar as arestas do grafo e pygame.draw.circle para representar os vértices do grafo.

De acordo com o site do Pygame. O pygame.draw.circle tem quatro parâmetros, sendo eles: circle(superfície, cor, centro, raio). A superfície é onde o desenho será feito, a cor pode ser escolhida através de uma tupla, o centro é o ponto central do círculo e é determinado através de x e y. E por último, o raio que é medido através do parâmetro centro. Além disso, se o raio for menor do que 1, nada será desenhado.

O módulo pygame.draw.line tem cinco parâmetros, sendo eles: line(superfície, cor, ponto inicial, ponto final e largura). Superfície e cor são semelhantes aos parâmetros do pygame.draw.circle. O ponto inicial e o ponto final são determinados através de x, y.

A figura 23 mostra como o mapa foi desenhado. As cores escolhidas foram baseadas nas cores da água e de areia, pois as arestas são as águas onde os piratas e a marinha navegam e os vértices são ilhas onde os navios fazem uma parada.

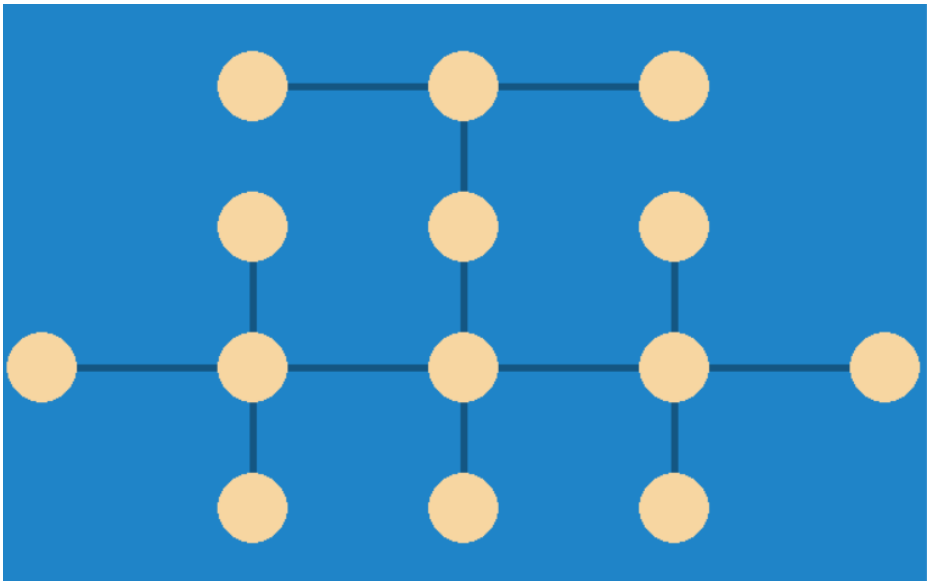
Figura 23 - Código do mapa

```
# MAPA
# aresta1
pygame.draw.line(screen, (20, 86, 130), [30, 300], [150, 300], 5)
# aresta2
pygame.draw.line(screen, (20, 86, 130), [630, 300], [150, 300], 5)
# aresta3
pygame.draw.line(screen, (20, 86, 130), [180, 300], [180, 100], 5)
# aresta4
pygame.draw.line(screen, (20, 86, 130), [330, 300], [330, 100], 5)
# aresta5
pygame.draw.line(screen, (20, 86, 130), [180, 300], [450, 300], 5)
# aresta6
pygame.draw.line(screen, (20, 86, 130), [180, 300], [580, 300], 5)
# aresta7
pygame.draw.line(screen, (20, 86, 130), [480, 300], [480, 100], 5)

# vertice1
pygame.draw.circle(screen, (247, 214, 161), (30, 300), 25)
# vertice2
pygame.draw.circle(screen, (247, 214, 161), (180, 300), 25)
# vertice3
pygame.draw.circle(screen, (247, 214, 161), (180, 100), 25)
# vertice4
pygame.draw.circle(screen, (247, 214, 161), (330, 300), 25)
# vertice5
pygame.draw.circle(screen, (247, 214, 161), (330, 100), 25)
# vertice6
pygame.draw.circle(screen, (247, 214, 161), (480, 300), 25)
# vertice7
pygame.draw.circle(screen, (247, 214, 161), (480, 100), 25)
```

Fonte: autoria própria

Na figura 24 é mostrado o mapa sem os personagens, para dar uma visão geral de como ficou a representação do sistema de grafo neste jogo de perseguição.

Figura 24 - Mapa do Jogo



Fonte: autoria própria

4.1.2 LÓGICA DO JOGO

Nos capítulos anteriores, foi explicado como o jogo funciona. Primeiro o jogador que está perseguindo se move e depois o que está sendo perseguido também pode se mover. Durante o jogo, os jogadores vão jogando alternadamente até que tenha um vencedor. Além disso, foi mencionada a versão passiva do jogo. Nessa versão, o jogador que está sendo perseguido pode permanecer na mesma posição quando for sua vez de jogar, porém quem está perseguindo deve sempre se mover no seu momento de jogar.

Foram utilizadas essas regras do jogo. Porém, ao iniciar, o jogo determina uma posição aleatória para os personagens usando a função random do Python. Com base na posição de cada vértice, foram criados dois vetores para os piratas e dois vetores para a marinha.

Foi necessário criar dois vetores para cada personagem porque foi utilizada a ideia do plano cartesiano na tela do jogo. O plano cartesiano é um plano composto por duas retas que possuem um ponto em comum e assim formando um ângulo de 90°, essas retas são chamadas de perpendiculares. As duas retas são a abcissa e a ordenada, sendo a abcissa responsável pela coordenada horizontal, conhecida como “X”, e a ordenada responsável pela coordenada vertical, conhecida como “Y”. Um ponto no plano cartesiano é representado pela notação P(x,y).

Seguindo a ideia do plano cartesiano, os pontos, que são os vértices do grafo no jogo, são V={(30,300); (180,300); (180,200); (330,300); (330,200); (480,300); (480,200); (630,300); (330,100); (480,100); (180,100); (330,400); (480,400)}, sendo V o conjunto de vértices. Então, um dos vetores contém os valores de x e o outro contém os valores de y.

Primeiro, a função random sorteia um número do vetor que contém os valores de x para o pirata. Após isso, é sorteado um número que contém os valores de y, porém se X=30 ou X=630, y deve ser obrigatoriamente igual a 300 por causa do conjunto V. Entretanto, se x não for 30 e nem 630, y poder ser qualquer número que está no valor do vetor y. Para sortear a posição da marinha, é utilizada a mesma lógica, com apenas uma diferença. Os piratas e a marinha não podem iniciar o jogo no mesmo vértice então foi criado um contador e foi atribuído o valor 0. Enquanto o contador for igual a 0, será realizado o sorteio. Após ser realizado o sorteio, é verificado se a marinha está no mesmo vértice que o pirata. Se os vértices forem diferentes, o contador irá receber o valor 1 e encerrará o sorteio, porém, se os vértices forem iguais, o contador continuará sendo igual a 0 e será realizado um novo sorteio para o vértice da marinha.

A figura 25 mostra o código que determinou a posição dos piratas e a figura 26 mostra o código que determinou a posição da marinha.

Figura 25 - Posição aleatória do navio pirata

```
# posicao aleatoria pirata
posicaop1 = [30, 180, 330, 480, 630]
posicaop2 = [100, 200, 300, 400]
ppirata1 = random.choice(posicaop1)
if (ppirata1 == 30) or (ppirata1 == 630):
    ppirata2 = 300
else:
    ppirata2 = random.choice(posicaop2)

pirata = pygame.sprite.Sprite(drawGroup)
pirata.image = pygame.image.load("data/pixil-frame-0.png")
pirata.image = pygame.transform.scale(pirata.image, [60, 60])
pirata.rect = pygame.Rect(ppirata1 - 30, ppirata2 - 30, 0, 0)
```

Fonte: autoria própria

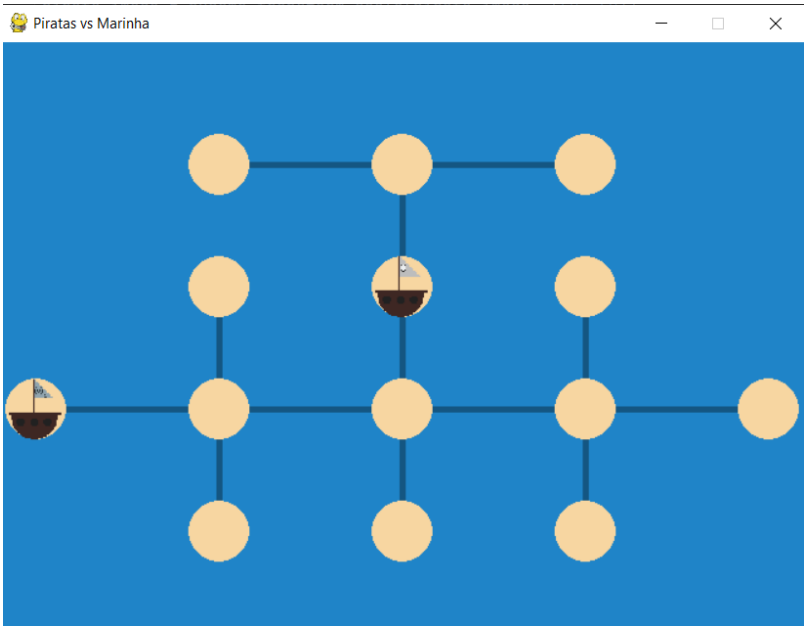
Figura 26 - Posição aleatória do navio da marinha

```
# posicao aleatoria marinha
cont = 0
posicaomar1 = [30, 180, 330, 480, 630]
posicaomar2 = [100, 200, 300, 400]

while cont == 0:
    pmarinha1 = random.choice(posicaomar1)
    if (pmarinha1 == 30) or (pmarinha1 == 630):
        pmarinha2 = 300
    else:
        pmarinha2 = random.choice(posicaomar2)
    if (pmarinha1 == ppirata1) and (pmarinha2 == ppirata2):
        cont = 0
    else:
        cont =1

marinha = pygame.sprite.Sprite(drawGroup)
marinha.image = pygame.image.load("data/pixil-frame-0 (1).png")
marinha.image = pygame.transform.scale(marinha.image, [60, 60])
marinha.rect = pygame.Rect(pmarinha1 - 30, pmarinha2 - 30, 0, 0)
```

Fonte: autoria própria

A figura 27 mostra a posição de cada personagem no grafo após a função random sortear os posicionamentos dos jogadores.

Figura 27 - Tela com personagens



Fonte: autoria própria

Após a escolha da posição inicial dos jogadores, eles já podem começar a se movimentar no jogo. Para os jogadores se moverem, é necessário utilizar o teclado. Foi usada a função pygame.key.get_pressed para verificar se alguma tecla foi pressionada. Essa função retorna se o valor é verdadeiro ou falso e se for verdadeiro, é executado o que está no código. No pygame, é possível consultar uma tecla em particular. (KINSLEY; MCGUGAN, 2019).

Para determinar quem vai jogar foi criado um contador chamado “contpirata”. Sempre que o contpirata for igual a 0, a marinha irá se movimentar. E sempre que contpirata for igual a 1, os piratas irão se movimentar ou apenas ficar na mesma posição. Quem começa jogando é a marinha então contpirata se inicia com 0. Quando for detectada uma tecla pressionada, é verificado se existe algum comando para a tecla. Se houver, é feito o movimento da marinha e contpirata será igual a 1. Quando for a ver dos piratas jogarem, é utilizada a mesma ideia da jogada da marinha, porém contpirata irá receber o valor 0, sendo assim a marinha irá jogar novamente e assim sucessivamente.

Para a marinha se movimentar, é necessário apertar as teclas das setas do teclado. A seta direita faz a marinha se movimentar para a direita, a seta esquerda faz se movimentar para a esquerda, a seta para faz se movimentar para baixo e a seta para cima faz se movimentar para cima. Porém, sempre é verificado se esse movimento é válido pois alguns vértices podem não possuir um vizinho nessa direção. Para movimentar os piratas, é utilizada essa mesma lógica, porém as teclas utilizadas pelos piratas são diferentes. Para se movimentar para a direita é pressionada a tecla “d”, para se movimentar para a esquerda é pressionada a tecla “a”, para se movimentar para baixo é pressionada a tecla “s”, para se movimentar para cima é pressionada a tecla “w” e para continuar no mesmo vértice é pressionada a tecla “x”.

Sempre que um jogador se movimenta, é verificado se eles estão no mesmo vértice. Se eles estiverem no mesmo vértice, contpirata recebe o valor 2 e o jogo se encerra. Como esse grafo é *cop-win*, a marinha sempre irá vencer. Na figura 28 mostra a tela do encerramento do jogo, onde aparece a mensagem de vitória da marinha e também a opção de voltar para o menu.

Figura 28 - Vitória da marinha

Fonte: autoria própria

4.1.3 MENU

Geralmente, em jogos são utilizados menus antes dos jogos se iniciarem. Nos menus podem conter opções para alterar a dificuldade do jogo, o volume do jogo, outras configurações ou apenas pedindo para iniciar o jogo.

No jogo desenvolvido, foi feito um menu apenas com as instruções necessárias para a inicialização do jogo e para jogar e o nome do projeto desenvolvido. Na figura 29 é mostrada a tela do menu.

Figura 29 - Menu

Fonte: autoria própria

Para importar as fontes para o jogo, foi necessário fazer o download delas no site dafont, uma fonte se chama “Pirate Kids”, de autoria de Fachrul Rozi, e a outra se chama “Dogica”, feita por Roberto Mocci. Após realizar o download e copiar as fontes para a pasta do jogo, foi usado o comando pygame.font.init() para iniciar o uso de fontes no jogo. Em seguida, as fontes foram importadas usando o comando pygame.font.Font. Também é necessário usar a função render para renderizar o texto na tela e a função blit para o texto aparecer na tela do menu.

O quadrado contendo as informações do jogo foi feito com o módulo pygame.draw.rect. Esse módulo é utilizado para desenhar retângulos na tela do jogo, podendo escolher a cor, o tamanho e também opções como cantos arredondados.

Após acessar a tecla home no teclado, a tela do menu irá sair para que possa iniciar o jogo.

5. RESULTADOS

Visando que o objetivo do trabalho era estudar a teoria dos jogos de perseguição utilizando grafos e desenvolver um jogo utilizando Pygame, o objetivo foi atingido com sucesso. Entretanto, é um jogo com estratégia *cop-win* e seria interessante desenvolver um outro mapa que utiliza estratégia *robber-win* para que os piratas possam vencer. Além dessa ideia, poderia criar um mapa com mais vértices onde tenha armadilhas que faça os piratas ganharem. Como exemplo, criar um grafo que contém um subgrafo que é uma árvore e que contenha outro subgrafo que é um ciclo com no mínimo quatro vértices.

No capítulo 2 foram abordados os principais conceitos sobre teoria dos grafos e sobre os jogos de perseguição utilizando grafos. Após esse capítulo, foi iniciada a parte de desenvolvimento.

Na **Erro! Fonte de referência não encontrada.** é mostrado que o mapa do jogo é um grafo que contém 14 vértices e existe um caminho entre cada par de vértice do grafo, portanto é um grafo conexo. Esse grafo também é acíclico então, com essas características, é considerado uma árvore. Uma árvore é um grafo *cop-win* obvio. Nesse jogo, *cop-win* é quando a marinha ganha e é *robber-win* quando os piratas ganham. Então, nesse grafo, a marinha sempre irá ganhar dos piratas.

Outro conceito estudado e que foi apresentado no grafo do mapa do jogo é a versão passiva dos jogos de perseguição, onde o pirata não é obrigado a se mover quando for sua vez de jogar.

6. CONSIDERAÇÕES FINAIS

Os jogos de perseguições podem ter muitas versões, variação de número de jogadores e diferentes estratégias de vitória. Nesse trabalho foi abordado de forma teórica como os jogos funcionam, apresentando as diferentes alternativas e teoremas do jogo.

Considerando que o objetivo desse trabalho era apresentar como os jogos de perseguição funcionam e implementar um jogo de acordo com as regras e definições estudadas, o objetivo do trabalho foi concluído.

O jogo implementado apresentou um grafo *cop-win* óbvio, ou seja, uma árvore. A quantidade de jogadores no jogo foi a mínima, apenas um perseguidor e um perseguido. E foi implementado usando a versão passiva do jogo, onde o perseguido não é obrigado a se movimentar. Sendo assim, o jogo mostrou os conceitos estudados.

É importante conhecer alguma das várias aplicabilidades dos grafos, ampliando o conhecimento que foi visto durante o curso. O desenvolvimento em Pygame é interessante porque pode ser visto na prática como os jogos de perseguição com estratégia *cop-win* funcionam.

Para desenvolver um jogo é necessário utilizar cinco pilares: tela, inputs, desenho, som e lógica. Todos esses pilares foram utilizados no desenvolvimento do jogo. No jogo contém uma tela com o desenho do mapa e com os personagens e também tem as instruções do jogo para o melhor entendimento do usuário. É possível jogar utilizando as teclas e cada tecla pressionada é analisada para executar a lógica do jogo.

Apesar do objetivo ser atingido, foi desenvolvido apenas um jogo com estratégia *cop-win*. Uma boa ideia para trabalhos futuros seria desenvolver um jogo com estratégia *robber-win*. A pesquisa apresentada nesse trabalho mostra que existem grafos *cop-win* e *robber-win* e foi mostrado na prática como os jogos com grafo *cop-win* funcionam. Então seria interessante ver através de um jogo desenvolvido como um jogo com grafo *robber-win* funciona.

REFERÊNCIAS BIBLIOGRÁFICAS

GOLDBARG, Marco; GOLDBARG, Elizabeth. **Grafos:** Conceito, algoritmos e aplicações. 1ª edição. Rio de Janeiro: Elsevier, 2012.

ISAACS, Rufus. **Differential Games:** A Mathematical Theory With Applications To Warfare and Pursuit, Control and Optimization. 1ª edição. Nova Iorque: Dove Publications NY, 1999.

KINSLEY, Harrison; MCGUGAN, Will. **Introdução ao desenvolvimento de Jogos em Python com Pygame**. 1ª edição. São Paulo: Novatec Editora Ltda, 2019.

MOTA, Guilherme Oliveira. **Teoria dos Grafos**. 1ª edição. Santo André: CMCC – Universidade Federal do ABC, 2019.

NETTO, Paulo Oswaldo Boaventura. **Grafos:** Teoria, Modelos, Algoritmos. 5ª edição. São Paulo: Blucher, 2012, p. 2.

NEUFELD, S.; NOWAKOWSKI. **A game of cops and robbers played on products of graphs**. Discrete Mathematics, 1998, p. 253-268.

NICOLETTI, MARIA DO CARMO; JR, ESTEVAM R. HRUSCHKA. **Fundamentos da Teoria dos Grafos para Computação**. 3ª edição. Rio de Janeiro: LTC, 2018.

NOWAKOWSKI, Richard J.; WINKLER, Peter. **Vertex-to-vertex pursuit game in a graph**. Discrete Mathematics, 43:235-239, 1983.

RANGEL, S.; OLIVEIRA, V.; ARAUJO, S., **Elementos de Teoria dos Grafos**. 24-24 set. 2018. Notas de aula. Mimeografado.

ROSEN, Kenneth H. **Matemática Discreta e Suas Aplicações**. 6ª edição. Porto Alegre: AMGH Editora LTDA, 2009, p. 589, p. 592-593.

Sem autor: Regra Oficial Grow. Ludopedia, 2021. Disponível em: <https://www.ludopedia.com.br/jogo/interpol/anexos/173517>. Acesso em 26 de novembro de 2021.

SZWARCFITER, Jayme Luiz; OLIVEIRA, Fabiano S.; PINTO, Paulo E. D. **Teoria Computacional de Grafos:** Os Algoritmos. 1ª edição. Rio de Janeiro: Elsevier, 2018, p. 28-30.

Uniday Studio. Como Criar um Jogo 2D | Ninja 2D. Youtube. 24 de abril de 2020. Disponível em: <https://www.youtube.com/playlist?list=PLkei3LlusC-GDUrxjrGFyXNaYcJEt94nn>