UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

MARCO ANTÔNIO ATHAYDE DE AGUIAR VIEIRA

LoBoGames: Uma plataforma virtual acessível de jogos de tabuleiro

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Renato Perez Ribas
Co-orientador: Prof. Dr. Anderson Rocha Tavares

Porto Alegre
2023

AGRADECIMENTOS

Agradeço primeiramente a minha família, por incentivar minhas ideias e por se esforçarem para que eu sempre tivesse as melhores oportunidades possíveis.

Agradeço a todos que me auxiliaram no processo de ingresso à universidade, em especial ao advogado Rogério Bastos, cuja ajuda foi fundamental para que eu pudesse ingressar e manter minha vaga nesta universidade, e à minha mãe Jaqueline, que me motivou a lutar por minha vaga até o final.

Agradeço aos professores e colegas que me ajudaram durante minha jornada acadêmica, em destaque aos meus professores orientadores Renato Ribas e Anderson Tavares, por suas orientações que vão muito além deste trabalho, e ao meu colega Gustavo Kremer, por me acompanhar durante toda essa trajetória.

Agradeço aos meus amigos, que me proporcionaram bons momentos e tornaram todas as dificuldades mais simples de serem enfrentadas.

A todos que, de alguma forma, olharam por mim e torciam pelo meu sucesso, muito obrigado!

RESUMO

Jogos de tabuleiro são uma ótima fonte de entretenimento para todas as idades, pois criam um ambiente competitivo e envolvente, além de estimular o aprendizado e pensamento estratégico. Neste trabalho de graduação, será apresentado como proposta o aplicativo *mobile* LoBoGames, uma plataforma virtual de jogos de tabuleiro, que é oferecida como alternativa aos equipamentos físicos necessários para jogar estes jogos, além de oferecer um adversário de inteligência artificial com opções de níveis de dificuldade. Foi proposta para a plataforma uma nova forma de implementar níveis de dificuldade para agentes de inteligência artificial em jogos de soma zero. A característica principal desta plataforma é a acessibilidade, promovida principalmente através da adaptabilidade para usuários portadores de deficiência visual, pois desta forma o alcance da plataforma pode tornar o impacto destes jogos ainda maior.

Palavras-chave: Jogos de tabuleiro. jogos digitais. acessibilidade. inclusão. inteligência artificial. reúso.

LoBoGames: An accessible virtual board game platform

ABSTRACT

Board games are a great source of entertainment for all ages, as they create a competitive and engaging environment, as well as stimulating learning and strategic thinking. In this graduation work, the mobile application LoBoGames will be presented as a proposal, a virtual platform for board games, which is offered as an alternative to the physical equipment needed to play these games, in addition to offering an artificial intelligence opponent with options of difficulty levels. A new way of implementing difficulty levels for artificial intelligence agents in zero-sum games was proposed for the platform. The main feature of this platform is accessibility, promoted mainly through adaptability for visually impaired users, as this way the reach of the platform can make the impact of these games even greater.

Keywords: Board games. digital games. accessibility. inclusion. artificial inteligence. reuse.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

IA	Inteligência Artificial
IBGE	Instituto Brasileiro de Geografia e Estatística
GGP	General Game Playing
SHOT	Sequential Halving On Trees
MCTS	Monte Carlo Tree Search
UCB	Upper Confidence Bound
AMM	Adaptive MiniMax
API	Application Programming Interface
UI	User Interface

SUMÁRIO

1 INTRODUÇÃO

Jogos digitais são temas de diversos estudos atualmente, desde sua importância para a educação (MCCLARTY et al., 2012), até seu impacto no desenvolvimento das relações sociais (PAPOUTSI; DRIGAS, 2016). Jogos digitais podem ser utilizados como forma de entretenimento, pois proporcionam uma experiência imersiva e engajadora, o que também os torna uma poderosa ferramenta de aprendizado para diversas áreas do conhecimento e diversos tipos de aplicações (MCCLARTY et al., 2012).

Jogos de tabuleiro são um exemplo de desafio instigante que pode ser facilmente adaptado ao mundo digital. Jogos de tabuleiro bem projetados não só criam uma atmosfera envolvente, eles também fornecem um ambiente não ameaçador, lúdico, mas competitivo no qual foca no conteúdo e reforça e aplica o aprendizado (TREHER, 2011).

Neste projeto, serão utilizados apenas jogos de tabuleiro abstratos e de estratégia, isto é, que não possuem temas ou contextualização (RIBAS, 2020) e também não possuem elementos não-determinísticos, como o uso de dados. Estes jogos, além de estimularem o aprendizado e pensamento estratégico, são famosos por promoverem um alto engajamento e competitividade, como é o caso do Xadrez.

A maior parte dos jogos digitais com interface compatível para deficientes visuais que temos no momento são aplicações feitas exclusivamente para portadores de deficiência visual, o que segrega estas pessoas do resto da comunidade de jogos. Mesmo assim, a quantidade de aplicações deste tipo que existe no mercado é escassa. Este baixo investimento não condiz com a dimensão deste tipo de deficiência no país: Segundo o censo demográfico do IBGE (SAÚDE, 2023), cerca de 6.5 milhões de brasileiros (3.5% da população) são portadores de deficiência visual, sendo que 506 mil têm perda total da visão (0.3%) e 6 milhões possuem grande dificuldade para enxergar (3.2%).

Um problema que acompanha os jogos de tabuleiro é sua dependência de equipamentos específicos para cada jogo, como tabuleiros, peças, dados, etc. Este problema pode obter altas proporções quando considerada a vasta quantidade de jogos de tabuleiro abstratos que hoje é conhecida, o que

motiva os desenvolvedores a criarem versões digitais destes jogos. Ainda que diversas plataformas de jogos estejam disponíveis digitalmente, sua maioria contém poucos jogos, priorizando sempre os mais populares. Por conta disso, o acesso aos jogos menos populares se torna limitado, o que impede que a cultura dos jogos se espalhe ainda mais.

Aplicativos para dispositivos *mobile* podem ter seu alcance restringido por conta do alto custo da aquisição de novos dispositivos. No segundo trimestre de 2022 foi identificado que o preço médio de um *smartphone* vendido no Brasil era de R$ 1.800,00 (MEDEIROS, 2022), valor consideravelmente superior ao valor do salário mínimo brasileiro na mesma época de R$ 1.212,00 (RODRIGUES, 2023).

A proposta deste trabalho é desenvolver uma plataforma digital de jogos de tabuleiro que seja totalmente adaptável para usuários com deficiência visual e que também ofereça uma interface familiar ao usuário vidente (portador de visão), a fim de promover ainda mais a inclusão de deficientes visuais na comunidade de jogos digitais e jogos de tabuleiro abstratos.

Esta plataforma ainda deve ser de fácil expansão, para futuramente suprir a baixa quantidade deste tipo de jogo no mercado. Também deve ter requisitos de dispositivo flexíveis, para que a visibilidade do aplicativo engaje projetos de reuso de tecnologia obsoleta, proporcionando uma solução sustentável e de baixo custo para instituições de ensino e de inclusão.

No Capítulo 2 é feita uma revisão técnica sobre jogos de tabuleiro, inteligência artificial para jogos de tabuleiro e softwares inclusivos para deficientes visuais. No Capítulo 3 é apresentada uma revisão bibliográfica, onde são explorados trabalhos relacionados aos principais conceitos a serem utilizados neste trabalho. No Capítulo 4 serão explicados os detalhes da aplicação proposta e sua implementação. O Capítulo 5 mostra as principais características da plataforma obtida e alguns de seus resultados. O Capítulo 6 conclui o trabalho e apresenta ideias de trabalhos futuros.

2 REVISÃO TÉCNICA

2.1 Jogos de tabuleiro

Jogos de tabuleiro são jogos que geralmente envolvem o uso de peças e uma superfície pré-marcada, o tabuleiro. O conjunto destes jogos não inclui outros jogos da categoria *tabletop*, como jogos de cartas, de dados e *wargames*.

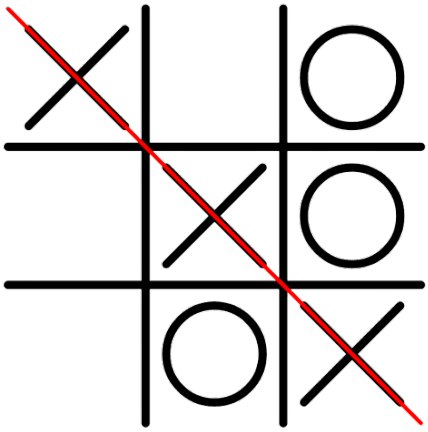
Serão utilizados neste trabalho somente jogos de tabuleiro pertencentes à subcategoria de jogos abstratos e de estratégia. Nestes jogos não há contextualização, além de não possuírem elementos não-determinísticos e garantirem informação perfeita sobre o estado do jogo para todos os jogadores. Nas subseções seguintes, serão descritas as regras dos jogos implementados no projeto.

2.1.1 Jogo-da-velha

O clássico Jogo-da-velha é um jogo muito presente na cultura brasileira. O jogo utiliza um tabuleiro de tamanho 3×3. Em sua vez, o jogador pode inserir uma peça em qualquer posição do tabuleiro, desde que esteja vazia. Ganha o jogador que conseguir alinhar três de suas peças em qualquer direção (horizontal, vertical e diagonal). Quando o tabuleiro é preenchido por completo e nenhum dos jogadores alinhou três peças, é atingido um estado de empate, popularmente descrito como ”deu velha”.

O tabuleiro é popularmente representado por uma matriz quadriculada em formato de cerca, e as peças do jogo são representadas através das formas “X” e “O”, pois são símbolos fáceis de se reproduzir a mão utilizando lápis ou caneta. Também é possível utilizar para este jogo o tabuleiro do jogo Tapatan (descrito na Seção 2.1.2), sem a necessidade de alterar nenhuma das regras.

Figura 2.1: Exemplo de estado terminal do Jogo-da-velha.

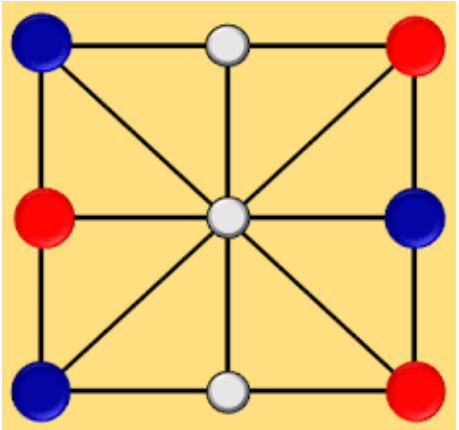


Fonte: (WEISSTEIN, 2022).

2.1.2 Tapatan

O jogo Tapatan é jogado em um tabuleiro 3×3 formado por pontos e linhas. Cada jogador possui três peças posicionadas no tabuleiro. É possível mover as peças, deslocando-as para posições adjacentes conectadas por uma das linhas do tabuleiro. Assim como no Jogo-da-velha, o objetivo é alinhar três peças em qualquer direção, porém este jogo não possui condição de empate.

Figura 2.2: Posição inicial do Tapatan.

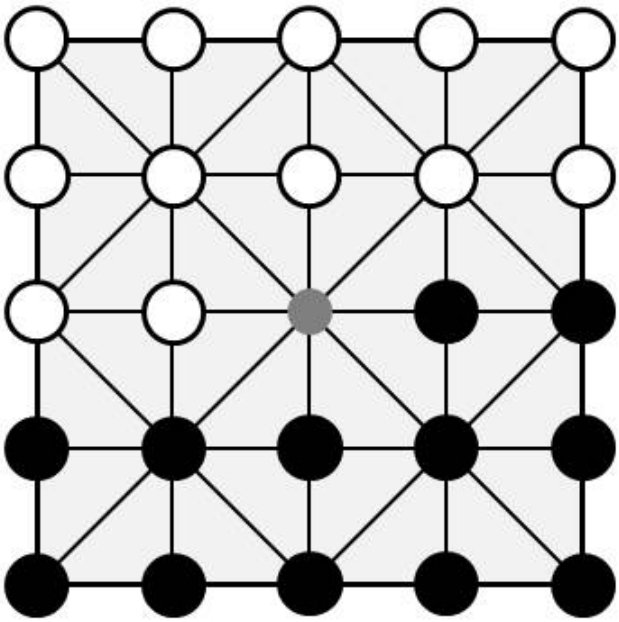


Fonte: (GIORDANI; RIBAS, 2014).

2.1.3 Alquerque

O jogo Alquerque é um jogo de captura, onde cada jogador possui doze peças que iniciam posicionadas no tabuleiro. O tabuleiro do jogo tem tamanho 5×5, e assim como o tabuleiro do Tapatan, é formado por pontos e linhas.

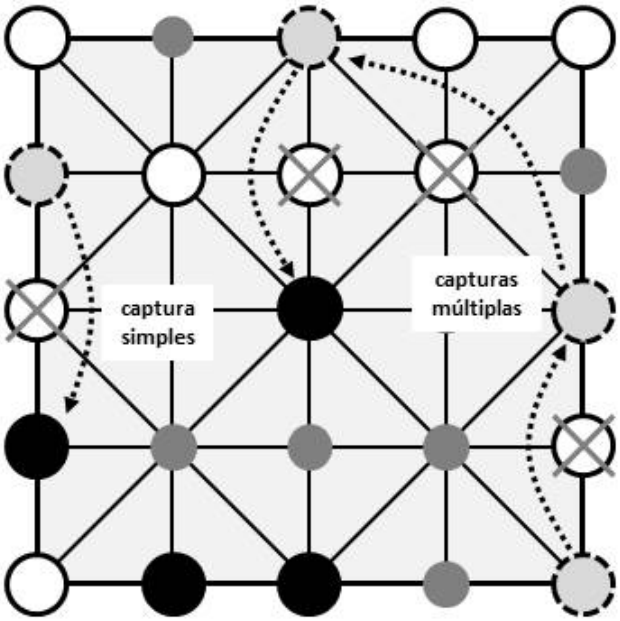
Figura 2.3: Posição inicial do Alquerque.



Fonte: (RIBAS, 2020).

O jogador pode capturar peças adversárias saltando sobre elas, respeitando as linhas do tabuleiro. Capturas em sequência são permitidas, sendo que o jogador é obrigado a capturar o máximo de peças que puder em sua jogada. Caso não existam possibilidades de captura, as peças podem se deslocar para posições adjacentes. Vence o jogador que capturar todas as peças do adversário. O jogo não possui estado de empate, mas em situações onde os dois jogadores possuem poucas peças o jogo pode se tornar monótono, o que neste caso permite que os jogadores possam alcançar o empate do jogo por meio de um acordo.

Figura 2.4: Exemplos de captura do Alquerque.

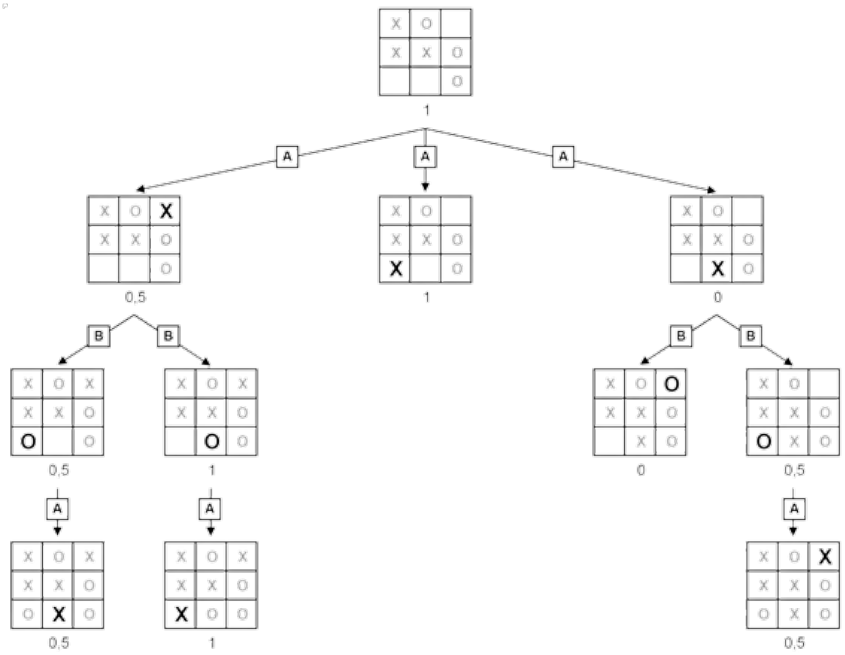


Fonte: (RIBAS, 2020).

2.2 IA aplicada a jogos de soma zero

A maioria dos jogos de tabuleiro pertencem a categoria de jogos de soma zero, tal que a vitória de um jogador resulta na derrota dos demais. Este tipo de jogo pode ser analisado através de uma “árvore de jogo”, onde cada nodo representa um estado e cada aresta representa uma jogada. A Figura 2.5 representa uma parte da árvore de jogo do Jogo-da-velha, onde “A” simboliza uma ação da peça X e “B” simboliza uma ação da peça O. Árvores de jogo apresentam características relevantes para a análise de um jogo, como a profundidade da árvore e o *branching factor* (quantidade de possíveis jogadas por nodo).

Figura 2.5: Uma parte da árvore de jogo do Jogo-da-velha.



Fonte: (MELKÓ; NAGY, 2007).

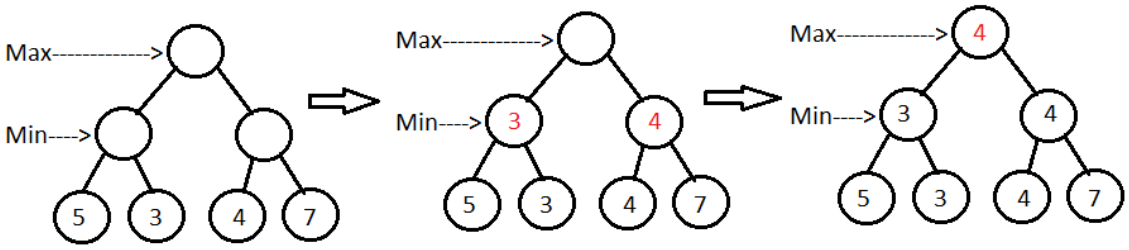
Os jogos de tabuleiro abstratos e de estratégia também possuem propriedades específicas, como o determinismo e a informação perfeita, o que torna conceitualmente possível gerar uma árvore de jogo completa, inferindo a melhor jogada para todos os possíveis estados. Na prática, a complexidade exponencial da árvore de jogo pode limitar a capacidade de sua exploração em um tempo satisfatório. Por isso, algoritmos de inteligência artificial costumam explorar parcialmente a árvore de jogo.

2.2.1 Algoritmo Minimax

O algoritmo minimax, que é um dos algoritmos clássicos da inteligência artificial para jogos de tabuleiro, explora a definição de jogos de soma zero, onde cada jogador deve escolher a jogada de utilidade máxima para si mesmo, considerando que o adversário também fará o mesmo, minimizando sua utilidade. A busca explora cada ramificação da árvore de jogo até encontrar um estado terminal do jogo (nodo folha da árvore de jogo), onde a utilidade do estado é avaliada conforme as condições de vitória do próprio jogo: Valor máximo para uma vitória, e valor mínimo para uma derrota. Guiado pelo princípio de que um jogador tenta maximizar sua

própria vantagem enquanto seu adversário tenta minimizá-la, o algoritmo infere o valor dos estados não-terminais até por fim encontrar o valor da raiz. A Figura 2.6 exemplifica o comportamento da avaliação, diferenciando os estados onde o valor é minimizado e maximizado.

Figura 2.6: Avaliação do algoritmo minimax.



Fonte: (BAELDUNG, 2022).

Uma forma de reduzir a complexidade de tempo do algoritmo sem comprometer seu resultado é a poda alfa-beta (Algoritmo 1). Nesta poda, o valor α representa a melhor escolha para a maximização até o momento, e o valor β a melhor escolha para a minimização. Os valores α e β são comparados para definir a necessidade de explorar as ramificações: Tanto durante a maximização, no caso onde $\beta < \alpha$, quanto durante a minimização, no caso onde $\alpha > \beta$, a exploração das ramificações seguintes é cortada, pois seria desnecessária.

Este algoritmo é completo para qualquer jogo de soma zero determinístico, mas por conta da complexidade de tempo exponencial é comum que a implementação deste algoritmo seja limitada por uma profundidade máxima de exploração da árvore. Podemos também implementá-lo com profundidade iterativa, recomeçando a busca com uma profundidade máxima maior a cada iteração, o que permite limitar a execução do algoritmo pela quantidade de tempo investido.

Em estados não-terminais na profundidade máxima de exploração, é necessário utilizar uma heurística de avaliação para inferir a utilidade do estado obtida pelo agente. A elaboração desta heurística normalmente requer o conhecimento prévio do programador sobre o jogo a ser jogado pelo agente, o que limita este algoritmo, tal que o torna eficiente somente para uma quantidade pré-estabelecida de jogos.

Algoritmo 1 Minimax com poda alfa-beta

```
1:  procedure minimax(state, depth, alpha, beta)
2:      if depth = 0 or state is terminal then
3:          return evaluation of state
4:      if is maximizer's turn then
5:          u ← −∞
6:          for all possible moves in state do
7:              nextState ← apply move in state
8:              v ← minimax(nextState, depth − 1, alpha, beta)
9:              u ← max(u, v)
10:             if u ≥ beta then
11:                 break
12:             alpha ← max(u, alpha)
13:         return u
14:     else
15:         u ← ∞
16:         for all possible moves in state do
17:             nextState ← apply move in state
18:             v ← minimax(nextState, depth − 1, alpha, beta)
19:             u ← min(u, v)
20:             if u ≤ alpha then
21:                 break
22:             beta ← min(u, beta)
23:         return u
```

2.2.2 *General Game Playing*

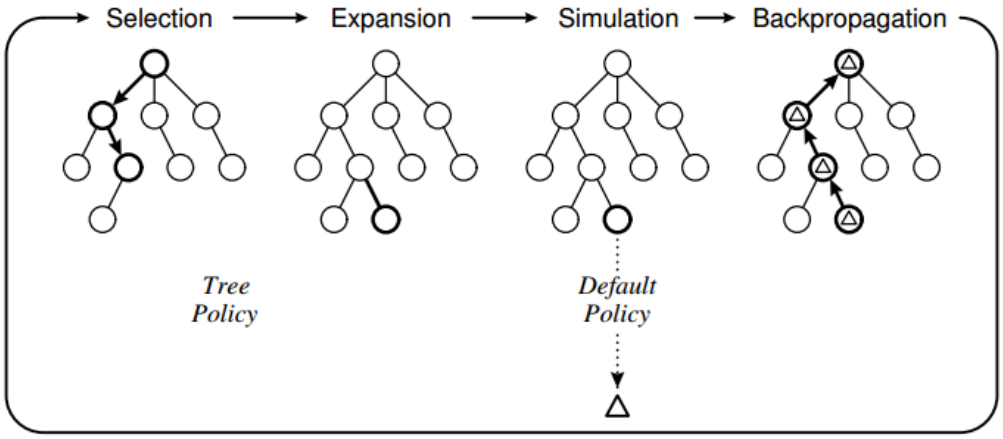
Como dito na seção anterior, algoritmos como o minimax são limitados pela necessidade do programador encontrar uma função de avaliação adequada para cada jogo específico. Agentes de *General Game Playing* (GGPs) são agentes de inteligência artificial projetados para jogar com um conjunto variado de jogos, e devem encontrar boas soluções para problemas ainda não vistos (BROWNE et al., 2012).

Alguns algoritmos se destacam por serem mais adequados para a construção de GGPs do que o algoritmo minimax, como o SHOT (CAZENAVE, 2014) ou o MCTS (Seção 2.2.3). Estes algoritmos utilizam simulações aleatórias de jogo, também chamadas de *playouts*, para gerar uma estimativa da utilidade dos estados e guiar a exploração de acordo com sua política.

2.2.3 Algoritmo MCTS

O processo de busca algoritmo MCTS (*Monte Carlo Tree Search*) pode ser dividido em quatro partes: Seleção, expansão, simulação e retro-propagação (BROWNE et al., 2012).

Figura 2.7: Processo de busca do algoritmo MCTS.



Fonte: (BROWNE et al., 2012).

1. Seleção: Uma política é aplicada recursivamente a partir da raiz até que encontre um nodo expansível (i.e., um nodo não-terminal que contenha filhos não visitados);

2. Expansão: Um (ou mais) nodos filhos são adicionados para expandir a árvore;
3. Simulação: Uma simulação, com escolha de ações normalmente aleatória, roda a partir do novo nodo até um nodo terminal, obtendo um resultado de fim de jogo;
4. Retro-propagação: O resultado da simulação atualiza a estatística dos nodos percorridos durante a seleção.

Uma política de seleção amplamente utilizada é a UCB, sendo que a política UCB mais simples é chamada de UCB1, onde para o nó n com s sucessores, é dada pela fórmula:

$$UCB1 = \frac{\#vitorias_s}{\#jogadas_s} + 2C\sqrt{\frac{2ln(\#jogadas_n)}{\#jogadas_s}}$$

Por conta da seletividade, o algoritmo se torna assimétrico, tal que jogadas que são mais promissoras são exploradas com maior profundidade, enquanto jogadas ruins são menos exploradas (COULOM, 2007).

2.3 Tecnologia assistiva para deficientes visuais

A deficiência visual pode ser classificada em dois grupos: cegueira e baixa visão (ou visão subnormal) (SAÚDE, 2023). As tecnologias evoluíram de forma a utilizar o sentido da visão para comunicar e acelerar a interação homem-computador, o que torna adaptar estas interfaces para deficientes visuais uma tarefa desafiadora.

A tecnologia de leitores de tela é uma das tecnologias assistivas de mais amplo uso por portadores de deficiência visual para este fim. Leitores de tela são produtos de software que ajudam pessoas incapazes de ler, lendo o texto que está na tela e mostrando ao usuário através de um sintetizador de voz ou *display* de Braile (HERSH; JOHNSON, 2008).

Atualmente os principais sistemas operacionais para dispositivos mobile oferecem alguma aplicação ou configuração nativa para leitura de tela e ferramentas para usuários deficientes visuais, sendo o Google Talkback e o VoiceOver os leitores de tela padrão dos sistemas Android e iOS,

respectivamente. Neste trabalho, exploraremos com mais detalhes as funcionalidades do leitor de tela Google Talkback.

O sistema Android oferece nativamente o leitor de tela Google Talkback desde 2009, na versão 1.6 do sistema, podendo ser ativado através das configurações de acessibilidade (CHEN et al., 2009). Atualmente o leitor de tela faz parte do pacote de acessibilidade oficial do Android, disponível para download na Play Store para usuários de Android 6 (M) e versões mais recentes (GOOGLE, 2023).

Durante seu uso, o Google TalkBack insere um cursor na tela, que permite uma navegação sequencial pelos elementos de interface, e informa por meio de uma voz sintetizada o texto no qual está selecionado, se comportando como um mediador entre a interface feita para o usuário vidente e o usuário portador de deficiência visual. Além disso, o leitor de tela também pode mostrar aos usuários informações que não estão escritas na tela, como descrição de imagens e instruções para a interação com os elementos da interface. O cursor pode ser movido através do toque ou de gestos na tela.

3 REVISÃO BIBLIOGRÁFICA

3.1 Acessibilidade em jogos digitais

Podemos definir a acessibilidade como um fator que ajuda pessoas que possuem algum tipo de impedimento (de natureza física, mental, intelectual ou sensorial), e que as possibilitam viver de forma independente e participar plenamente de todos os aspectos da vida. Desta forma, a acessibilidade é um fator impactante para a qualidade de vida de pessoas com deficiência, pois estas possuem impedimentos de longo prazo (BRASIL, 2019).

Jogos são utilizados principalmente como fonte de entretenimento, mas diversos problemas experienciados por jogadores portadores de deficiência podem tornar a experiência de jogar um jogo frustrante (BIERRE et al., 2005). A necessidade de prover formas de acessibilidade para os consumidores de jogos digitais é alta, assim como para qualquer outro produto no mercado.

3.1.1 Acessibilidade para deficientes visuais

Jogadores deficientes visuais podem apresentar dificuldade ou total incapacidade de perceber estímulos visuais nos jogos, o que pode torná-los inaptos a tomar as decisões corretas e pode restringir sua interação com o jogo. Por conta disso, jogos acessíveis para deficientes visuais normalmente substituem os estímulos visuais necessários para a jogabilidade por estímulos auditivos ou hápticos. Podemos identificar algumas das principais estratégias tomadas para tornar os jogos acessíveis para deficientes visuais (YUAN et al., 2011):

- Substituir estímulos visuais por auditivos, geralmente através das seguintes técnicas:
  - Fala, seja fala sintetizada (leitor de tela) ou voz própria.
  - Dicas de áudio, que utilizam sons da vida real para passar informações, como o som do vento ou de passos.
  - Sonificação de objetos ou ações (efeitos sonoros).
- Substituir estímulos visuais por hápticos.

- Realçar os estímulos visuais, através da personalização de esquemas de cores (como cores em alto contraste, esquema de cores para usuários daltônicos ou cores customizáveis) ou opções de zoom e tamanho de fonte.

3.2 IA com níveis de dificuldade

Ajustar a dificuldade percebida ao jogar contra um agente de inteligência artificial é um problema recorrente no mundo dos jogos digitais, com diversas soluções possíveis. O desafio de encontrar uma solução para este problema está em proporcionar uma experiência igualmente satisfatória e com um nível de desafio adequado para jogadores de diferentes níveis de proficiência nos jogos.

3.2.1 Jogos de tempo real

Em jogos de reposta em tempo real, a dificuldade normalmente é ajustada alterando alguns parâmetros do jogo, como quantidade de inimigos, quantidade de obstáculos, precisão dos inimigos, etc. Esta alteração pode ser feita manualmente ou utilizando alguma heurística.

Alguns estudos buscam ainda encontrar maneiras de ajustar dinamicamente esta dificuldade, permitindo que o agente se adapte ao comportamento do jogador (WEBER; NOTARGIACOMO, 2020; SEGUNDO et al., 2016). Para tanto é necessário uma maneira de avaliar o desempenho do agente e do jogador, como um sistema de pontuação ou contagem de falhas e vitórias. Jogos em tempo real podem utilizar um algoritmo de otimização de parâmetros, como um algoritmo genético, para nivelar a dificuldade de forma dinâmica (WEBER; NOTARGIACOMO, 2020).

3.2.2 Jogos de tabuleiro

Para versões digitais de jogos de tabuleiro o problema de ajuste de dificuldade se torna ligeiramente diferente. Não há um consenso sobre qual

fator deve determinar a dificuldade do agente, mas nas soluções propostas foi possível identificar dois fatores que são normalmente considerados: A recompensa das escolhas (desempenho do agente) e a complexidade das escolhas (dificuldade do problema). Um algoritmo guiado pela recompensa das escolhas utiliza uma política que seleciona a jogada através de sua avaliação, como por exemplo escolhendo propositalmente uma jogada avaliada como ruim para simular um nível baixo de dificuldade. Já um algoritmo que utiliza a complexidade das escolhas limita sua avaliação através de seus parâmetros de exploração, como a profundidade máxima ou seu tempo de execução.

O algoritmo AdaptiveMiniMax (AMM) propõe uma extensão ao algoritmo Minimax (descrito na Seção 2.2.1) onde a qualidade das jogadas selecionadas é adaptada conforme a média de avaliação das jogadas do adversário humano (DZIEDZIC, 2016). Este algoritmo determina a dificuldade a partir da seleção dependente da sua avaliação das jogadas, sendo assim guiado pela recompensa das escolhas. Por outro lado, no artigo em que este algoritmo é proposto, também é levado em consideração a profundidade de exploração do AMM no momento da seleção, o que também torna a solução sensível à complexidade das escolhas.

A plataforma *open source* de Xadrez Lichess[1] utiliza o motor de avaliação Stockfish, que por sua vez implementa uma versão do algoritmo Minimax. Explorando seu código aberto, é possível analisar os três parâmetros utilizados para diferenciar os oito níveis de dificuldade disponíveis na plataforma: Duração, *Skill Level* e profundidade. A Tabela 3.1 mostra os valores dos parâmetros, assim como foram encontrados no código. Segundo a documentação do motor Fairy Stockfish (FAIRY-STOCKFISH, 2023), o parâmetro *Skill Level* determina a chance de selecionar outra jogada que não a melhor avaliada, sem maiores explicações de como esta seleção é feita. Com estas informações, podemos identificar os parâmetros de duração e profundidade como sensíveis somente à complexidade das escolhas, mas a classificação do *Skill Level* fica dependente de sua implementação, tal que poderia ser feita tanto com base nas avaliações como com base na hierarquia das avaliações (sensível ao *branching factor*).

[1] Disponível em <https://lichess.org/>

Tabela 3.1: Parâmetros dos níveis de dificuldade do site Lichess

Nível	**Duração (ms)**	***Skill Level***	**Profundidade**
1	50	-9	5
2	100	-5	5
3	150	-1	5
4	200	3	5
5	300	7	5
6	400	11	8
7	500	16	13
8	1000	20	22

Fonte: Repositório de código aberto “fishnet” (LICHESS, 2022).

3.3 Obsolescência de *smartphones*

O termo obsolescência simboliza o processo de algo se tornar obsoleto, desatualizado, não funcional, entre outros. A obsolescência pode ser identificada a partir de cinco principais tipos (MELLAL, 2020):

- Obsolescência técnica (ou tecnológica): Ocorre quando uma nova tecnologia é lançada e o produto é substituído, mesmo se ainda estiver funcional;
- Obsolescência funcional: Quando a função principal do componente é degradada ou insuficiente sem a oportunidade de atualização;
- Obsolescência programada: Obsolescência introduzida na política de produção pelos fabricantes, a fim de aumentar as vendas diminuindo o tempo de vida útil do produto;
- Obsolescência de estilo (ou psicológica): Quando o produto não é mais atrativo porque se encontra fora de moda;
- Obsolescência opcional: Referente à situação onde melhorias tecnológicas não são aplicadas a determinado produto, mesmo que pudessem ser.

A indústria de *smartphones* lança novos produtos em ciclos extremamente rápidos. Grandes empresas que investem na área, como Apple e Samsung, costumam lançar uma nova versão de cada um de seus diversos

modelos de *smartphone* todo ano. Podemos também ver reflexos deste comportamento competitivo nos consumidores, tal que este mercado gera uma pressão para que comprem sempre o último modelo lançado. Segundo um estudo austríaco, a média do tempo de uso de um *smartphone* é de 1.8 anos, sendo que, segundo outro estudo, a principal razão para a substituição é o desejo por um dispositivo novo com mais funcionalidades (PROSKE et al., 2016).

Podemos identificar este padrão de obsolescência dos *smartphones* como predominantemente obsolescência técnica, e possivelmente, obsolescência de estilo, tal que o lançamento de um novo modelo de *smartphone* pode vender a ideia de necessidade por mais funcionalidades, simplesmente por serem novidades. Isso implica que os *smartphones* são majoritariamente descartados e substituídos ainda em estado funcional.

Por outro lado, não podemos considerar que todos os *smartphones* fora de uso por este motivo continuam em pleno estado de funcionamento: A produção e atualização de software para estes dispositivos tende a acompanhar a frequência do lançamento de novos modelos. Desta forma, não só o sistema operacional em dado momento para de ser atualizado, como também os desenvolvedores perdem o interesse de manter a compatibilidade de seus aplicativos com as versões mais antigas do sistema operacional, tornando a obsolescência funcional um fim comum praticamente inevitável.

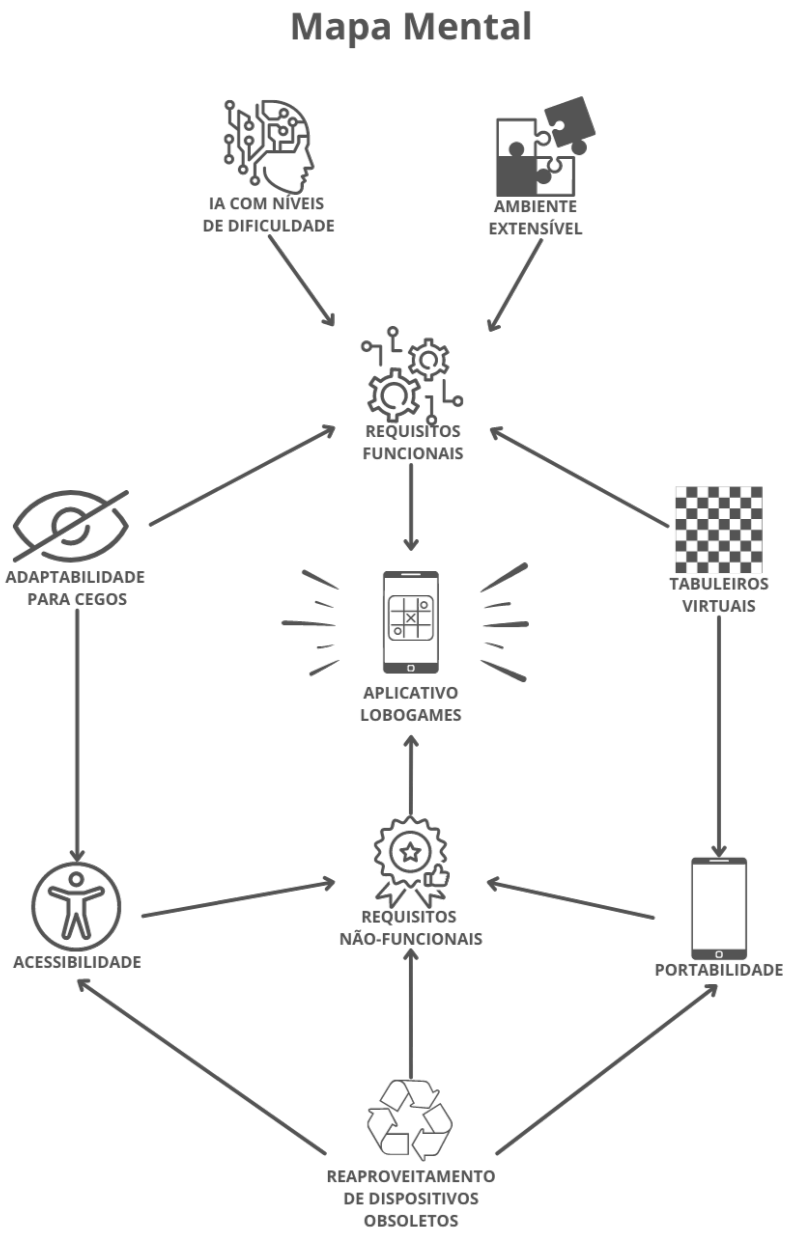
4 METODOLOGIA E IMPLEMENTAÇÃO

4.1 Proposta: Aplicativo LoBoGames

A proposta deste trabalho é desenvolver uma plataforma de jogos de tabuleiro para o sistema Android. O aplicativo foi nomeado ”LoBoGames”, tal que fora agregado ao projeto de extensão da UFRGS de mesmo nome. A seguir, uma lista dos principais requisitos funcionais e não-funcionais atribuídos ao projeto do aplicativo, que serão discutidos com maiores detalhes nas subseções seguintes, acompanhada da Figura 4.1 que apresenta um mapa mental da elaboração da proposta do aplicativo que correlaciona estes requisitos.

- Deve oferecer **tabuleiros virtuais** que substituam a necessidade dos tabuleiros físicos dos jogos;
- Deve ser acessível, incluindo uma forma de **adaptabilidade para deficientes visuais**;
- Deve ser um **ambiente extensível** para que novos jogos sejam implementados com facilidade;
- Deve possuir um modo de jogo para apenas um jogador, sendo possível jogar contra uma **IA com níveis de dificuldade**;
- Deve promover o **reaproveitamento de dispositivos obsoletos** para tornar a solução ainda mais acessível.

Figura 4.1: Mapa mental da proposta do aplicativo.



Fonte: Elaborado pelo autor.

4.1.1 Tabuleiros virtuais

Uma das características da plataforma é oferecer a função de "tabuleiro virtual" para os jogos, ou seja, ela deve servir como uma alternativa a versão física dos jogos oferecidos. Desta forma, um dos modos presentes é o multijogador local, em que todas as peças sejam controladas a

partir do mesmo dispositivo, para que possa ser utilizado como tabuleiro.

Para que a plataforma possa oferecer uma adaptação digital dos jogos oferecidos, é fundamental que a plataforma ofereça uma alternativa digital para todos os componentes físicos necessários para jogar os jogos. Para os jogos de tabuleiro abstratos e de estratégia, apenas dois tipos de componentes se fazem necessários: os tabuleiros e as peças. Neste trabalho, são utilizados apenas jogos que utilizam peças sem funções distintas dentro do jogo, fazendo com que sua única característica específica necessária para os jogos seja uma forma de identificar o jogador que as possui.

4.1.2 Adaptabilidade para deficientes visuais

Assim como foi visto na Seção 3.1, a acessibilidade é uma ferramenta que pode ser utilizada para aumentar a inclusão de portadores de deficiência no mundo dos jogos digitais. O interesse por oferecer soluções acessíveis para deficientes visuais em versões digitais de jogos de tabuleiro tem tomado grandes proporções: Em 2014, a plataforma de Xadrez Lichess anunciou a primeira versão de compatibilidade com leitores de tela oferecida pelo site (DUPLESSIS; POLášEK, 2014).

Tomando como base as estratégias de acessibilidade para deficientes visuais definidas na Seção 3.1.1, nesta proposta é utilizada como principal estratégia substituir os estímulos visuais por estímulos auditivos. Utilizar a fala para descrever os elementos e estado do jogo é uma estratégia muito utilizada por versões digitais de jogos de tabuleiro, tal como é adotada pela plataforma Lichess.

É utilizada a áudio-descrição do jogo por meio de leitores de tela, pois estes já são utilizados no dia a dia pela maior parte dos deficientes visuais brasileiros para utilizar seus celulares e *tablets* (BRASIL, 2019), o que torna a transição entre a utilização do aplicativo proposto e outros recursos do dispositivo fluida, tal que não se faz necessário nenhum conteúdo, configuração ou recurso adicional para tornar o aplicativo utilizável.

Também é oferecido pela plataforma opções configuráveis para as cores das peças do tabuleiro, para que o usuário possa personalizar o esquema de cores de forma que realce as diferenças visuais perceptíveis, o que favorece

usuários deficientes visuais com baixa visão e usuários daltônicos.

4.1.3 Ambiente extensível

Jogos de tabuleiro abstratos como Xadrez, damas, gamão e ludo, são populares no Brasil e podem ser encontrados em lojas de jogos e brinquedos de todo o país, porém estes representam somente uma pequena parte da grande variedade de jogos deste tipo. Um dos objetivos deste projeto é possibilitar que uma maior variedade destes jogos alcance o grande público, mas para isso ser possível é necessário que seja simples a tarefa de desenvolver e adicionar novos jogos para a plataforma.

4.1.4 IA com níveis de dificuldade

Uma característica desejável de uma plataforma de jogos de tabuleiro é um modo *singleplayer*, que consiga oferecer um nível de desafio adequado para jogadores de diferentes níveis de proficiência em cada jogo. Foi analisado na Seção 3.2 como esta questão é abordada em diferentes tipos de jogos digitais, demonstrando que não há uma abordagem canônica a ser seguida.

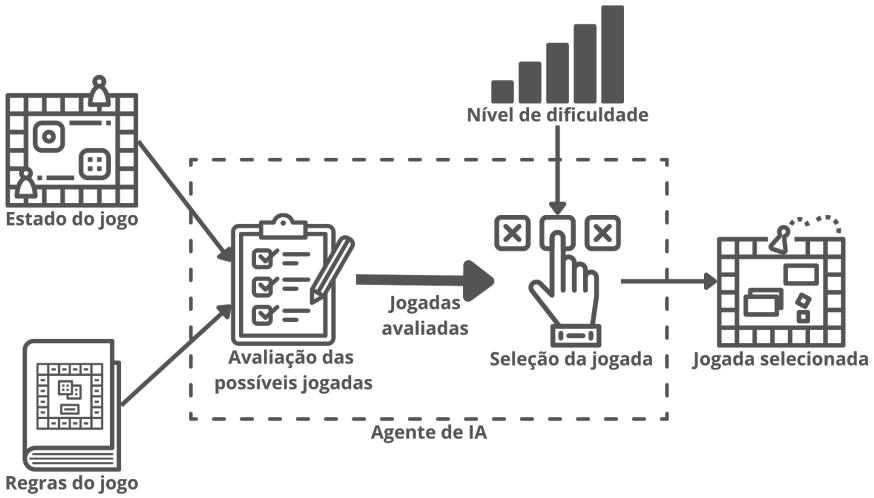
O escopo de jogos que devem ser compatíveis com o agente proposto é diferente dos outros trabalhos antes citados: O agente não deve ser adequado para apenas um jogo, mas sim para uma quantidade indefinida de jogos, com características diferentes e árvores de jogo de complexidades diversas. Deve ser permitido alterar os parâmetros que definem os níveis de dificuldade para cada jogo, mas a solução deve prover por padrão um desempenho satisfatório para diferentes jogos, deixando o nivelamento do agente como opção para quem programa o jogo, mas não como dever do mesmo.

Utilizaremos o conceito de dificuldade determinada por recompensa das escolhas, identificado na Seção 3.2.2, para formular uma proposta para o agente. Dado que a complexidade dos jogos não é uma característica fixa, o nivelamento por complexidade das escolhas também não pode ser fixado. Para exemplificar este comportamento, vamos tomar como exemplo o

nivelamento do site Lichess feito para o Xadrez (Tabela 3.1): Este agente limita a busca do Minimax para a profundidade 22 em seu nível mais alto (nível 8), mas para o Jogo-da-velha, presente na plataforma LoBoGames, uma profundidade 8 (nível 6) partindo da primeira jogada seria o suficiente para o agente analisar a árvore de jogo quase por completo, já que o Jogo-da-velha possui uma árvore de jogo de altura 9.

O agente tem seu trabalho dividido em duas etapas: Avaliação e Seleção. A etapa de avaliação, sem ter o conhecimento do nível de dificuldade, deve analisar as jogadas permitidas pelo agente a partir de um dado estado e atribuir uma avaliação numérica para cada uma delas, simbolizando a recompensa de cada escolha. A etapa de seleção deve ser agnóstica quanto ao jogo e sua complexidade, realizando a escolha da jogada ideal com base apenas no nível de dificuldade e na avaliação realizada pela etapa anterior. A Figura 4.2 ilustra o fluxo do processo de decisão do agente.

Figura 4.2: Proposta para o comportamento do agente de IA.



Fonte: Elaborado pelo autor.

Para este trabalho, utilizaremos o intervalo $[0, 1]$ para os valores de avaliação. Neste caso, se cada jogada estiver perfeitamente avaliada:

- A avaliação 0 caracteriza uma jogada que encaminha ou provoca a derrota do agente;
- A avaliação 0.5 caracteriza uma jogada que encaminha ou provoca empate (caso exista estado de empate), ou ainda que prolonga o jogo

sem contribuir na mudança da vantagem;

- A avaliação 1 caracteriza uma jogada que encaminha ou provoca a vitória do agente.

Para garantir que o processo de seleção seja preciso, é necessário que o algoritmo escolhido para avaliar as jogadas consiga distinguir com clareza jogadas que levam a uma vitória ou derrota eminente. Ainda, é desejável que sua busca explore todas as opções com uma profundidade igual e simétrica, pois a qualidade da exploração de cada jogada deve ser a mesma: Se uma jogada for identificada como uma vitória garantida em n rodadas, é esperado que uma jogada que leve a uma derrota garantida em n rodadas também seja identificada.

O aplicativo LoBoGames não é identificado como uma plataforma de GGP, pois não oferece uma linguagem de descrição generalista para todos os tipos de jogo de tabuleiro, mas sim apresenta uma interface extensível para desenvolvimento dos jogos. Mesmo assim, o agente desenvolvido deve utilizar estratégias de GGP para facilitar a extensão da plataforma. O conhecimento prévio dos jogos que serão analisados pelo agente não deve fazer parte de seu processo de avaliação, que deve apresentar respostas em tempo constante, independentemente da complexidade da árvore de jogo e do hardware e sistema operacional do dispositivo.

Quanto ao processo de seleção, a proposta é caracterizar o agente a partir de três níveis de dificuldade com comportamento distinto: fácil, médio e difícil. O algoritmo de seleção deve utilizar o nível de dificuldade para regular o valor de avaliação da jogada a ser escolhido, regulando a escolha com base em uma chance, mas não deve eliminar a possibilidade do agente escolher uma jogada avaliada em 1 ou 0, o que garante que em qualquer nível de dificuldade o agente poderá vencer ou perder uma partida.

4.1.5 Reaproveitamento de dispositivos obsoletos

Como antes discutido na Seção 3.3, *smartphones* tendem a ser descartados em estado de funcionamento, mas se tornam funcionalmente obsoletos com a medida do tempo. Consequentemente, estes aparelhos

deixam de ser compatíveis com grande parte dos aplicativos, além de não receberem nenhuma atualização do sistema operacional.

Dito isso, pode se concluir que dificilmente dispositivos desatualizados possam ser utilizados para os mesmos fins dos quais foram originalmente projetados, o que reduz drasticamente (ou anula) seus valores de aquisição. Por outro lado, utilizar estes dispositivos de outras formas ainda é possível. O sistema operacional desatualizado reduz as ferramentas para produção de software, tornando mais limitante e desafiador desenvolver aplicativos, mas de forma geral não torna a tarefa infactível.

Devemos lembrar que jogos de tabuleiro abstratos são uma fonte de entretenimento e uma ferramenta de aprendizado, e que uma das missões deste projeto é disseminar a cultura destes jogos. A condição destes dispositivos cria uma oportunidade para aumentar a acessibilidade do aplicativo proposto, já que o investimento necessário é baixo para sua aquisição. O acesso ao aplicativo não só será facilitado para pessoas de baixa renda, como também para instituições de ensino e inclusão de deficientes visuais. Instalar o aplicativo nestes dispositivos pode dá-los um novo objetivo, tornando-os em plataformas de jogos de tabuleiro portáteis, o que reforça a ideia dos tabuleiros virtuais.

4.2 Implementação

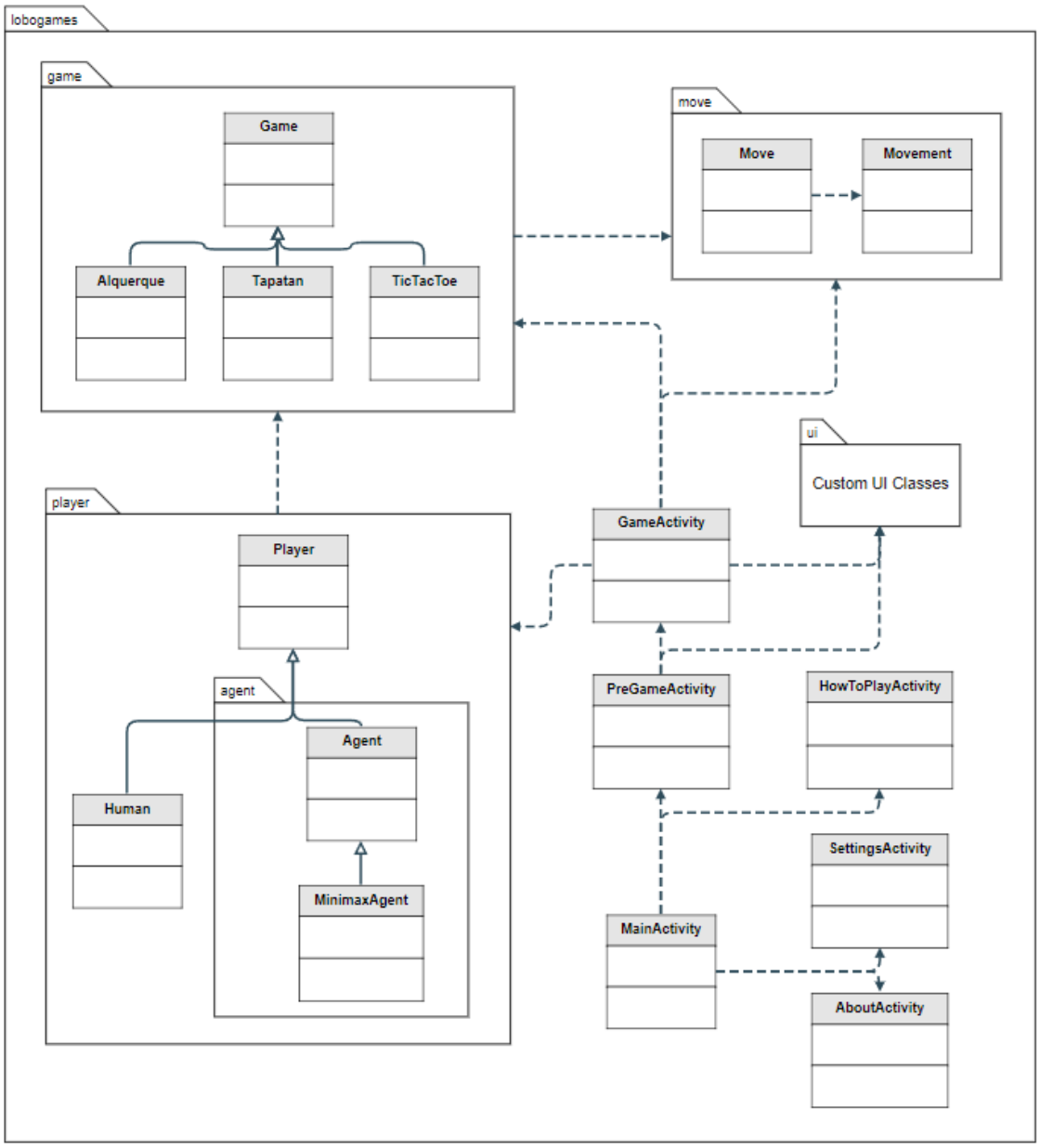
O sistema operacional escolhido para a implementação foi o Android, visto que é o sistema operacional mais presente em dispositivos móveis no Brasil, atualmente com uma parcela de mais de 80% do mercado (STATS, 2023). O aplicativo foi desenvolvido com a linguagem de programação Java, utilizando o ambiente de desenvolvimento integrado (IDE) Android Studio.

A organização do projeto foi feita com base na estrutura padrão de projetos do Android Studio (DESENVOLVEDORES, 2023), que é gerada automaticamente pela IDE e fornece os arquivos e recursos básicos para o desenvolvimento de um *app* na plataforma. Ao invés de começar a execução de um aplicativo a partir de um método *main()*, o sistema Android inicia o código instanciando uma *Activity*. Estas atividades são classes que fornecem uma janela para o desenho da UI, o que geralmente leva ao fato de cada

atividade representar uma tela do aplicativo (DESENVOLVEDORES, 2019).

A hierarquia das classes foi feita pensando em facilitar a mudança e expansão do software, agrupando em pacotes (diretórios) classes com características e interesses em comum. A arquitetura divide o projeto em duas camadas principais, sendo uma camada para as atividades e outra para componentes dos jogos e UI customizada. A Figura 4.3 mostra a estrutura de dependências do aplicativo, através de um diagrama de pacotes. Nas subseções seguintes, os principais detalhes da implementação serão apresentados, relacionando com elementos da estrutura presente no diagrama de pacotes.

Figura 4.3: Diagrama de dependência de pacotes do aplicativo.



Fonte: Elaborado pelo autor.

4.2.1 Atividades

As atividades do aplicativo, identificadas pelo sufixo *Activity*, são agrupadas na raiz do diretório principal (pacote *lobogames*), tal que estas classes representam a base da hierarquia de classes do projeto. A função de cada atividade é apresentada na Tabela 4.1.

Tabela 4.1: Descrição das atividades do aplicativo.

Atividade	**Descrição**
MainActivity	Atividade de menu principal.
PreGameActivity	Atividade de seleção de jogo e seleção de nível de dificuldade.
GameActivity	Atividade de interação com os jogos.
HowToPlayActivity	Atividade de consulta às regras de cada jogo.
SettingsActivity	Atividade de configurações.
AboutActivity	Atividade de informações do aplicativo e desenvolvedor.

4.2.2 Pacote move

O pacote move contém as duas classes que definem a maneira com que os jogadores humanos e agentes de IA interagem com as peças do tabuleiro durante o jogo: A classe *Movement* e a classe *Move*.

A classe *Movement* representa o movimento de uma peça realizado por um jogador, seja ele um humano ou agente de IA. Na abstração escolhida para esta classe, cada movimento pode ser identificado por uma posição inicial, uma posição final e o tipo de peça movimentada. Uma posição pode indicar as coordenadas do tabuleiro ou pode indicar que se encontra fora do tabuleiro. Seguindo esta abstração, movimentos de inserção de peça têm sua posição inicial definida como “fora do tabuleiro”, e analogamente movimentos de remoção de peça têm sua posição final definida como “fora do tabuleiro”.

Em muitos jogos, é possível que um jogador realize mais de um movimento de peças em sua vez de jogar, como por exemplo os jogos de captura, em que o movimento de uma peça de um dos jogadores pode implicar na remoção da peça do outro jogador. Seguindo esta lógica, uma jogada é definida como um conjunto de todos os movimentos que são realizados em um turno por um jogador. A classe *Move* é uma representação deste conceito de jogada, e possui como principais atributos o autor da jogada e a lista de movimentos.

4.2.3 Classes abstratas *Game* e *Agent*

O software fornece classes abstratas para facilitar a extensão do sistema, estas que deixam de forma explícita todos os métodos que têm sua implementação necessária para que tenham compatibilidade com o restante da plataforma. A classe abstrata *Game* (Tabela 4.2) fornece uma interface para a implementação de novos jogos, tornando simples adicioná-los utilizando os tabuleiros dos jogos Alquerque e Tapatan, já presentes na plataforma. Também é possível adicionar novos tabuleiros à plataforma, mas para isso seria também necessário alterar as classes de UI já definidas, o que por sua vez exige que o programador consiga manter a interface da plataforma acessível. A classe abstrata *Agent* possui somente o método abstrato *selectMove*, o qual deve retornar a jogada selecionada pelo agente em um dado estado.

Tabela 4.2: Métodos da classe abstrata Game

Classe abstrata *Game*	
Método	**Descrição**
getName	Retorna o nome do jogo.
getInitialBoard	Retorna o tabuleiro inicial do jogo.
isVictory	Verifica se o estado do jogo é uma vitória de um dado jogador.
isDraw	Verifica se o estado do jogo é um empate.
isLegalMove	Verifica se uma jogada é permitida.
getLegalMoves	Retorna a lista de possíveis jogadas, dado um jogador e o estado do jogo.
getPlayerMove	Interpreta os toques na tela feitos pelo jogador humano, retornando sua jogada.
getRules	Retorna um *String* contendo as regras do jogo.

4.2.4 Compatibilidade com o leitor de tela

O aplicativo foi desenvolvido para ter compatibilidade total com o leitor de tela Google TalkBack, que segundo uma pesquisa brasileira é o leitor de tela mais utilizado para *tablets* e celulares, sendo a opção de 49.7% do total de participantes da pesquisa (BRASIL, 2019).

Diversos componentes disponíveis nos pacotes *android* e *androidx* possuem compatibilidade inata com os serviços de leitor de tela. É o caso da classe *Activity*: Alterar o título de uma atividade com o método *setTitle* altera o título que será lido pelo leitor de tela ao entrar em uma atividade.

Os componentes de interface (como textos e botões) disponíveis no pacote *android* são extensões da classe *View*, sendo que aqueles que possuem texto visível são extensões da classe *TextView*. Estas *views* oferecem compatibilidade para os leitores de tela, tanto através do texto visível inserido quanto através de informações inseridas exclusivamente para os leitores de tela. Além destas informações, *views* interativas também inserem instruções sobre seu uso no modo de acessibilidade, sem qualquer interferência do programador. A Tabela 4.3 mostra como os métodos fornecidos foram utilizados para garantir a acessibilidade.

4.2.5 Cores configuráveis

É possível selecionar as cores para as peças do tabuleiro e para o cursor de seleção de peça dentro do menu de configurações da plataforma. As seguintes opções de cores estão disponíveis para seleção: Verde, vermelho, azul, amarelo, magenta e ciano. É impossível para o usuário definir a mesma cor para dois jogadores diferentes. As mudanças nas configurações são salvas através da classe *SharedPreferences* do pacote *android.content*, que permite manter as modificações feitas no esquema de cores mesmo depois de encerrar o aplicativo.

Tabela 4.3: Métodos utilizados para acessibilidade.

Classe / Método	**Comportamento (leitor de tela)**	**Uso**
Activity.setTitle	Altera o título da *activity*, lido assim que a *activity* é aberta.	Informar nome do *app* e do jogo.
TextView.setText	Altera o texto visível da *view*, lido quando o cursor seleciona a *view*.	Botões de menu e navegação; Textos e *labels* no geral.
View.setContentDescription	Altera a descrição da *view*, lida quando o cursor seleciona a *view*.	Descrição das peças e posições do tabuleiro; Seletor de cores (configurações).
View.announceForAccessibility	Lê um dado texto, independentemente da posição do cursor.	Informar vez do jogador, descrição da jogada e término do jogo.

4.2.6 Agente de IA

A avaliação das jogadas é feita por uma variação do algoritmo Minimax com elementos de GGP, sendo melhor classificado como um algoritmo híbrido de Minimax e MCTS. Este algoritmo foi desenvolvido com o propósito de satisfazer todos os requisitos estabelecidos na proposta deste trabalho, ou seja, não foi identificado na literatura um algoritmo que utilize a mesma combinação de estratégias. A base deste algoritmo é a busca Minimax com poda alpha-beta apresentada no Algoritmo 1. O algoritmo Minimax foi escolhido pois permite que os nodos terminais próximos a raiz da árvore sejam avaliados perfeitamente, identificando com clareza vitórias e derrotas eminentes para o algoritmo de seleção. Do mesmo modo, o Minimax também satisfaz o critério de profundidade simétrica da busca e constante entre todas as possíveis jogadas.

Uma característica que não é presente no algoritmo Minimax clássico é ser independente de conhecimento prévio do jogo, pois se faz necessário a elaboração de uma heurística. Na variação implementada, a avaliação de

estados não-triviais efetuada na linha 3 do algoritmo Minimax, onde tradicionalmente seria o espaço de uma avaliação heurística, é substituída pela média da execução de uma quantidade fixa de *playouts*, similar a avaliação feita pelo MCTS. A quantidade de *playouts* por avaliação de um estado foi fixada em 25 como tentativa de balancear o tempo gasto para adquirir informação de um nó com o tempo total disponível para a avaliação.

A avaliação descrita até então ainda não atende um dos critérios da proposta: Seu tempo de execução deve ser constante. Para satisfazer este critério foi necessário a implementação da busca com profundidade iterativa, o que torna possível limitar sua execução por um valor de tempo investido. O tempo definido para a avaliação foi de 1,5 segundos.

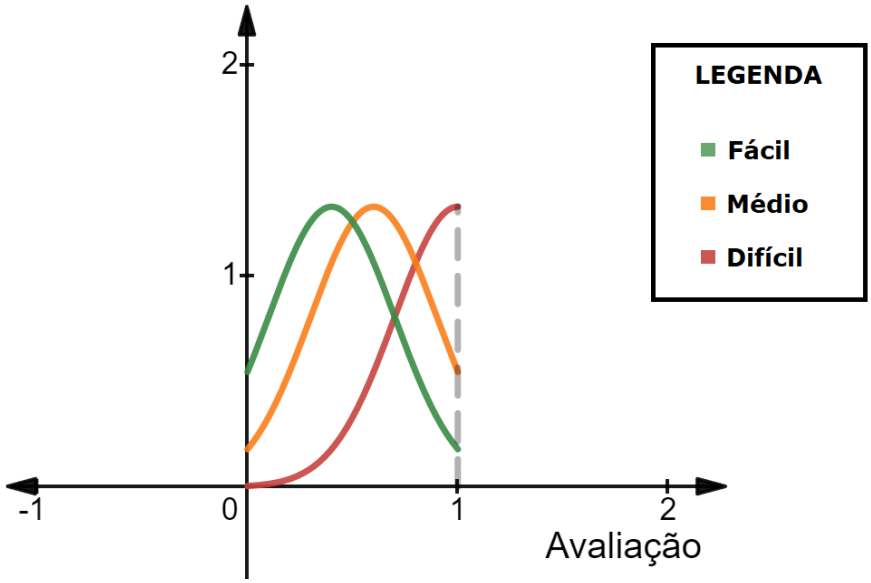
O algoritmo de ajuste de dificuldade implementado aplica uma seleção estocástica, para haver variações nas jogadas escolhidas. Este algoritmo começa sorteando um valor, e depois seleciona a jogada com a avaliação mais próxima a este valor. A ideia do balanceamento é que em níveis mais difíceis o valor sorteado tende a ser maior do que em níveis mais fáceis, implicando na seleção de jogadas com melhor avaliação.

O sorteio do valor é feito através de uma distribuição gaussiana, pois desta forma podemos utilizar a média μ e o desvio padrão σ da distribuição como os parâmetros que definem cada dificuldade. Os três níveis de dificuldade implementados foram nivelados através da média μ e desvio padrão σ da gaussiana, conforme os valores especificados na Tabela 4.4. A Figura 4.4 compara as distribuições, utilizando os valores de μ e σ definidos.

Tabela 4.4: Níveis de dificuldade implementados.

Dificuldade	μ	σ	$P(X < 0.25)$	$P(0.25 \leq X \leq 0.75)$	$P(X > 0.75)$
Fácil	0.4	0.3	30.85%	56.98%	12.17%
Médio	0.6	0.3	12.17%	56.98%	30.85%
Difícil	1.0	0.3	00.62%	19.61%	79.77%

Figura 4.4: Distribuição gaussiana da avaliação selecionada para as jogadas, utilizada pelo algoritmo de seleção.



Fonte: Elaborado pelo autor.

Para analisar o impacto da distribuição escolhida, visto que os jogos implementados até então são simples, podemos utilizar a hipótese de que o processo de avaliação das jogadas obtêm um resultado próximo ao perfeito, ou seja, as avaliações se aproximam dos valores ideais 0, 1 ou 0.5.

Seguindo esta hipótese, tal que um valor X é sorteado, se o valor de X for menor que 0.25 e se existir pelo menos uma jogada cuja avaliação A se aproxima de 0, então uma jogada cuja avaliação A se aproxima de 0 será selecionada, tal que por definição sempre será selecionada a jogada de avaliação mais próxima ao valor sorteado.

Além disso, se adicionarmos a hipótese que, em cada estado, existe pelo menos uma jogada avaliada com uma aproximação de cada um dos valores ideais de avaliação, teremos que $P(X < 0.25) = P(A = 0)$. Se explorarmos ainda mais esta hipótese, também será possível observar que $P(0.25 \leq X \leq 0.75) = P(A = 0.5)$ e $P(X > 0.75) = P(A = 1)$. Esta hipótese é dependente de características específicas do jogo, as quais o algoritmo de seleção não terá acesso, porém, para o conjunto de jogos implementados as métricas inferidas por esta hipótese apresentam uma aproximação do comportamento observável. A Tabela 4.4 utiliza estas métricas para apresentar uma estimativa da proporção da qualidade das escolhas.

4.2.7 Jogos implementados

Para demonstrar a capacidade da plataforma, três jogos foram implementados a partir de extensões da classe *Game* (descrita na Seção 4.2.3): Jogo-da-Velha, Tapatan e Alquerque (descritos na Seção 2.1). Estes jogos utilizam os dois tabuleiros disponíveis, e podem ser utilizados como exemplo para o desenvolvimento de futuros jogos, já que possuem características distintas, identificadas na Tabela 4.5.

Tabela 4.5: Características dos jogos implementados.

Característica	**Jogo-da-Velha**	**Tapatan**	**Alquerque**
Tamanho do tabuleiro	3x3	3x3	5x5
Inicia o jogo com peças colocadas	Não	Sim	Sim
Permite inserção de peças	Sim	Não	Não
Permite movimentação de peças	Não	Sim	Sim
Permite captura de peças	Não	Não	Sim
Permite capturas em sequência	Não	Não	Sim
Condição de vitória	Alinhar 3 peças	Alinhar 3 peças	Capturar todas as peças do adversário
Possui estado de empate	Sim	Não	Não

5 RESULTADOS E AVALIAÇÃO

5.1 Protótipo

O protótipo do aplicativo LoBoGames já se encontra para download gratuitamente na loja de aplicativos Play Store[1]. A tela inicial (figura 5.1) apresenta o logo do projeto LoBoGames e cinco botões para navegar por outras atividades do aplicativo (descritas nas subseções seguintes):

- "UM JOGADOR": Abre a atividade de seleção de jogos e dificuldades *PreGameActivity*;
- "MULTIJOGADOR": Abre a atividade de seleção de jogos e dificuldades *PreGameActivity*, sem permitir a seleção da dificuldade;
- "COMO JOGAR": Abre a atividade *HowToPlay*, que contém a descrição das regras de cada jogo;
- "CONFIGURAÇÕES": Abre a atividade de menu de configurações *SettingsActivity*;
- "SOBRE": Abre a atividade de informações sobre o aplicativo *AboutActivity*.

Figura 5.1: Menu principal do aplicativo LoBoGames.

[1] Disponível em <https://play.google.com/store/apps/details?id=com.marcoantonioaav.lobogames>.

5.1.1 Seleção de jogos e dificuldades

A atividade *PreGameActivity* (Figura 5.2) permite a seleção dos jogos por meio de um menu suspenso que pode ser facilmente estendido com a implementação de novos jogos. No caso da atividade ter sido selecionada através do botão “UM JOGADOR” no menu principal, também é possível selecionar dentre as três dificuldades presentes através de botões de rádio. O botão “JOGAR” abre a atividade *GameActivity* com o jogo e todas as opções e configurações selecionadas.

Figura 5.2: Menu de seleção de jogos e dificuldades do aplicativo LoBoGames.

5.1.2 Tela de jogo

A atividade *GameActivity* (Figura 5.3) representa a tela de jogo da plataforma. Ao topo da tela, o título informa o nome do jogo que está sendo jogado. Logo abaixo do título, um texto informa o estado atual do jogo, indicando a vez do jogador ou o vencedor do jogo, escrito utilizando a cor selecionada do jogador em evidência. O jogador que começa jogando é sorteado a cada inicio de partida.

Cada posição do tabuleiro é pressionável assim como um botão, e só

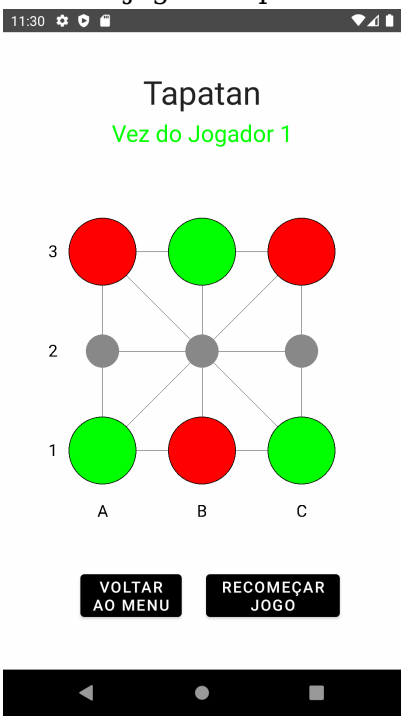
é permitido interagir com o tabuleiro em sua vez, utilizando apenas jogadas permitidas. Em jogos onde a inserção de peças é permitida, pressionar uma posição livre do tabuleiro fará com que uma peça seja inserida. Em jogos onde é possível realizar a movimentação das peças, é necessário que o jogador pressione a peça que deseja mover, e depois pressione a posição de destino.

De forma similar, ao abrir a atividade com o leitor de tela, o título é lido e começa o ciclo de leitura do jogo:

1. O estado atual do jogo é lido antes da vez de qualquer jogador.
2. A descrição de cada posição do tabuleiro informa a peça que está na posição (ou se a posição está vazia) e a coordenada da posição indicada por uma letra e um número, assim como na interface do usuário vidente.
3. Ao selecionar uma peça para movimentação, a descrição da posição muda, indicando também que a peça está selecionada.
4. Ao realizar uma jogada os movimentos que descrevem a jogada são anunciados.

Por fim, o botão “VOLTAR AO MENU” reabre a atividade *MainActivity*, enquanto o botão “RECOMEÇAR JOGO” reinicia o jogo atual, recuperando o estado inicial do tabuleiro e sorteando novamente o jogador inicial.

Figura 5.3: Tela de jogo do aplicativo LoBoGames.



5.1.3 Configurações

A atividade *SettingsActivity* (Figura 5.4) permite alterar as configurações de cores do aplicativo. É possível alterar as cores atribuídas as peças de cada jogador e a cor do cursor de seleção de peças, utilizando um menu suspenso com todas as opções de cores disponíveis. Ao pressionar o botão “SALVAR E VOLTAR”, as mudanças nas configurações são salvas e a atividade *MainActivity* é reaberta.

Figura 5.4: Menu de configurações do aplicativo LoBoGames.

5.1.4 Outras atividades

A atividade *HowToPlayActivity* (Figura 5.5) oferece um menu “Como Jogar”, que serve como um livro de regras rápidas para os jogos. Nesta atividade , o jogo pode ser selecionado em um menu suspenso, assim como na atividade de seleção de jogos. A atividade *AboutActivity* (Figura 5.6) apresenta informações sobre o projeto e sobre contato com o desenvolvedor, além informar a versão do aplicativo instalada.

Figura 5.5: Atividade “Como Jogar” do aplicativo LoBoGames.

Figura 5.6: Atividade “Sobre” do aplicativo LoBoGames.

5.2 Avaliação da Compatibilidade

O aplicativo foi desenvolvido para ser compatível com um grande número de dispositivos, mesmo aqueles não estão mais disponíveis no

mercado. Isto foi alcançado utilizando apenas componentes de software compatíveis com níveis da API do sistema operacional mais baixos que os atuais, ou seja, recursos de desenvolvimento mais antigos. O reflexo disso pode ser observado nos requisitos mínimos do dispositivo (Tabela 5.1), onde está apontada a versão 4.1 do sistema operacional Android, versão lançada em junho de 2012 (GHOSH, 2012).

Tabela 5.1: Requisitos mínimos de dispositivo do aplicativo LoBoGames.

Característica	**Requisito**
Sistema operacional	Android 4.1 Jelly Bean (API de nível 16)
Tamanho de tela seguro	3.7 polegadas - Resolução 480 x 854

Em uma consulta ao catálogo de dispositivos presente no Google Play Console, o aplicativo foi identificado como compatível com 20.671 modelos de dispositivo dentre de 22.284 modelos existentes no catálogo segundo o manifesto definido no aplicativo, totalizando cerca de 92.76% do total.

Desde o lançamento do protótipo, o aplicativo foi instalado e utilizado sem apresentar falhas críticas por usuários com dispositivos com características diversas, totalizando 6 diferentes versões do Android. Usuários do sistema iOS também demonstraram interesse em utilizar a plataforma, mas como a linguagem Java é nativa do sistema Android, adaptar o aplicativo existente para outro sistema operacional também implica em uma adaptação para outra linguagem programação e outro ambiente de desenvolvimento.

5.3 Avaliação da Extensibilidade

Para avaliar a capacidade de expandir o catálogo de jogos da plataforma, foi atribuída a função de desenvolver novos jogos para a plataforma para dois outros desenvolvedores voluntários. Os jogos utilizados para esta experiência foram retirados do livro “21 jogos abstratos e de estratégia no mesmo tabuleiro” (RIBAS, 2020), pois os jogos deste livro utilizam apenas o tabuleiro do jogo Alquerque, que já se encontrava disponível na plataforma. Desta forma, o trabalho estaria voltado em tentar implementar as regras dos jogos, utilizando somente extensões da classe

abstrata *Game*. Os voluntários são estudantes do curso de Ciência da Computação da UFRGS e não possuíam nenhum conhecimento sobre o código do projeto antes da experiência proposta. O prazo estabelecido para a entrega dos jogos foi de um mês, sendo que não foi exigida dos voluntários a dedicação exclusiva para realização desta experiência. Para trabalhar cooperativamente foi utilizada a ferramenta GitHub, onde utilizamos um repositório de código na web para trabalhar remotamente no mesmo projeto.

Um dos desenvolvedores ficou encarregado de desenvolver o jogo Five Field Kono, enquanto o outro desenvolveu o jogo Tsoro Yematatu em uma de suas três variações descritas pelo livro referência (descritos no Apêndice A). Por mais que possuam similaridades com os outros jogos presentes na plataforma, ambos os jogos escolhidos possuem propriedades únicas. O jogo Five Field Kono é classificado como jogo de deslocamento, enquanto os jogos que já haviam sido implementados até então podem ser classificados como jogos de posicionamento (Jogo-da-velha), de alinhamento (Tapatan) e de captura (Alquerque). O jogo Tsoro Yematatu propõe tipos diferentes de interação com as peças de acordo com as duas etapas do jogo: Etapa de inserção e etapa de movimentação.

Ambos os jogos foram adicionados a plataforma com sucesso, e foram desenvolvidos individualmente utilizando somente extensões da classe *Game*. Os voluntários reportaram apenas dificuldades de adaptação com o ambiente de desenvolvimento (Android Studio), mas nenhuma com relação a estrutura do projeto.

Por outro lado, os jogos adicionados possuem propriedades da árvore de jogo que dificultaram a avaliação do agente de IA, como a profundidade dos nodos terminais e seu *branching factor*, o que impacta negativamente a performance dos níveis de dificuldade, principalmente o nível difícil. Em especial, o jogo Five Field Kono apresenta sequências cicilicas de jogadas e estados repetidos (transposições) em sua árvore de jogo, o que torna o uso de *playouts* muito ineficiente. Como solução provisória, foi adicionado um valor limite de profundidade para as *playouts*, fazendo com que a *playout* retorne o valor neutro (0.5) caso chegue ao limite de exploração. O limite de profundidade escolhido foi 75, pois não afeta negativamente os demais jogos. Com esta solução, os resultados da avaliação para este jogo se tornam menos

diferenciáveis, porém o tempo de resposta do agente passa a seguir o tempo de avaliação definido. Algumas outras possíveis soluções ou alternativas mais completas que mantém os objetivos da plataforma seriam:

- Implementar uma tabela de transposição, para evitar exploração excedente em estados repetidos;
- Tornar possível o desenvolvimento de uma função heurística personalizada para cada jogo;
- Utilizar redes neurais como heurística de avaliação, tornando necessário treinar o modelo logo após o desenvolvimento de um jogo.

Foi demonstrado que diminuir as dependências entre as classes e elaborar interfaces enxutas para o desenvolvimento das regras dos jogos permitiu com que os jogos possam ser adicionados de forma simples e independente, o que contribui para a extensibilidade da plataforma. Uma opção para aumentar ainda mais esta extensibilidade seria replicar este processo para a implementação de tabuleiros, a fim de facilitar também a adição de novos tabuleiros na plataforma, considerando o fato de que existem diversos tipos de tabuleiros para jogos abstratos e de estratégia, incluindo diversos tamanhos e formas.

5.4 Avaliação do Usuário

O aplicativo foi apresentado a um grupo usuários cegos para que fosse coletado um retorno sobre sua usabilidade utilizando os leitores de tela. Estes retornos foram obtidos em conversas por áudio, no formato de relato livre. Os três usuários que participaram desta experiência eram adultos cegos, professores vinculados à ACERGS (Associação de Cegos do Rio Grande do Sul), que haviam tido previamente uma formação de trinta horas com os jogos de tabuleiro em sua versão física. Apenas o conjunto original de jogos (Jogo-da-velha, Tapatan e Alquerque) foram utilizados nos testes com usuários.

Segundo estes usuários, a navegação pelo aplicativo ficou simples, sendo fácil e rápido de acessar os recursos da plataforma. A descrição do estado do tabuleiro e das jogadas através do ciclo de leitura definido na

subseção 5.1.2 foram suficientes para localizar os usuários ao jogar o Jogo-da-Velha. Isto significa que há um forte indicativo da viabilidade da proposta de utilizar o leitor de tela para navegar pela plataforma e jogar os jogos de tabuleiro.

Os usuários reportaram dificuldades de entender por completo as possibilidades de jogadas nos jogos Tapatan e Alquerque, isto porque os movimentos nestes jogos estão limitados a andar sobre as linhas do tabuleiro, mas não há na plataforma a descrição do posicionamento destas linhas. Também foi relatado que em alguns dispositivos as jogadas do adversário não estavam sendo anunciadas, o que pode ser um problema de compatibilidade do método *announceForAccessibility* com a versão do leitor de tela utilizada.

Após esta avaliação, foi possível identificar que as maiores barreiras no processo de implementar a adaptabilidade para deficientes visuais não foram por conta das barreiras tecnológicas, mas sim pela complexidade de abordar por completo as dificuldades enfrentadas por estes usuários. Para expandir o conhecimento de como agregar ainda mais na experiência do usuário, o retorno sobre a qualidade da experiência deve ser algo frequente e ativo no processo de desenvolvimento.

Também foi possível, através destes testes, confirmar que há interesse por parte destes usuários de uma aplicação com as características propostas, tal que o projeto foi muito bem recebido por estes usuários, que demonstraram interesse em cooperar com seu processo de desenvolvimento.

6 CONCLUSÃO

O aplicativo desenvolvido neste trabalho conseguiu atingir resultados satisfatórios para todos os requisitos definidos em sua proposta, o que foi fortemente indicado por todas as etapas de avaliação do Capítulo 5. As principais contribuições concebidas pelo projeto podem ser identificadas a partir de cada elemento da proposta:

- Tabuleiros virtuais: Permitem que o usuário tenha uma experiência completa com os jogos de tabuleiro sem necessitar de outros equipamentos além de seu dispositivo móvel, e ainda pode levar o usuário a conhecer jogos de tabuleiro diferentes dos mais populares.
- Adaptabilidade para deficientes visuais: Aplicações que implementam técnicas de acessibilidade podem tornar o uso da tecnologia muito mais inclusivo, não só no nicho dos jogos como também de aplicações de diversas naturezas.
- Ambiente extensível: Aumenta a escalabilidade do projeto e torna mais simples o processo de adicionar novos títulos no catálogo de jogos, o que também contribui com a missão de aumentar a variedade de jogos disponíveis digitalmente.
- IA com níveis de dificuldade: O estudo de IA feito neste trabalho pode servir como base para outras aplicações de jogos de tabuleiro, além de definir um ponto de partida para discussões mais aprofundadas sobre a implementação de níveis de dificuldade para este tipo de problema.
- Reaproveitamento de dispositivos obsoletos: Com a parceria de projetos educacionais e de acessibilidade (como o projeto de extensão LoBoGames), o aplicativo possui o potencial de beneficiar números significativos de usuários deficientes visuais, não só proporcionando uma forma de entretenimento, como também podendo ser utilizado como uma ferramenta de aprendizado e inclusão.

Para trabalhos futuros, é possível indicar o estudo e a aplicação de outras técnicas de acessibilidade para deficientes visuais, como o desenvolvimento de interfaces hápticas, assim como outras formas de aumentar a acessibilidade dos jogos.

O estudo de inteligência artificial deste trabalho deixa aberto um caminho promissor, que seria aprofundar o conhecimento sobre as diferentes formas de implementar níveis de dificuldade para agentes de IA em diferentes tipos de jogos.

REFERÊNCIAS

BAELDUNG. **Introduction to Minimax Algorithm with a Java Implementation**. 2022. Disponível na Internet: <https://www.baeldung.com/java-minimax-algorithm>.

BIERRE, K.; CHETWYND, J.; ELLIS, B.; HINN, D. M.; LUDI, S.; WESTIN, T. Game not over: Accessibility issues in video games. Em: **Proc. of the 3rd International Conference on Universal Access in Human-Computer Interaction**. [S.l.: s.n.], 2005. p. 22–27.

BRASIL everis. **pesquisa brasileira do uso de leitores de tela**. everis Brasil, 2019. Disponível na Internet: <https://mwpt.com.br/wp-content/uploads/2019/07/Pesquisa-LDT_Relatorio.pdf>.

BROWNE, C. B.; POWLEY, E.; WHITEHOUSE, D.; LUCAS, S. M.; COWLING, P. I.; ROHLFSHAGEN, P.; TAVENER, S.; PEREZ, D.; SAMOTHRAKIS, S.; COLTON, S. A survey of monte carlo tree search methods. **IEEE Transactions on Computational Intelligence and AI in games**, IEEE, v. 4, n. 1, p. 1–43, 2012.

CAZENAVE, T. Sequential halving applied to trees. **IEEE Transactions on Computational Intelligence and AI in Games**, IEEE, v. 7, n. 1, p. 102–105, 2014.

CHEN, C.; GANOV, S.; RAMAN, T. **TalkBack: An Open Source Screenreader For Android**. Google Open Source Blog, 2009. Disponível na Internet: <https://opensource.googleblog.com/2009/10/talkback-open-source-screenreader-for.html>.

COULOM, R. Efficient selectivity and backup operators in monte-carlo tree search. Em: SPRINGER. **Computers and Games: 5th International Conference, CG 2006, Turin, Italy, May 29-31, 2006. Revised Papers 5**. [S.l.], 2007. p. 72–83.

DESENVOLVEDORES, A. para. **Introdução a atividades**. 2019. Disponível na Internet: <https://developer.android.com/guide/components/activities/intro-activities?hl=pt-br>.

DESENVOLVEDORES, A. para. **Conhecer o Android Studio**. 2023. Disponível na Internet: <https://developer.android.com/studio/intro?hl=pt-br>.

DUPLESSIS, T.; POLášEK, V. **Accessibility for blind players**. Lichess, 2014. Disponível na Internet: <https://lichess.org/blog/U5AX_DcAADkAz-L5/accessibility-for-blind-players>.

DZIEDZIC, D. Dynamic difficulty adjustment systems for various game genres. **Homo Ludens**, Polskie Towarzystwo Badania Gier (Games Research Association of Poland), v. 9, n. 1, p. 35–51, 2016.

FAIRY-STOCKFISH. **Fairy-Stockfish**. GitHub, 2023. Disponível na Internet: <https://github.com/fairy-stockfish/Fairy-Stockfish/blob/master/README.md>.

GHOSH, A. **Introducing Android 4.1 (Jelly Bean) preview platform, and more**. Android Developers Blog, 2012. Disponível na Internet: <https://android-developers.googleblog.com/2012/06/introducing-android-41-jelly-bean.html>.

GIORDANI, L. F.; RIBAS, R. P. **Jogos Lógicos de Tabuleiro - Jogos de Bloqueio e Alinhamento**. UFRGS, 2014. Disponível na Internet: <https://www.inf.ufrgs.br/lobogames/wp-content/uploads/2015/09/jogos_modulo1_texto.pdf>.

GOOGLE. **Acessibilidade do Android**. Google Play, 2023. Disponível na Internet: <https://play.google.com/store/apps/details?id=com.google.android.marvin.talkback&hl=pt_BR&gl=US>.

HERSH, M. A.; JOHNSON, M. A. **Assistive technology for visually impaired and blind people**. [S.l.]: Springer, 2008.

LICHESS. **fishnet/src/api.rs**. GitHub, 2022. Disponível na Internet: <https://github.com/lichess-org/fishnet/blob/master/src/api.rs#L208>.

MCCLARTY, K. L.; ORR, A.; FREY, P. M.; DOLAN, R. P.; VASSILEVA, V.; MCVAY, A. A literature review of gaming in education. **Gaming in education**, Pearson, v. 2012, p. 1–35, 2012.

MEDEIROS, H. **Preço médio do smartphone no Brasil cresce 10% em um ano**. 2022. Disponível na Internet: <https://www.mobiletime.com.br/noticias/21/09/2022/preco-medio-do-smartphone-no-brasil-cresce-10-em-um-ano/>.

MELKÓ, E.; NAGY, B. Optimal strategy in games with chance nodes. **Acta Cybernetica**, University of Szeged, v. 18, n. 2, p. 171–192, 2007.

MELLAL, M. A. Obsolescence–a review of the literature. **Technology in Society**, Elsevier, v. 63, p. 101347, 2020.

PAPOUTSI, C.; DRIGAS, A. Games for empathy for social impact. International Association of Online Engineering, 2016.

PROSKE, M.; WINZER, J.; MARWEDE, M.; NISSEN, N. F.; LANG, K.-D. Obsolescence of electronics-the example of smartphones. Em: IEEE. **2016 Electronics Goes Green 2016+(EGG)**. [S.l.], 2016. p. 1–8.

RIBAS, R. P. **21 jogos abstratos e de estratégia no mesmo tabuleiro**. [S.l.]: editora metamorfose, 2020.

RODRIGUES, A. **Salário mínimo de R$ 1.320 começa a valer hoje**. 2023. Disponível na Internet: <https://agenciabrasil.ebc.com.br/economia/noticia/2022-12/salario-minimo-de-r-1320-comeca-valer-hoje>.

SAÚDE, B. V. em S. M. D. **13/12 – Dia do Cego**. 2023. Disponível na Internet: <https://bvsms.saude.gov.br/13-12-dia-do-cego-4/#:~:text=De\%20acordo\%20com\%20dados\%20do,ainda\%20que\%20usando\%20\%C3\%B3culos\%20ou>.

SEGUNDO, C. V. N.; EMERSON, K.; CALIXTO, A.; GUSMAO, R. Dynamic difficulty adjustment through parameter manipulation for space shooter game. **Proceedings of SB Games**, 2016.

STATS, S. G. **Mobile Operating System Market Share Brazil**. Statcounter, 2023. Disponível na Internet: <https://gs.statcounter.com/os-market-share/mobile/brazil>.

TREHER, E. N. Learning with board games. **The Learning Key Inc**, 2011.

WEBER, M.; NOTARGIACOMO, P. Dynamic difficulty adjustment in digital games using genetic algorithms. Em: IEEE. **2020 19th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)**. [S.l.], 2020. p. 62–70.

WEISSTEIN, E. W. **Tic-Tac-Toe**. MathWorld–A Wolfram Web Resource, 2022. Disponível na Internet: <https://mathworld.wolfram.com/Tic-Tac-Toe.html>.

YUAN, B.; FOLMER, E.; HARRIS, F. C. Game accessibility: a survey. **Universal Access in the information Society**, Springer, v. 10, p. 81–100, 2011.

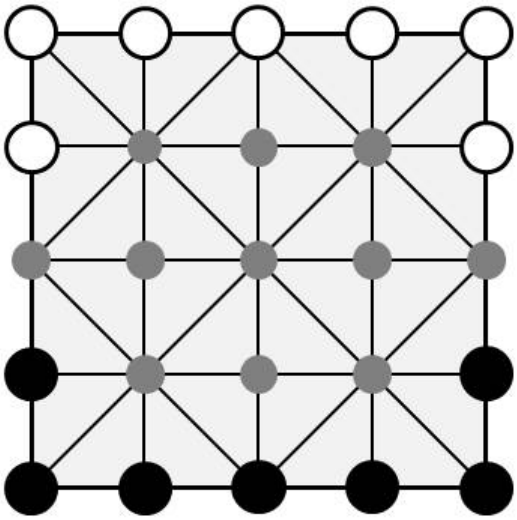
APÊNDICE A — OUTROS JOGOS IMPLEMENTADOS

Neste Apêndice serão descritas as regras dos jogos Five Field Kono e Tsoro Yematatu, estes que foram implementados para avaliar a extensibilidade da plataforma. Ambos os jogos se tratam de adaptações apresentadas no livro “21 jogos abstratos e de estratégia no mesmo tabuleiro” (RIBAS, 2020), onde o tabuleiro original destes jogos é substituído pelo mesmo tabuleiro do jogo Alquerque (descrito na Seção 2.1.3), o que também leva a pequenas alterações das suas regras originais.

A.1 Five Field Kono

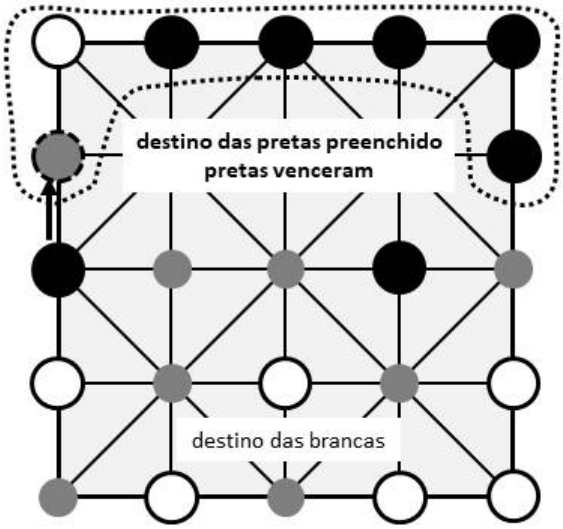
No jogo Five Field Kono, cada jogador possui sete peças, que iniciam posicionadas no tabuleiro (Figura A.1). O objetivo do jogo é preencher o lado adversário (sua posição inicial) com peças (suas ou dele), assim como demonstrado na Figura A.2. É possível deslocar as peças para posições adjacentes conectadas por uma das linhas do tabuleiro, de maneira similar aos jogos Tapatan (Seção 2.1.2) e Alquerque (Seção 2.1.3), porém neste jogo uma peça não poderá mais ser movimentada caso atinja umas das posições finais. Caso um jogador seja bloqueado e fique sem jogadas, então ele perde o jogo.

Figura A.1: Posição inicial do Five Field Kono.



Fonte: (RIBAS, 2020).

Figura A.2: Exemplo da condição de vitória do Five Field Kono.

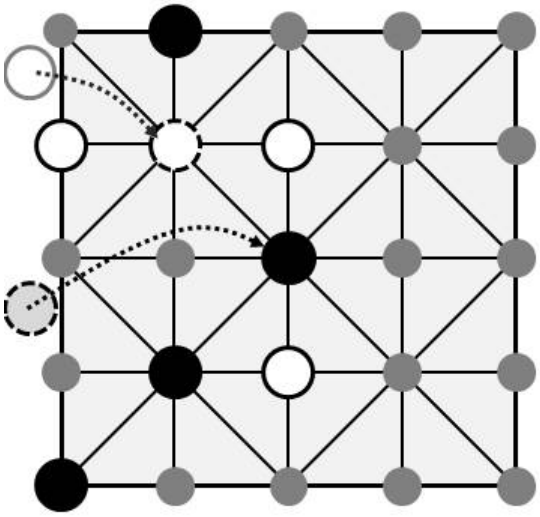


Fonte: (RIBAS, 2020).

A.2 Tsoro Yematatu

No jogo Tsoro Yematatu, cada jogador possui quatro peças, que iniciam fora do tabuleiro. Na primeira etapa do jogo, é possível inserir estas peças (uma por turno) em posições vazias (Figura A.3). O objetivo do jogo é alinhar as quatro peças em uma posição onde haja uma linha do tabuleiro.

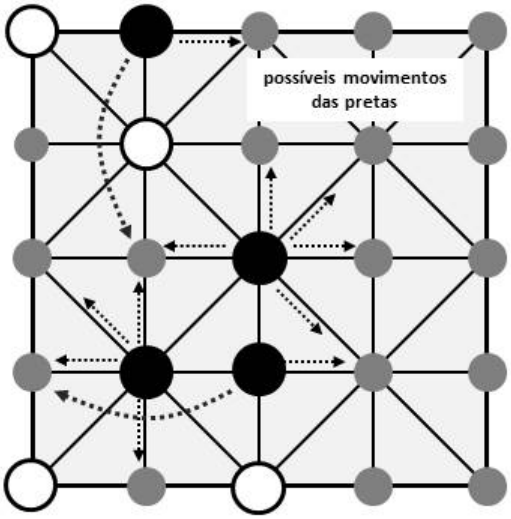
Figura A.3: Exemplo de jogadas na etapa de inserção do Tsoro Yematatu.



Fonte: (RIBAS, 2020).

Assim que todas as quatro peças forem inseridas, se o jogo não tiver terminado, começa a etapa de movimentação. A etapa de movimentação (Figura A.4) possui três variações descritas no livro referência, mas somente uma destas foi implementada na plataforma. Nesta versão, os movimentos podem ser feitos apenas em posições adjacentes e conectadas por uma linha, assim como outros jogos citados. A versão desta etapa representada na Figura A.4 também inclui saltos sobre peças, não presente na versão implementada.

Figura A.4: Exemplo de jogadas na etapa de movimentação do Tsoro Yematatu.



Fonte: (RIBAS, 2020).