Universidade Federal dos Vales do Jequitinhonha e Mucuri
Faculdade de Ciências Exatas e Tecnológicas
Departamento de Computação

Bacharelado em Sistemas de Informação

Aplicação do algoritmo SARSA no balanceamento dinâmico de dificuldade de um jogo digital ortográfico

Vinícius Cordeiro Santos

Diamantina
02 de março de 2016

Universidade Federal dos Vales do Jequitinhonha e Mucuri
Faculdade de Ciências Exatas e Tecnológicas
Departamento de Computação

Vinícius Cordeiro Santos

Aplicação do algoritmo SARSA no balanceamento dinâmico de dificuldade de um jogo digital ortográfico

Trabalho apresentado ao Programa de Bacharelado em Sistemas de Informação do Departamento de Computação da Universidade Federal dos Vales do Jequitinhonha e Mucuri como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Orientadora: *Profa. Dra. Luciana Pereira de Assis*

Diamantina
02 de março de 2016

À minha família e amigos.

Agradecimentos

Desejo expressar minha imensa gratidão...

Aos meus pais, João e Rosa, pelo amor, incentivo e apoio incondicional. A presença de vocês em minha vida faz com que eu tenha certeza que não estou sozinho nessa caminhada, sem vocês eu nada seria.

À meu irmão Fabrício, pela amizade e companheirismo, aos meus amigos Ademir e Hudson, companheiros de jornada ao longo do curso.

À minha namorada Jussara, por todo amor e carinho.

À Profa. Dra. Luciana Pereira de Assis pela paciência na orientação e incentivo que tornaram possível a conclusão deste trabalho de conclusão de curso.

À Profa. Dra. Adriana Nascimento Bodolay pelo seu apoio e incentivo na elaboração deste trabalho.

Finalmente, a Deus por ter me dado força e saúde para superar todas as dificuldades.

Só se pode alcançar um grande êxito
quando nos mantemos fiéis a nós mesmos.
—FRIEDRICH NIETZSCHE (Resíduo)

Resumo

Este trabalho apresenta o desenvolvimento de um jogo digital ortográfico para dispositivos móveis que utilizam a plataforma Android. O jogo proposto busca transformar o momento de aprendizagem da criança numa atividade lúdica e voluntária. O seu cenário é baseado no velho oeste e as suas imagens e animações possuem características direcionadas ao público infantil. O objetivo é auxiliar as crianças no aprendizado do campo de palavras que possuem letras ou dígrafos concorrentes, que representam o mesmo som, no mesmo contexto, e que não existam regularidades de escrita para essas palavras. Para que o jogo se torne interessante e interativo, é necessário apresentar os desafios próprios para cada jogador. Para atender a essa necessidade, foi desenvolvido um algoritmo de inteligência artificial baseado no SARSA para ajustar dinamicamente sua dificuldade, utilizado para alternar os níveis de dificuldade existentes no jogo de acordo com os acertos e erros do jogador, fazendo com que sua jogabilidade não se torne fácil demais, ou por outro lado, difícil demais. A criança pode utilizar esse jogo digital ortográfico onde e quando quiser, para se divertir e aprender seu conteúdo ao mesmo tempo.

Palavras-chave: Algoritmo SARSA. Aprendizagem por Reforço. Inteligência Artificial. Jogos Digitais Educacionais. Jogo Digital Ortográfico.

Abstract

This study presents the development of a digital spelling game for mobile devices based on Android platform. The proposed game seeks to turn the learning time of a child into a playful, voluntary activity. The game scenario is based on the Old West and its pictures and animation have features directed to children. The purpose of the game is to help children in learning a set of words that has competitor digraphs or letters, i.e. digraphs or letters with the same sound and written in the same context in words, however, without grammatical regularities to explain these words. In order to make a more interesting and interactive game, it is necessary to present individual challenges for each player. An artificial intelligence algorithm was developed to fulfill this need. It was based on SARSA to dynamically adjust its difficulty. The algorithm was used to alternate difficulty levels in the game according to hits and misses of players, thus, the gameplay will not be too easy, or too hard. Children can play the digital spelling game wherever and whenever they want, having funny and learning at the same time.

Keywords: SARSA Algorithm. Reinforcement Learning. Artificial Intelligence. Digital Educational Games. Digital Spelling Game.

Sumário

Lista de Figuras

Capítulo 1

Introdução

O presente Trabalho de Conclusão de Curso propõe desenvolver um jogo digital ortográfico direcionado ao público infantil. O nível de dificuldade do jogo proposto é definido dinamicamente por um algoritmo de inteligência artificial baseado no SARSA (Estado-Ação-Recompensa-Estado-Ação).

O jogo explora as palavras com letras ou dígrafos concorrentes, que possuem o mesmo som, no mesmo contexto, e que não existam regularidades de escrita para essas palavras. São abordadas no jogo as classes a seguir:

- Palavras que possuem o mesmo som entre as letras “U” e “L”, por exemplo, “cacarol”, “azul”, “anel”, “mau”, “mal”, etc.;
- Palavras que possuem o mesmo som entre as letras “G” e “J”, por exemplo, “página”, “fugir”, “hoje”, etc.;
- Palavras que possuem o mesmo som entre o dígrafo “SS” e a letra “Ç” , por exemplo, “pássaro”, “girassol”, “engraçado”, etc.;
- Palavras que possuem o mesmo som entre as letras “S” e “Z”, por exemplo, “vaso”, “doze”, “blusa”, etc.;

A seção 1.1 deste capítulo objetiva contextualizar o problema em questão, a seção 1.2 que apresenta os objetivos a serem atingidos e, por último, a seção 1.3 que fornece a estrutura do trabalho.

1.1 Apresentação

Os jogos e brincadeiras são atividades lúdicas que existem desde os tempos mais remotos e desde os primórdios as crianças buscam se divertir usando sua imaginação e o que há

em sua volta, ou seja, qualquer ferramenta que possa desenvolver a criação de situações imaginárias libertando-se do mundo real, tendo como principal importância a espontaneidade e o prazer em brincar e jogar.

É inerente o aprendizado através do jogos. Nestes, as crianças adquirem novos conhecimentos, costumes, aprendem a respeitar regras, aumentam seu raciocínio, trabalham em equipe, entre outros. Pode-se definir que é muito mais do que um instante de lazer, pois trazem uma parcela de atividade sérias sendo um momento que exige disciplina para desenvolver e concluir as atividades e tarefas. "O jogo é uma atividade ou ocupação voluntária, exercida dentro de certos e determinados limites de tempo e de espaço, segundo regras livremente consentidas, mas absolutamente obrigatórias" (Huizinga, 1971, p. 24).

Com o decorrer do tempo e o avanço da tecnologia, surgiram os recursos digitais, que podem ser utilizados por vários meios diferentes e ter determinados fins. Contudo, este trabalho busca estudar os jogos/aplicativos digitais, que são ferramentas que abrem um grande espaço para o entretenimento e a diversão de crianças e adultos e que vêm dominando cada vez mais o cenário mundial.

O surgimento dos aplicativos digitais trazem também a ideia na sua criação com foco em áreas educacionais, onde o usuário aprende determinados tópicos e ao mesmo tempo se diverte, transformando a aprendizagem numa atividade lúdica e voluntária, no qual o usuário não é obrigado a seguir metas e horários podendo aprender onde e quando quiser, mas não quer dizer que metas e horários são prejudiciais para o aprendizado, ao contrário, são de suma importância para o desenvolvimento da criança. O que busca explicitar é a complementação em que essas ferramentas tecnológicas trazem para o usuário na fase de aprendizado.

As tecnologias móveis têm uma parcela essencial na consolidação de aplicativos educacionais, conseguindo atender aos usuários com uma maior usabilidade, portabilidade e principalmente mobilidade. Lee *et al.* (2005) definem esses três requisitos como sendo:

- Mobilidade, estabelecida como a capacidade em que dispositivos móveis tem em exercer várias funcionalidades importantes e facilmente ser deslocado para qualquer ambiente;
- Portabilidade, dispositivos portáteis podem ser facilmente transportados na mão de um ambiente para outro;
- Usabilidade, definido como a capacidade em que o dispositivo móvel possa ser utilizado por usuários diferentes em ambientes totalmente diferentes, a facilidade em que o usuário possa utilizar esse dispositivo para solucionar uma objetivo específico e importante;

Aprendizagem não condiz somente em ficar em uma sala de aula assistindo ao professor com uma postura de transmissor de informações e o aluno receptor. A facilidade de uso desses aplicativos móveis vai transformar qualquer ambiente em um ambiente de estudo e a interatividade vai tornar o ensino muito mais interessante.

Através desses estudos sobre formas lúdicas de aprendizagem e da importância do jogo para a criança é que surge neste trabalho a proposta de desenvolver um jogo digital ortográfico, com o objetivo de explorar o campo de palavras que possuem letras ou dígrafos concorrentes, que representem o mesmo som, no mesmo contexto, e que não existam regularidades de escrita para essas palavras. Como exemplo a palavra "raposa", que contém a letra "s" com som de "z", podendo causar uma certa confusão no momento de aprendizagem.

A ideia em implementar um algoritmo de inteligência artificial no jogo desenvolvido surgiu através da necessidade em fazer com que o mesmo se tornasse interessante e interativo, a função do algoritmo é ajustar dinamicamente a dificuldade do jogo apresentando os próprios desafios para cada usuário, para que não se torne um jogo fácil e tedioso ou também não se torne difícil e cansativo.

1.2 Objetivos

O objetivo geral deste trabalho é desenvolver um jogo digital ortográfico para dipositivos móveis. A utilização do dispositivo móvel busca trazer mobilidade, usabilidade e portabilidade para o usuário. O jogo ortográfico proposto consiste em abordar o campo de palavras com letras ou dígrafos concorrentes, que representam o mesmo som, no mesmo contexto, e que não existam regularidades que determinem qual letra ou dígrafo se deve utilizar. O jogo, portanto, é uma boa alternativa para auxiliar o aprendizado destas palavras. Para que o jogo não se torne tedioso demais, ou por outro lado, difícil demais, este trabalho tem como objetivo implementar um algoritmo de aprendizado por reforço baseado no SARSA para ajustar dinamicamente sua dificuldade.

Os objetivos específicos, são:

- Mostrar a importância dos jogos digitais educacionais na fase do domínio da ortografia dos indivíduos;
- Estudar as ferramentas para tornar possível o desenvolvimento de aplicações para dispo-

sitivos móveis;

- Desenvolver um jogo digital ortográfico para dispositivos móveis que utilizam a plataforma Android, com seu foco direcionado ao público infantil;
- Estudar sobre aprendizagem por reforço;
- Implementar um algoritmo baseado no SARSA para ajustar dinamicamente a dificuldade do jogo proposto;
- Comparar os resultados da aplicação do algoritmo SARSA com a aplicação do algoritmo Q-Learning, no jogo digital ortográfico desenvolvido;
- Identificar aplicações práticas do problema.

1.3 Estrutura do trabalho

Este trabalho possui 7 capítulos, o Capítulo 1 introduz sobre o projeto desenvolvido.

No Capítulo 2 são apresentados os conceitos sobre jogos digitais, jogos digitais educativos, dispositivos móveis, plataforma Android, aprendizagem da ortografia e exemplos de jogos digitais ortográficos. Os tópicos abordados neste capítulo objetivam conduzir o projeto.

No Capítulo 3 são descritas as ferramentas utilizadas no desenvolvimento do projeto.

O Capítulo 4 apresenta os passos para que fosse possível o desenvolvimento do jogo digital ortográfico e o algoritmo de inteligência artificial baseado no SARSA.

No Capítulo 5 são apresentados os testes feitos no algoritmo de inteligência artificial proposto para o jogo, com objetivo de demonstrar o seu comportamento.

O capítulo 6 apresenta as conclusões e o capítulo 7 apresenta os trabalhos futuros.

O trabalho ainda contém 6 Apêndices, os Apêndices possuem códigos referentes a implementação do jogo digital ortográfico e do algoritmo de inteligência artificial, também possui todas as palavras e frases utilizadas no jogo.

Capítulo 2

Referencial Teórico

Este capítulo tem como objetivo realizar um levantamento do suporte teórico sobre o tema a ser tratado, apresentando o estudo de um conjunto de obras literárias que já foram publicadas sobre o tema em questão com intuito de conduzir o projeto.

A seção 2.1 apresenta uma explanação de jogos digitais e sua subseção 2.1.1 sobre jogos digitais educativos. A seção 2.2 apresenta os conceitos sobre dispositivos móveis e a subseção 2.2.1 sobre a plataforma Android. A seção 2.3 aborda aprendizagem da ortografia e, por fim, a seção 2.4 lista alguns exemplos de jogos digitais ortográficos.

2.1 Jogos Digitais

Com a grande expansão da tecnologia na década de 60, surgiram os primeiros jogos digitais, que foram pequenos projetos desenvolvidos por consequência das restrições de conhecimento sobre o assunto e restrições também impostas pela capacidade de processamento de informações dos hardwares da época. “Os primeiros jogos eletrônicos, uma das primeiras formas de interatividade digital de massa, mostravam a capacidade das novas máquinas eletrônicas de representar “ações” onde os homens podem, e devem participar” (Lemos, 1997, p. 1).

Os jogos digitais ganharam espaço através da aceitação dos usuários que buscavam entretenimento e diversão. Vista como uma atividade lúdica e prazerosa, os jogos digitais trazem uma interatividade para o usuário, o que prende a atenção em busca de atingir objetivos específicos ou promover a competição entre jogadores. “Essa liberdade de ação do jogador, essa margem concedida à ação, é essencial ao jogo e explica, em parte, o prazer que ele suscita” (Caillois, 1990, p. 28). “É difícil sustentar que a realidade do jogo deve se encerrar em si mesmo. As consequências do aprendizado e do exercício das funções lúdicas constroem, também, o imaginário” (Pinheiro, 2007, p. 27).

Há uma diversidade de critérios e objetivos entre os jogos digitais, podendo ser classifi-

cados em diversas categorias, as principais são: Aventura, estratégia, combate, esporte, corrida, guerra, raciocínio, simulação, infantil, educacional, etc. O que não deixa de inter-relacionar as categorias, podendo o jogo ser com foco educacional e de raciocínio ao mesmo tempo, por exemplo.

2.1.1 Jogos digitais educativos

Os usuários buscam encontrar nos jogos digitais uma atividade que seja lúdica e prazerosa, o que pode acarretar em seu primeiro malefício, que é o vício. “Nas últimas décadas, crianças, em todo o mundo, vêm fascinando-se com jogos digitais e passam mais horas em frente a telas de computador do que a maioria dos seu pais, avós e professores gostariam” (Canuto e Moita, 2011, p. 2). Muitos usuários ficam demasiado tempo comprometidos nos objetivos desses artefatos, tempo no qual poderia ser proveitoso para fazer outras tarefas, como estudar ou praticar esportes, por exemplo. Outro fator de desvantagem são diversas categorias de jogos que existem, o jogo violento, por exemplo, está entre o topo dessa lista. Dessa forma o que mais se discute é o perigo na associação do mundo imaginário com o mundo real.

De fato, não se podem negar alguns malefícios quando o assunto são jogos digitais, o que divide a opinião de muitos em elogios e críticas. Desenvolvimento desses jogos com foco na educação vem mudando o cenário dos *games*. A intenção é buscar unir o entretenimento e o ensino ao mesmo tempo, transformando o momento de lazer em um momento também de aprendizagem, desenvolvendo uma ferramenta bastante poderosa para auxiliar o ensino. “A motivação do aprendiz acontece como consequência da abordagem pedagógica adotada que utiliza a exploração livre e o lúdico” (Tiellet *et al.*, 2007, p. 4).

Para ser atrativo e divertido, esses jogos devem ter suas interfaces bem projetadas de maneira intuitiva para que o usuário fique acostumado com essa ferramenta e ser didático de maneira em que o usuário tenha interesse em jogar e não note em primeira instância que se trata de um meio de ensino e aprendizagem. “Os aprendizes se envolvem na trama do jogo, fazendo o possível para vencer determinados desafios, em conseqüência, aprendem os conteúdos inseridos” (Tiellet *et al.*, 2007, p. 4).

O que não pode acontecer é de o jogo digital educativo perder o foco e se tornar apenas um jogo de diversão qualquer. As regras devem ser claras e o conteúdo a ser tratado tem que ser analisado com prioridade para que não exista possíveis erros no conteúdo a ser passado, assim tirando a credibilidade do *game*.

Silva *et al.* (2009) deixa claro que não se pode considerar como bom um jogo educativo que, por exemplo, não seja embasado em práticas pedagógicas adequadas ao usuário.

2.2 Dispositivos móveis

Os dispositivos móveis são aparelhos tecnológicos portáteis desenvolvidos com intuito em trazer portabilidade, usabilidade e mobilidade para o usuário. Esses dispositivos trazem a facilidade na locomoção para outros ambientes, buscam atender às mais diversas exigências de seus usuários, e também, buscam suportar as mais diversas aplicações disponíveis. "O desenvolvimento das tecnologias móveis promove novos modelos de comunicação e consumo de informação, produtos e serviços, a partir da criação de experiências em detrimento do simples acesso aos conteúdos e à aquisição de bens" (Fedoce e Squirra, 2011, p. 14).

Esses aparelhos hoje já são integrados aos mais diversos recurso. Muitos Smartphones, por exemplo, possuem recursos alternativos como: GPS, TV digital, câmera fotográfica, Wi-Fi, lanterna, calculadora, rádio, gravador, entre outros, uma ferramenta portátil, de pequeno porte e que se torna um facilitador das tarefas do dia a dia do usuário.

Fedoce e Squirra (2011) reforçam que as mídias portáteis surgem como tecnologias que aumentam a conectividde no mundo dos usuários. Cada vez mais surgem aplicativos completamente avançados com alta rapidez no processamento de dados, auxiliando no contato com outros usuários localizados em outros ambientes, também auxiliando nas suas tarefas do dia a dia, dentre diversas outras utilidades.

A demanda por esses dispositivos portáteis cresceu exponencialmente nos últimos dez anos. A quantidade de pessoas com idade de 10 anos ou mais que possuíam telefones celulares no ano de 2005 era de 55,7 milhões, 36,6% da população brasileira. No ano de 2011, a quantidade de proprietários de telefone móvel aumentou para 115,4 milhões, 69,1% da população brasileira. Ou seja, em 6 anos houve um crescimento de mais de 100% na aquisição desses aparelhos eletrônicos (IBGE, 2013).

2.2.1 Plataforma Android

Existem diversos sistemas operacionais para dispositivos móveis, como: Android, iOS, Windows Phone, dentre outros. O jogo digital ortográfico desenvolvido neste trabalho funciona

somente na plataforma Android. Essa plataforma foi baseada no Linux e é utilizada principalmente em dispositivos que possuem a tela *touchscreen* (tela sensível ao toque), como os *tablets* e *smartphones*. "O Android é uma plataforma para tecnologia móvel completa, envolvendo um pacote com programas para celulares, já com um sistema operacional, *middleware*, aplicativos e *interface* do usuário." (Pereira e Silva, 2009, p. 3).

A plataforma Android foi desenvolvida pela empresa Google, tendo como parceria a aliança OHA (*Open Handset Alliance*). O grupo OHA é composto por organizações que estão no topo do mercado, empresas como a Samsung, Acer, Motorola, Sony Ericsson e entre outras. O Grupo OHA tem como objetivo criar uma plataforma de código aberto e totalmente livre para dispositivos móveis, podendo atender todas as tendências do mercado (Lecheta, 2010). "O fato de o Android ser um código aberto contribui muito para seu aperfeiçoamento, uma vez que desenvolvedores de todos os lugares do mundo podem contribuir para seu código-fonte" (Lecheta, 2010, p. 21).

Bastante flexível e de fácil usabilidade, a plataforma conquista cada vez mais usuários que buscam um aparelho portátil que atenda todas suas necessidades. Seus aplicativos podem ser desenvolvidos por qualquer usuário, basta instalar o Android SDK, uma ferramenta utilizada para desenvolver aplicativos na plataforma Android e, a partir daí, podendo começar a produzir seu próprio aplicativo e até mesmo disseminar na loja virtual do Android, que é a *play store*, para que outras pessoas também possa utilizar seu aplicativo.

Com constantes atualizações, o Android é a plataforma mais utilizada no desenvolvimento de dispositivos móveis na atualidade. A organização especializada em estudos sobre tecnologia e internet (GLOBALWEBINDEX, 2014) fez uma pesquisa onde comprova que 69% dos dispositivos existentes possuem o sistema operacional Android, possuindo uma vantagem imensa sobre seus concorrentes, como o iOS da apple e o Windows Phone, que pertence Microsoft.

2.3 Aprendizagem da ortografia

O aplicativo educacional desenvolvido neste trabalho objetiva auxiliar os usuários na aprendizagem de um determinado conjunto de palavras, com intuito em transformar o momento de aprendizagem em um momento divertido. Para que as pessoas utilizem a escrita de forma correta, em cada língua existe um conjunto de regras que tem por objetivo instruir o aluno

na aprendizagem. Ortografia é o nome dado a esse conjunto de regras. “A ortografia é uma convenção social que visa padronização da forma escrita, evitando por meio dela, diversas maneiras de escrever dentro de uma mesma língua” (Sampaio, 2012, p. 29).

É de suma importância que a ortografia seja inserida em sala de aula no começo da aprendizagem, quando o aluno estiver começando a ler e escrever pequenas frases ou textos e já esteja interpretando valor sonoro de cada letra. Sendo assim, o aluno já começa desde cedo acostumar com as diversas regras da escrita. Nobile e Barrera (2009) descreve a convenção ortográfica como sendo um meio facilitador na comunicação social, que independe da cultura, tempo ou região de um país.

O que chama atenção é a dificuldade que alguns alunos têm em dominar a escrita, com erros ortográficos constantes. Faz-se necessário um diagnóstico de onde está surgindo essas dificuldades ortográficas. “Pode ser difícil, para muitas crianças, compreender como as palavras devem ser apropriadamente grafadas, o que pode ser observado nas alterações ortográficas presentes em suas produções escritas” (Zorzi e Ciasca, 2009, p. 406).

Zorzi e Ciasca (2009) ainda completam que a escrita e a leitura utilizam táticas ortográficas e fonológicas. Diversos indivíduos enfrentam obstáculos e dificuldades quando se trata das questões fonológicas, ao mesmo tempo que, outros indivídos possuem uma limitação maior quando se trata de questões ortográficas.

Segundo Lemle (2009), as palavras que contenham letras ou dígrafos concorrentes, que representam o mesmo som, na mesma posição da palavra, são as mais difíceis de se aprenderem. Existem vários casos que geram uma imensa confusão, as palavras “reza” e “mesa” por exemplo, estão localizadas entre duas vogais e uma se escreve com “z” e a outra se escreve com “s”.

Dentre o caso dessas palavras abordadas no parágrafo acima, existem as que possuem regras que determinam qual letra ou dígrafo é correto de se utilizar e as que são irregulares. Segundo Morais (2007), as palavras irregulares não possuem regularidades, como é o caso mencionado anteriormente das palavras “reza” e “mesa”, pelo simples fato de que a escrita dessas palavras foram determinadas levando em conta as suas questões históricas. Os hábitos de uso dessa palavra no passado fez com que sua forma de escrita se tornasse convencional. Para que a criança aprenda a maneira correta de se escrever esse conjunto de palavras, é necessário recorrer ao dicionário, tendo que decorá-las. Porém, memorizar essas palavras uma por uma pode ser um processo bastante desgastante. Para sanar esse possível problema, métodos alternativos de ensino podem ser explorados fazendo com que essa aprendizagem se torne mais

interessante e intuitiva, como exemplo, o desenvolvimento de um jogo digital ortográfico.

2.4 Exemplos de jogos digitais ortográficos

As subseções abaixo apresentam jogos digitais ortográficos que auxiliam o usuário no processo de aprendizagem e alfabetização. Os jogos digitais educativos surgem em uma mistura de aspectos lúdicos que os jogos possuem com conteúdos pedagógicos e vem fazendo sucesso com as crianças e até mesmo com os adultos, “Não há como negar a presença dos recursos tecnológicos no dia a dia e se associados ao processo lúdico permitem trabalhar qualquer conteúdo de forma prazerosa e divertida."(Falkembach, 2006, p. 1). As subseções abaixo contém variados objetivos de aprendizagem, como exemplo, aprendizagem de substantivos, coletivos, etc.

2.4.1 Jogo das Palavras

Jogo das Palavras é um jogo gratuito disponível na *Internet* no qual o usuário tem a opção de entrar com o nível de dificuldade das perguntas. Os níveis variam de 1 a 4, começando com o nível 1 que é o mais fácil e aumentando a dificuldade a cada nível posterior. O usuário escolhe a palavra que achar certa e o jogo responde se está correta ou não e dá uma breve descrição sobre a regra ortográfica dessa palavra. As figuras 2.1 e 2.2 ilustram o Jogo das Palavras[1]. Como pode ser observado na figura 2.1, o jogo apresenta a palavra escrita “incorretamente”, podendo fazer com que os usuários memorizem a palavra escrita de maneira errada.

[1] Site do Jogo das Palavras. URL http://educarparacrescer.abril.com.br/grafia/.

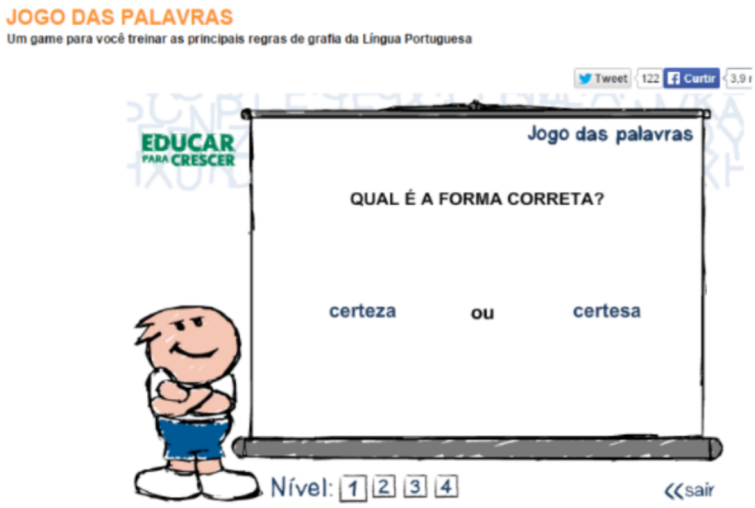


Figura 2.1 Jogo das Palavras. Fonte: *print screen* da tela do Jogo das Palavras

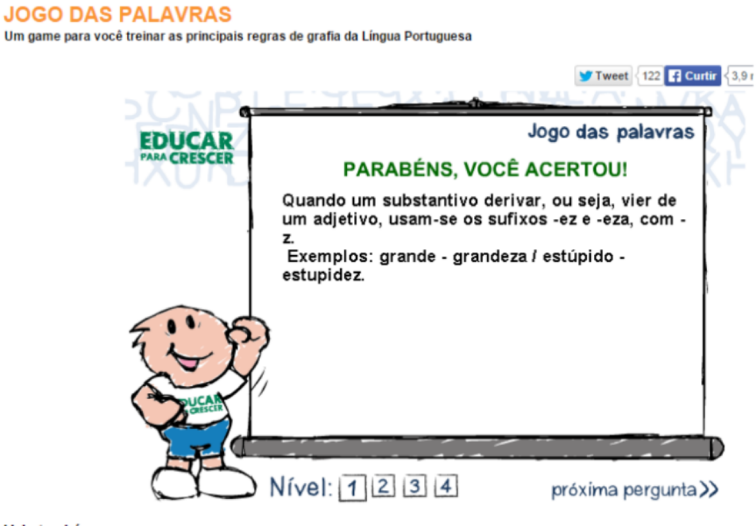


Figura 2.2 Jogo das Palavras. Fonte: *print screen* da tela do Jogo das Palavras.

2.4.2 JOE

JOE[2] (Jogo Ortográfico Educacional) é um jogo educativo gratuito desenvolvido para ser utilizado na plataforma Android e está disponível para *download* grátis na *play store*. Tem como objetivo ensinar o novo acordo ortográfico para os usuários. O aplicativo oferece a pos-

[2]Site do JOE. URL http://www.androidpit.com.br/app/br.rj.cefet.joe.app.

sibilidade em escolher as opções jogar e treinar, o que difere um do outro é que na opção jogar tem um tempo máximo para responder as perguntas. Como pode ser observado na figura 2.3, o usuário escuta a palavra através da opção "Ouvir", tem o campo onde possibilita escrever a palavra e a opção verificar palavra vai corrigir ou validar a resposta, contabilizando nos erros ou acertos.

Figura 2.3 JOE. Fonte: *print screen* da tela do JOE.

2.4.3 Pesca Letras

O Pesca Letras[3] é um jogo gratuito, em que aparece determinada imagem e de forma bem interativa. O jogador tem que pescar as letras correspondentes a essa imagem em um rio. As fases vão de 1 a 5 e o máximo de erros é 10 tentativas, se o jogador acabar com as chances, o jogo mostra a palavra certa e pula outra fase. A figura 2.4 ilustra a tela do jogo.

[3]Site do Pesca Letras. URL http://www.escolagames.com.br/jogos/pescaLetras/.

Figura 2.4 Pesca Letras. *print screen* da tela do Pesca Letras.

2.4.4 Forca dos Coletivos

Forca dos Coletivos[4] é um *game* gratuito baseado no jogo da forca. O objetivo do jogo é apresentar palavras na tela para que o jogador responda quais são seus coletivos. Existem 4 níveis de dificuldade e a cada palavra apresentada, o jogador possui 8 chances de acerto. A cada erro uma parte do corpo do personagem "enforcado" é desenhada na tela. A figura 2.5 demonstra a tela do jogo.

[4]Site do Forca dos Coletivos. URL http://educarparacrescer.abril.com.br/mini-jogos/forca/.

Figura 2.5 Forca dos Coletivos. *print screen* da tela do Forca dos Coletivos.

2.4.5 Fábrica de Palavras

O jogo gratuito Fábrica de Palavras[5] é bem simples e interativo. Na primeira fase é apresentada uma palavra e sua imagem, o jogador tem que completar essa palavra com algumas das letras que estão disponíveis na fábrica. São 5 palavras apresentadas. A 2.6 demonstra a tela do jogo. Na segunda fase, o jogador deve percorrer a fábrica com um robô buscando pelas letras que contém na palavra apresentada, como demonstrado na figura 2.7. São apresentadas 3 palavras e quando o jogo atinge seu objetivo ele volta para o menu contendo a opção de iniciar novo jogo.

[5]Site do Fábrica de Palavras. URL http://www.escolagames.com.br/jogos/fabricaPalavras/.

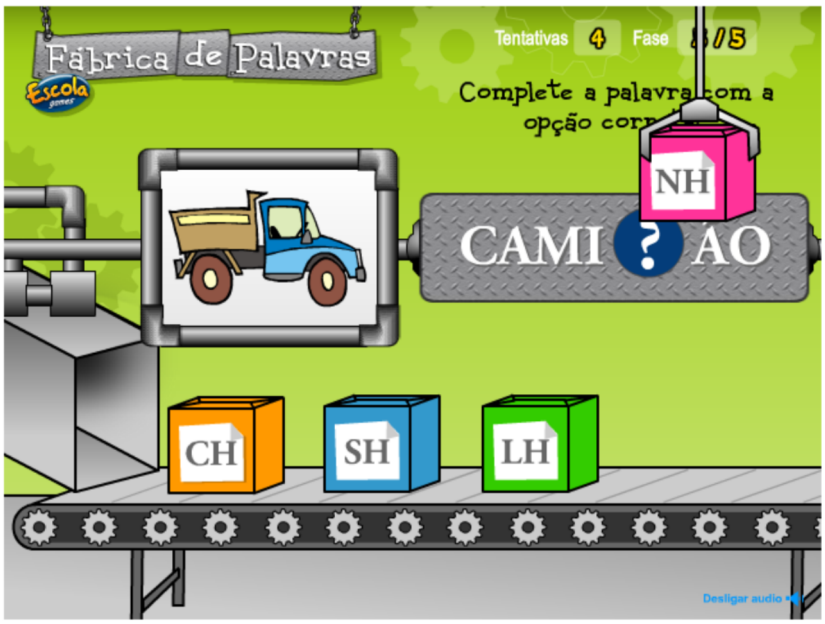


Figura 2.6 Fase 1 do Fábrica de Palavras. *print screen* da tela do Fábrica de Palavras.

Figura 2.7 Fase 2 do Fábrica de Palavras. *print screen* da tela do Fábrica de Palavras.

2.4.6 Alfabeto de Sabão

Alfabeto de Sabão[6] é um jogo gratuito que tem como finalidade ensinar as letras do alfabeto para o jogador. Uma bolha de sabão com uma letra fica vagando no cenário e quando a tecla correspondente a letra é utilizada a bolha estoura demonstrando que o jogador acertou a resposta. Também tem a opção de utilizar qualquer tecla para estourar a bolha. A figura 2.8 ilustra o jogo.

Figura 2.8 Alfabeto de Sabão. *print screen* da tela do Alfabeto de Sabão.

2.4.7 Como se Escreve?

Jogo totalmente gratuito e de fácil acesso. Nele, aparecem na tela duas palavras e a opção de reproduzir o áudio dessas palavras. O jogador escolhe a que achar correta e, se errar a resposta, uma nova tela é gerada com a correção, os níveis de dificuldade variam de 1 a 4. As figuras 2.9 e 2.10 exemplificam o cenário do "Como se Escreve?"[7]. Como demonstra a figura 2.9, o jogo apresenta a palavra escrita "incorretamente", o que pode confundir os usuários, podendo memorizar a palavra escrita de forma incorreta.

[6] Site do Alfabeto de Sabão. URL http://www.escolagames.com.br/jogos/alfabetoSabao/.
[7] Site do "Como se Escreve?". URL http://educarparacrescer.abril.com.br/como-se-escreve/.

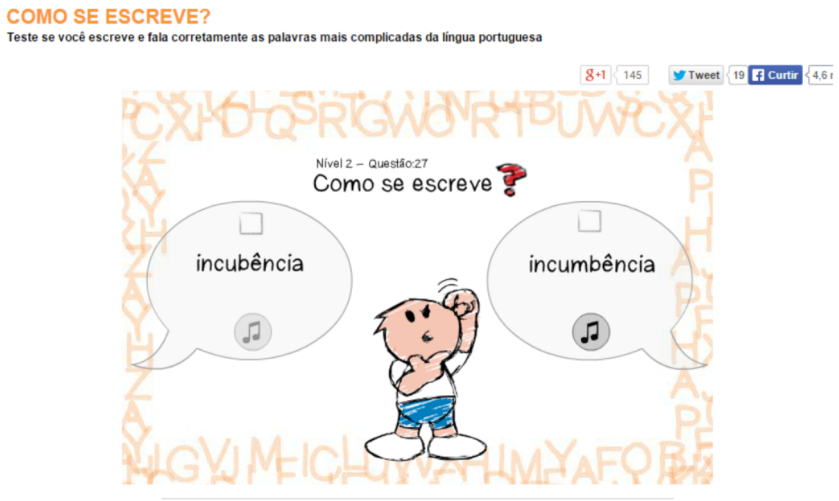


Figura 2.9 "Como se Escreve?". *print screen* da tela do "Como se Escreve?".

Figura 2.10 "Como se Escreve?". *print screen* da tela do "Como se Escreve?".

2.4.8 Jogo dos Substantivos

O objetivo do Jogo dos Substantivos[8] é fazer com que o usuário responda qual é o substantivo de da palavra apresentada na tela do jogo. Como pode ser observado na figura 2.11, o jogador tem que arrastar a figura que representa determinado substantivo para o local

[8]Site do Jogo dos Substantivos. URL http://www.soportugues.com.br/secoes/jogos/jogo.php?jogo=2.

indicado e clicar na opção conferir. O jogador pode errar no máximo 5 vezes.

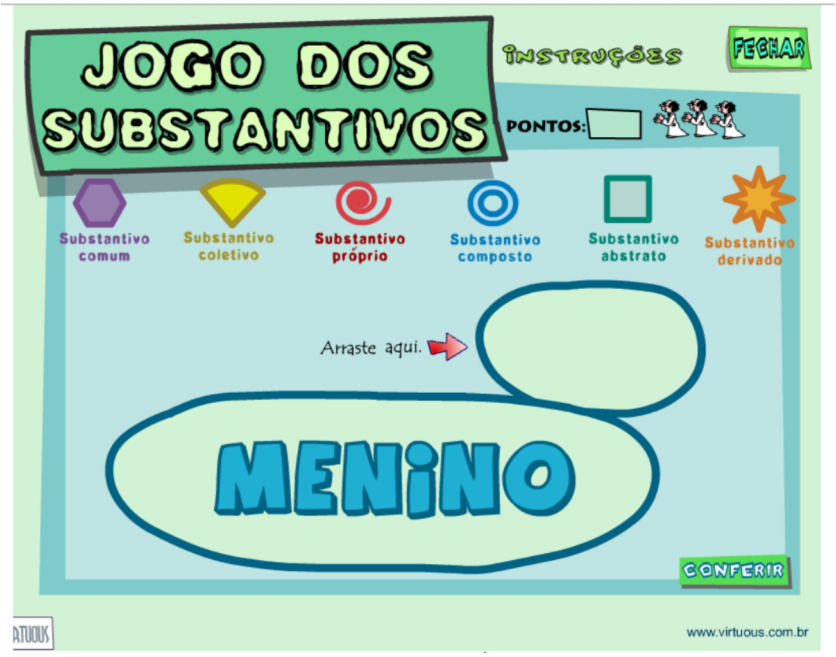


Figura 2.11 Jogo dos Substantivos. *print screen* da tela do Jogo dos Substantivos.

2.4.9 Descubra as Erradas

No jogo Descubra as Erradas[9], o jogador tem que selecionar as palavras que não estão graficamente corretas, clicando no círculo as que acharem incorretas e posteriormente clicando na opção conferir. Caso estiver incorreta a resposta, aparece uma mensagem na tela e o jogador volta na opção de escolher as palavras que estão escritas de maneira errada. Como demonstra a figura 2.12, o jogo apresenta palavras escritas "incorretamente", podendo confundir os usuários, eles podem memorizar as palavras escritas de maneira incorreta.

[9]Site do jogo Descubra as Erradas. URL http://www.soportugues.com.br/secoes/jogos/jogo.php?jogo=12.

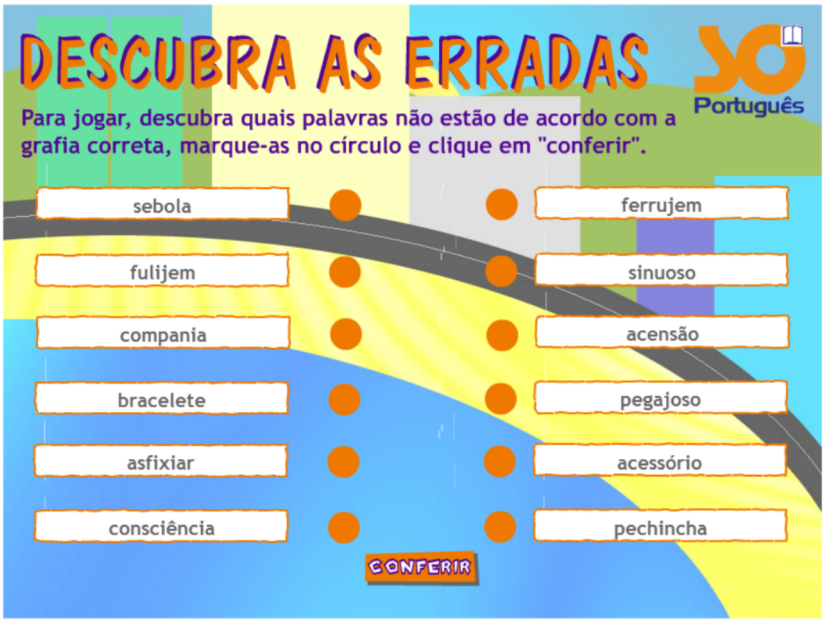


Figura 2.12 Descubra as Erradas. *print screen* da tela do Descubra as Erradas.

CAPÍTULO 3

Ferramentas Utilizadas

Este capítulo tem como objetivo descrever as ferramentas e técnicas utilizadas para que fosse possível o desenvolvimento do jogo digital ortográfico e a implementação do algoritmo de inteligência artificial baseado no SARSA para ajustar dinamicamente a dificuldade do jogo.

A seção 3.1 apresenta a ferramenta computacional Unity 3D e sua subseção 3.1.1 fala sobre os *Scripts* no Unity 3D. Na seção 3.3 e 3.2 é apresentada as ferramentas C# e SQLite, respectivamente. A seção 3.4 descreve a ferramenta CorelDRAW e sua subseção 3.4.1 descreve o suíte de programas CorelDRAW Graphics Suite X7. A seção 3.5 descreve o Ajuste Dinâmico de Dificuldade (DDA). Por último, a seção 3.6 descreve sobre a área de pesquisa de inteligência artificial, e suas subseções 3.6.1, 3.6.2 e 3.6.3, abordam sobre Aprendizado de Máquina, Aprendizagem por Reforço e a técnica de aprendizagem por reforço SARSA, respectivamente.

3.1 Unity 3D

O desenvolvimento de jogos digitais é um dos principais nichos de mercado no mundo da tecnologia. Com o avanço na qualidade de equipamentos tecnológicos nos últimos tempos, a demanda por novos *games* vem aumentando mais ainda, trazendo também a exigência de jogos bem produzidos e de qualidade. Para quem está começando a desenvolver jogos a escolha de uma boa ferramenta para desenvolvimento é de suma importância para produzir algo de qualidade e de maneira fácil e ágil.

O Unity 3D surgiu com intuito de simplificar o desenvolvimento de jogos. O Unity é um *game engine* (motor de jogo) que é utilizado para elaborar jogos digitais unindo imagens, áudios, animações, *scripts*, entre outros, criando ambientes e cenas de maneira simples e ágil. "O motor de jogo, também conhecido pela denominação em inglês *game engine*, é o mecanismo que controla a reação do jogo em função das ações do usuário" (Bittencourt e Giraffa, 2003, p. 4).

Extremamente poderoso e completo, o Unity é utilizado para desenvolver desde jogos simples em 2D até jogos realmente complexos e realísticos em 3D. Por ser uma ferramenta bastante utilizada pelos desenvolvedores de jogos, o Unity também tem como grande vantagem a disponibilidade de um acervo muito grande na internet de tutoriais, documentações, comunidades de perguntas e respostas e até mesmo treinamento ao vivo, facilitando o aprendizado do uso da ferramenta. O usuário pode entrar no site oficial do Unity que vai encontrar os *links* para o acesso desses meios de aprendizagem.

O Unity possui parceiros como a Microsoft, Samsung, Nintendo, Sony, entre diversos outras empresas gigantes no ramo da tecnologia. Um estudo realizado sobre a quantidade de desenvolvedores registrados no Unity no período de 2012 a 2015 comprova a popularidade da ferramenta, de 2012 a 2015 ouve um crescimento de mais de 3 milhões de usuários registrados, como ilustra a figura 3.1.

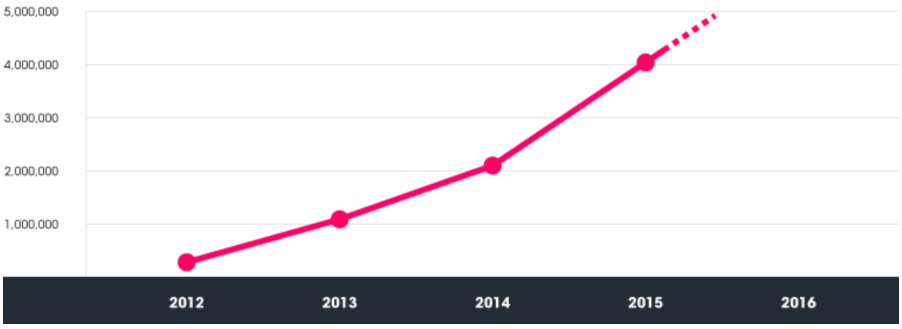


Figura 3.1 Pesquisa sobre os desenvolvedores registrados no Unity no período de 2012 à 2015 (Unity, 2015).

O Unity possibilita o desenvolvimento de jogos em diferentes plataformas. Os sistemas desenvolvidos podem ser exportados para outras plataformas sem precisar programar tudo novamente. O Unity oferece a opção de "traduzir" o sistema de acordo com a plataforma escolhida pelo usuário, sendo que este trabalha com as mais importantes plataformas existentes, como: Android, iOS, PS3, PS4, Mac, Linux, Windows, Windows Phone 8, etc.

Para instalar a última versão do Unity, que é o Unity 5, o usuário deve fazer o *download* da ferramenta no *site* oficial do mesmo, podendo optar pela versão gratuita e a versão profissional. A versão profissional cobra uma taxa a partir de R$ 75,00 mês ou pode-se comprar a licença por R$ 1500,00. Se o usuário fizer o *download* da versão gratuita, ele ainda recebe 30 dias para se utilizar a ferramenta na versão profissional (Unity3D, 2016). Para projetos de pequeno ou médio porte é aconselhável utilizar a versão gratuita, pois esta é uma ferramenta

bastante poderosa.

Existem outras ferramentas de desenvolvimento de jogos digitais, como: Native Code Only, Custom Solutlon, Coco2d, Adobe Air, dentre outras (Unity, 2015). A ferramenta escolhida para ser utilizada neste trabalho foi a Unity 3D.

3.1.1 Scripts no Unity 3D

Os *scripts* são conjuntos de instruções implementadas em linguagens de programação. Sua utilização no Unity objetiva possibilitar o desenvolvedor atribuir sequência de tarefas e comandos aos objetos e cenas do jogo. Para movimentar uma animação na cena, por exemplo, o desenvolvedor tem que criar um *script* implementando as ações dessa animação.

De acordo com Unity3D (2015), os *scripts* são essenciais em todos os jogos. Até mesmo o jogo mais simples desenvolvido precisa de *scripts* para implementar a transição das telas e organizar os eventos no jogo. Se a tela de um jogo possuir a animação de um pássaro, por exemplo, é necessário desenvolver um *script* para que esse possa se movimentar sozinho. Os *scripts* são utilizados para produzir efeitos gráficos, controlar o comportamento físico de objetos ou até mesmo implementar um sistema de inteligência artificial personalizado para o jogo.

O Unity utiliza a biblioteca “Mono” que permite ao usuário desenvolver seus *scripts* em três linguagens de programação diferentes: Boo, JavaScript e C#. O desenvolvedor também tem a possibilidade em utilizar mais de uma linguagem de programação no mesmo jogo. Uma maneira em criar *script* no Unity é clicar no canto superior esquerdo na opção *Assets*, posteriormente na opção *Create* e escolher dentre as linguagens, C# Script ou JavaScript ou Boo Script. A linguagem de programação C# foi utilizada neste trabalho.

3.2 C#

O C# é uma linguagem de programação que integra a plataforma .NET da Microsoft. Essa plataforma tem como principal objetivo o desenvolvimento de serviços *Web Services* XML, que são serviços que buscam ter diversos sistemas conectados na rede (Liberty, 2005). Ainda que esses sistemas sejam desenvolvidos em plataformas diferentes, a proposta do *Web Services* XML é basicamente conceder a todas as aplicações desenvolvidas para qualquer fim

específico se comuniquem e troquem informações, mesmo sendo de sistemas operacionais ou linguagens de programação diferentes.

O C# foi produzido pela Microsoft em 1999 pelo Anders Hejlsberb e sua equipe. A ideia era desenvolver a linguagem buscando unir as vantagens principais de várias linguagens já existentes, como o C++ e o JAVA, com a intenção de adicionar recursos novos e aprimorar suas implementações. "A linguagem C# (pronuncia-se C Sharp) faz parte desse conjunto de ferramentas oferecidas na plataforma .NET e surge como uma linguagem simples, robusta, orientada a objetos, fortemente tipada e altamente escalável" (Lima e Reis, 2002, p. 3–4).

Sabendo que a linguagem é totalmente orientada a objetos, todas as variáveis existentes fazem parte de alguma classe e, para que o programador não faça manipulações impróprias e ocasione um erro dificilmente de ser descoberto, a linguagem C# foi desenvolvida como sendo fortemente tipada. O *site* oficial da Microsoft[1] disponibiliza toda a documentação da linguagem de forma fácil de ser acessada o que auxilia bastante o usuário.

3.3 SQLite

O SQLite é um banco de dados de pequeno porte desenvolvido para ser de código aberto, ou seja, qualquer usuário tem o direito em usar, copiar ou modificar o código original do SQLite. Seu uso é livre tanto em projetos privados, quanto em projetos comerciais. Coelho e Prado (2015) destacam que o SQLite funciona de forma correspondente a um gerenciamento de banco de dados, sendo que existem funções diversas, como exemplo, controlar muitos bancos de dados e concomitantemente suas tabelas, sendo uma biblioteca de banco de dados fundamentada no SQL (*Structured Query Language*).

O SQLite gera um banco de dados que pode ser utilizado junto com a aplicação desenvolvida, sem precisar utilizar banco de dados Cliente/Servidor. Outra vantagem é que esse não requer muita memória, o que é um fator importante para projetos que precisam ser rápidos e leves. Por outro lado, não se recomenda o uso do SQLite para projetos de grande porte, pois o mesmo não suporta uma quantidade grande de acessos, de dados e também não funciona como banco de dados cliente/servidor.

A biblioteca SQLite vai coordenar os elementos diretamente do sistema de arquivos.

[1] Site oficial da Microsoft com toda documentação sobre C#. Para mais detalhes, veja https://msdn.microsoft.com/en-us/library/67ef8sbd.aspx.

Então não é necessário instalar e configurar para que o SQLite funcione. O SQLite oferece suporte a transações ACID (Atomicidade, Consistência, Isolamento e Durabilidade) e possui uma prática de armazenamento protegida.

O SQLite é amplamente utilizado na plataforma Android. "A utilização do SQLite em Android não requer nenhuma configuração inicial, apenas é necessário especificar a instrução SQL para gerar o banco de dados e ele é criado automaticamente." (Comachio, 2012, p. 25). SQLite pode ser utilizado em várias linguagens, dentre elas, o C#, C++, PHP, etc. Neste trabalho foi utilizada a linguagem C#, descrita na seção a seguir.

Galvão *et al.* (2009) destaca outras ferramentas importantes para gerenciamento de banco de dados, como: Firebird, MySQL, PostgreSQL e MaxDB. Neste trabalho foi utilizado o SQLite, pois esta ferramenta não requer muita memória, o que é necessário para que o jogo não fique "pesado" no dispositivo móvel do usuário.

3.4 CorelDRAW

A ferramenta CorelDRAW foi desenvolvida pela Corel há mais de 25 anos. O primeiro a ser desenvolvido foi o CorelDRAW 1.0 em 1989 que foi o pioneiro da época e revolucionou o mercado de *softwares* gráficos. O usuário podia manipular e criar logotipos, capas de livros, CDs, dentre outros. Desde seu primeiro desenvolvimento a ferramenta está a cada vez mais evoluindo e inovando com o decorrer do tempo foram 18 atualizações e modificações na ferramenta. Hoje é disponibilizado o CorelDRAW Graphics Suite X7, que é a versão mais recente e completa do CorelDRAW.

3.4.1 CorelDRAW Graphics Suite X7

Com diversas funcionalidades, o CorelDRAW Graphics Suite X7 é um suíte de programas que foi desenvolvido para ser utilizado somente em sistema operacional Windows. Neste suíte, o usuário encontra variados programas, como o Corel PHOTO-PAINT X7, que possui vários recursos avançados para edição de imagens. O CorelDRAW X7 é utilizado para layout de páginas e ilustração vetorial. O Corel CAPTURE X7 captura imagens da tela do computador do usuário com apenas um clique. O Corel CONNECT X7 é um browser que pode acessar conteúdos digitais no computador em que ele está instalado. O Corel PowerTRACE X7 é uma

ferramenta utilizada para rastrear bitmap para vetor. No PhotoZoom Pro 3 o usuário tem opção de aumentar as imagens digitais. Corel WebSite Creator é utilizado para criar e editar design de sites. ConceptShare é utilizada para compartilhar os arquivos on-line. O BarCode possibilita a criação de código de barras. O Bitstream Font Navigator gerencia fontes e o Duplexing Wizard possibilita imprimir frente e verso manualmente.

A versão X7 do CorelDRAW está disponível para *download* no site oficial do CorelDRAW. Nele, o usuário tem a opção de escolher a versão de avaliação, que é uma versão gratuita com tempo determinado de 30 dias para que o usuário conheça a ferramenta. A versão completa de *download* o usuário tem que desembolsar por volta de R$ 2300,00 para obter a ferramenta, podendo comprar também as atualizações por cerca de R$ 850,00 cada. Também há a opção de assinatura, em que o usuário paga uma taxa mensal de R$ 69,90 ou anual de R$ 899,00 para obter a ferramenta. Informações obtidas no site do produto[2].

Existem outras ferramentas para manipular imagens, como: Adobe Photoshop, The Gimp, Pixlr Editor, Illustrator, dentre outras. A ferramenta utilizada neste trabalho foi a CorelDRAW Graphics Suite X7.

3.5 Ajuste Dinâmico de Dificuldade

O Ajuste Dinâmico de Dificuldade (DDA, do inglês *Dynamic Difficulty Adjustment*), que também pode ser chamado por balanceamento dinâmico de dificuldade, tem como objetivo ajustar o jogo e apresentar os desafios próprios para cada jogador. Segundo Kuang (2012), a ideia principal do ajuste dinâmico é em ter componentes no jogo disponíveis para medir a habilidade e o desempenho do jogador, e ajustar a dificuldade ao decorrer da jogabilidade com o objetivo em disponibilizar uma experiência mais consistente e divertida.

O nível de desafio é que determina a qualidade de entretenimento do jogo digital. Para que o usuário se divirta e entretenha, é preciso que o jogo seja balanceado de forma que não seja tão fácil vencer os desafios, transformando-o em um jogo tedioso, e também nem tão difícil de forma que o jogador nunca consiga vencer a fase ou desafio proposto, gerando frustação e desânimo em jogar. O usuário tem que se sentir interessado em evoluir e concluir o jogo e sempre manter o interesse em jogar.

A figura 3.2 demonstra claramente os possíveis balanceamentos de um jogo. O melhor

[2]Site oficial do CorelDRAW. Para mais detalhes, veja http://www.coreldraw.com/br/.

balanceamento para um jogo não é o balanceamento linear. No balanceamento linear, o nível de dificuldade é igualmente o nível de destreza e malícia desenvolvida pelo jogador. O mais esperado é que o balanceamento seja como o não previsível, pois o jogador deve passar por momentos difíceis e momentos fáceis ao decorrer de sua jogabilidade. "Essa alternância no nível de dificuldade pode ser implementada em jogos de diferentes maneiras, como a disponibilização de novas armas (momento de relaxamento) após a superação de um chefe-de-base" (Andrade *et al.*, 2006, p.16).

Figura 3.2 Representação do balanceamento de jogos digitais (Andrade *et al.*, 2006).

Existem diversos métodos que propõem resolver o problema de ajuste dinâmico de dificuldade em um jogo. Monteiro (2009) destaca alguns métodos, como: manipulação de parâmetros, algoritmos genéticos coevolucionários, rtNEAT, aprendizagem por reforço e *scripts* dinâmicos. Neste trabalho foi utilizado o método de aprendizagem por reforço SARSA, descrito na seção 3.6.3, para implementar o balanceamento dinâmico de dificuldade do jogo digital ortográfico.

3.6 Inteligência Artificial

A inteligência artificial é uma área de pesquisa da Ciência da Computação que surgiu logo após a segunda guerra mundial. Tem como objetivo implementar sistemas que consigam aprender e se desenvolver por conta própria, a ideia é que o sistema computacional que utilize dessa técnica atue e pense como os seres humanos. A partir de diversas informações coletadas, o sistema tem a capacidade de criar novas informações e deduções.

Segundo Pereira (1988), a inteligência artificial é uma disciplina científica que utiliza as capacidades de processamento de símbolos computacionais com a intenção de encontrar maneiras para automatizar atividades perceptivas, cognitivas e manipulativas, por meio de uma máquina. Existem inúmeros sistemas computacionais na atualidade que realizam tarefas utilizando conhecimento e aprendizado similar ao raciocínio humano, porém, falta muito para que sejam comparados com a inteligência de uma pessoa.

A influência da inteligência artificial na atualidade é enorme. Segundo Russel e Norvig (2004), a IA (inteligência artificial) possui uma enorme variedade de subcampos, como: jogos de xadrez, demonstração de teoremas matemáticos, direção de um automóvel em um ambiente totalmente movimentado. A IA é utilizada até mesmo em diagnóstico de doenças. Consiste em uma área de estudo expressiva para qualquer tarefa intelectual, sendo verdadeiramente um campo universal.

A inteligência artificial é também muito utilizada em jogos digitais, a IA tem como objetivo implementar uma certa "inteligência" aos componentes do jogo. Imagine um jogo digital de xadrez, onde o usuário joga contra um oponente controlado pela máquina. É inevitável a implementação de inteligência artificial no *game* para que o mesmo se adapte ao estilo de jogo de seus usuários. Caso contrário, o usuário poderá aprender as técnicas do oponente e vencer sempre. "Os jogos de computador por seu caráter multidisciplinar prestam-se a demonstrar a validade das mais diversas técnicas de inteligência artificial." (Tatai, 2003, p. 2).

Um conceito abordado constantemente em inteligência artificial são os agentes. "Um agente é tudo o que pode ser considerado capaz de perceber seu ambiente por meio de sensores e de agir sobre esse ambiente por intermédio de atuadores" (Russel e Norvig, 2004, p. 33). A figura 3.3 exemplifica a interação dos agentes com os ambientes.

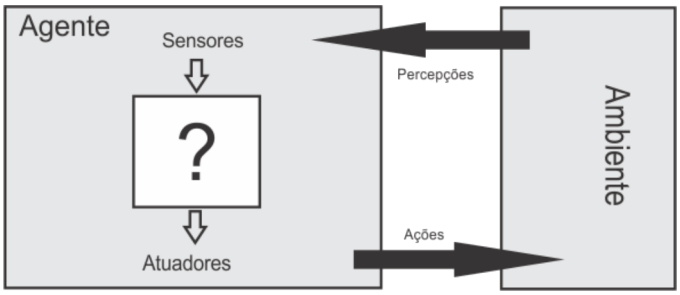


Figura 3.3 Agentes interagindo com ambientes atráves de sensores e atuadores. Adaptada de: Russel e Norvig (2004)

Dentre os subcampos de inteligência artificial, (Barto e Sutton, 2012) destaca o Aprendizado de Máquina. A área tem solucionado diversos problemas existentes em inteligência artificial e desenvolvido notáveis aplicações, sendo amplamente utilizada. A subseção 3.6.1 aborda o Aprendizado de Máquina.

3.6.1 Aprendizado de Máquina

O aprendizado pode ser adquirido por repetição ou por cognição. No aprendizado por repetição o aprendiz memoriza fatos ou sequência de ações ocorridas ao decorrer de sua vida. A informação "Belo Horizonte é a capital de Minas Gerais" é aprendida por repetição e nada pode generalizar dessa informação. Um *software* aprende facilmente quando não se envolve aprendizado cognitivo, a partir do momento que for salvo em seu banco de dados a informação acima. Quando perguntado ao *software* qual a capital de Minas Gerais, logo responderá Belo Horizonte. Na aprendizagem cognitiva, as informações conhecidas são analisadas, relacionadas e organizadas de forma que outras informações novas são descobertas. As informações podem ser generalizadas de forma que conhecimentos novos sejam adquiridos.

A aprendizagem de máquina busca desenvolver técnicas computacionais para que os *softwares* não somente memorizem as informações, mas que consigam generalizar e transformar dados ou informações conhecidas em diversas outras informações novas e importantes. Portanto, objetiva assemelhar-se com a aprendizagem cognitiva. Mitchell (1997) define que, embora não se saiba fazer com que as máquinas aprendam tão bem como os seres humanos, nos últimos anos, muitas aplicações baseadas nas técnicas de aprendizagem de máquina estão sendo desenvolvidas com sucesso, que vão desde mineração de dados, *softwares* personalizados, até filtragem de usuários, por exemplo.

Henke *et al.* (2011) afirma que sistemas de aprendizagem confiáveis são de suma importância estratégica nas mais variadas áreas de aplicação, dado que existem diversas tarefas que não podem ser solucionadas por meio de técnicas de programação clássicas. Sendo assim, as técnicas de aprendizagem de máquina são amplamente utilizadas para o desenvolvimento de sistemas de aprendizagem confiáveis, em busca da solução dos mais diversos problemas. Russel e Norvig (2004) distingue o campo de aprendizagem de máquina em três casos: aprendizagem supervisionada, por reforço e não-supervisionada. Neste trabalho, foi utilizada a técnica de aprendizagem por reforço, detalhada na subseção 3.6.2.

3.6.2 Aprendizagem por reforço

A aprendizagem por reforço objetiva maximizar o desempenho do agente no ambiente em que está inserido, baseando-se apenas em reforços positivos ou negativos. Segundo Kaelbling *et al.* (1996), aprendizagem por reforço é um problema em que um agente deve aprender o comportamento adequado através de interações de tentativa e erro com o ambiente, não há a necessidade de especificar como a tarefa deve ser alcançada, o agente aprende através de recompensas e punições.

Russel e Norvig (2004) reforça que um agente em aprendizagem por reforço não precisa ser informado por um instrutor sobre o que fazer no ambiente. O agente deve aprender apenas com os reforços recebidos após uma ação ou conjunto de ações. Uma multa de alto valor por bater na traseira de um carro é um exemplo de reforço, indica ao agente que seu comportamento foi indesejável.

Segundo Barto e Sutton (2012) é possível identificar quatro elementos formadores básicos de um sistema de aprendizagem por reforço, além do agente e ambiente. São eles:

- Uma política π, utilizada para mapear os estados s e as ações a em uma política de estados e ações. Define o caminho do agente para se comportar em um determinado momento, ou seja, os estados percebidos no ambiente são mapeados para formular ações a serem tomadas quando o agente estiver naqueles estados;
- Uma função de recompensa ou reforço R(s, a), que é o reforço positivo ou negativo recebido logo após um comportamento em uma ação a no estado s. O objetivo de um agente é sempre buscar otimizar o reforço total que ele recebe a longo prazo, o reforço vai definir quais são os bons e os maus eventos para o agente. Por exemplo, em um jogo

de gamão, o reforço é dado no final de uma partida, positivo caso vença ou negativo caso perca;

- Uma função de valor V(s), que é quanto o agente espera de reforço futuro saindo de algum estado s qualquer, ou uma função de valor Q(s, a), que é quanto o agente espera de reforço futuro saindo de um determinado estado s e escolhendo uma ação a qualquer. Utilizado para especificar a longo prazo uma política que seja adequada.
- Opcionalmente, um modelo de ambiente que vai imitar o comportamento do ambiente em um determinado estado-ação em que o agente estiver, o modelo pode prever o próximo estado resultante e o próximo reforço, por exemplo.

Dentre as técnicas de aprendizagem por reforço, (Monteiro e Ribeiro, 2004) destaca as principais, são elas: Q-Learning, SARSA, Q(λ), Dyna. Neste trabalho, foi utilizado a técnica de aprendizagem por reforço SARSA, abordada na subseção 3.6.3.

3.6.3 SARSA

O SARSA (*State-Action-Reward-State-Action*), que significa Estado-Ação-Recompensa-Estado-Ação, foi descrito na década de 90 por Rummery e Niranjan (1994), na universidade de Cambridge, e tem seu nome originado devido ao fato de que a sua atualização no algoritmo envolve um estado s_t, uma ação a_t, uma recompensa r_{t+1}, um estado posterior s_{t+1} e uma ação posterior a_{t+1}. Como demonstrado na figura 3.4.

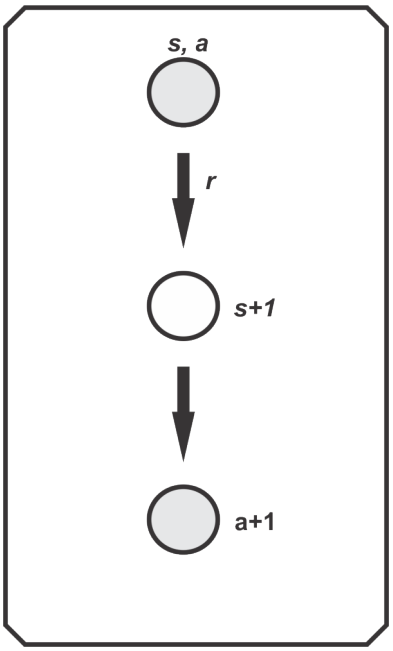


Figura 3.4 Diagrama de backup do algoritmo SARSA. Adaptada de: Monteiro (2002)

Seguindo os conceitos de Barto e Sutton (2012) e Alves (2010), a regra de atualização do SARSA é definida como:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r(s_t, a_t) + \gamma\, Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

Sendo:

- $s_t \leftarrow$ Estado corrente;
- $a_t \leftarrow$ Ação corrente;
- $s_{t+1} \leftarrow$ Estado novo;
- $a_{t+1} \leftarrow$ Ação nova;
- $r(s_t, a_t) \leftarrow$ Reforço recebido depois de ocorrer a ação a_t em um estado s_t;
- $\gamma \leftarrow$ Taxa de desconto ($0 \leq \gamma \leq 1$), para considerar mais a recompensa atual basta esse valor se aproximar ou igualar a zero, para considerar as recompensas ao decorrer do tempo basta se aproximar ou igualar a 1;
- $\alpha \leftarrow$ Taxa aprendizagem;

- $Q(s_t, a_t) \leftarrow$ Valor de uma ação a em um estado s;
- $Q(s_{t+1}, a_{t+1}) \leftarrow$ Valor de uma ação a em um estado s selecionada através de um método de exploração;

Pode-se definir o SARSA como uma modificação do *Q-Learning*, (Barto e Sutton, 2012). A política de atualização do SARSA é *On-Policy*, ou seja, nem sempre a maior recompensa calculada é que vai ser estabelecida para atualizar a função $Q(s_t, a_t)$. Já o *Q-Learning* tem sua política de atualização *Off-Policy*, "...a regra de atualização de um algoritmo off-policy pode atualizar sua política baseado em suposições, supostas ações que são diferentes daquelas seguidas pela política atual" (Prauchner, 2014, p. 27). Portanto, o SARSA difere do *Q-Learning* por possuir uma política que não utiliza a maximização das ações, mas atualiza a função $Q(s_t, a_t)$ através de um método de exploração.

A seção 4.2 aborda sobre a implementação do algoritmo de aprendizagem por reforço baseado no SARSA, utilizado para ajustar dinamicamente a dificuldade do jogo digital ortográfico.

Capítulo 4

Desenvolvimento

O presente capítulo objetiva abordar os passos para que fosse possível o desenvolvimento do jogo digital ortográfico, bem como aqueles para que também fosse possível a implementação do algoritmo de inteligência artificial baseado no SARSA para balancear dinamicamente a dificuldade do jogo em questão.

A seção 4.1 introduz sobre o jogo digital ortográfico, as subseções 4.1.1, 4.1.2, 4.1.3 e 4.1.4 abordam sobre o funcionamento do jogo, plataforma escolhida, estrutura do jogo e cenas do jogo, respectivamente. A seção 4.2 aborda sobre a implementação do algoritmo de inteligência artificial baseado no SARSA para balancear dinamicamente a dificuldade do jogo *Laça Palavras*.

4.1 Jogo Digital Ortográfico

A ideia do desenvolvimento de um jogo digital ortográfico surgiu a partir da observação das dificuldades em que algumas crianças tem na aprendizagem da escrita correta de palavras que possuem letras ou dígrafos concorrentes, que representam o mesmo som, no mesmo contexto, principalmente quando essas palavras não contenham regularidades de escrita. A palavra "casamento" é um exemplo: a letra "s" pode ser claramente confundida com a letra "z" e não existe regra que determine a escolha de uma outra letra, a criança tem apenas que memorizar sua forma de escrita.

O jogo digital ortográfico foi desenvolvido juntamente com Leal (2016) e recebeu o nome de *Laça Palavras*. A subseção 4.1.1 explicará com detalhes o funcionamento do jogo.

A versão gratuita do Unity 5 foi escolhida para ser utilizada no desenvolvimento do jogo digital, pelo fato de possuir uma versão gratuita, ser uma ferramenta bastante utilizada pelos desenvolvedores e pela disponibilidade de diversos meios de aprendizagem na *internet*. A seção 3.1 aborda com detalhes sobre o Unity.

Grande parte dos *sprites* e das imagens utilizadas no *Laça Palavras* foram retirados do *site* pixabay[1], para criar ou modificar alguma imagem ou *sprite* foi utilizado a versão de avaliação gratuita do suíte de programas CorelDRAW Graphics Suite X7 que contêm diversas ferramentas para edição e desenvolvimento de imagens, a seção 3.4.1 aborda sobre a ferramenta em questão.

Todos os áudios utilizados no *Laça Palavras* foram retirados da biblioteca de áudios do youtube[2] e o site mp3cut[3] foi utilizado para recortar os áudios.

Todas as palavras e frases utilizadas no *Laça Palavras* estão disponíveis no Apêndice F.

4.1.1 Funcionamento do Jogo Digital Ortográfico

O *Laça Palavras* é um jogo direcionado ao público infantil, no qual o jogador tem que completar as palavras apresentadas na tela escolhendo uma das letras disponibilizadas como resposta. O objetivo principal é auxiliá-lo no processo de aprendizagem da ortografia.

Quando o jogador criar seu *login*, o mesmo tem a possibilidade em escolher um *cowboy* ou uma *cowgirl* para ser representado. O objetivo do personagem é laçar a caixa com a letra ou dígrafo que o jogador escolher para completar a palavra. Para que o criança entenda o funcionamento do jogo, no momento inicial, dois quadros são apresentados na tela com as informações de ajuda, como demonstra a figura 4.1, quando a primeira pergunta for respondida os quadros somem da tela.

[1] pixabay é um site que fornece mais de 40 mil imagens sem direitos autorais para utilização comercial ou pessoal. Para mais detalhes, veja https://pixabay.com/.

[2] A biblioteca de áudios do youtube disponibiliza mais de 150 mil áudios para *download* gratuito. Para mais detalhes, veja https://www.youtube.com/audiolibrary/music.

[3] O *site* mp3cut disponibiliza uma ferramentas *online* gratuita para recortar áudios. Para mais detalhes, veja https://mp3cut.net/pt/.

Figura 4.1 Tela inicial do jogo *Laça Palavras*.

Para auxiliar a criança no momento de completar as palavras, o jogo disponibiliza um botão com formato de nota musical para executar um áudio de um menino (se o personagem escolhido for o *cowboy*) ou de uma menina (se o personagem escolhido for uma *cowgirl*) pronunciando uma frase que contextualiza a palavra a ser completada. A frase "O osso do meu cachorro é grande!" é apresentada no momento em que aparecer a palavra "Osso" na tela, por exemplo. O jogador pode escutar o áudio quantas vezes quiser antes de completar a palavra, basta pressionar o botão.

Este recurso foi implementado no jogo com objetivo de possibilitar que as crianças notem que ouvir o som da palavra não ajuda na escolha do dígrafo ou letra, sendo um diferencial do *Laça Palavras*. Atualmente, poucos jogos digitais ortográficos disponibilizam a opção de apresentar um áudio com a palavra a ser respondida, como pode ser observado na seção 2.4, sendo que, nenhum dos jogos listados nesta seção possui o recurso de apresentar um áudio com uma frase contextualizando a palavra.

Para manter o desempenho e a motivação do jogador, este recebe uma recompensa. A cada 10% de acertos, uma parte de uma casa com formato de quebra cabeças é montado no canto esquerdo da tela do jogo, como demonstra a figura 4.2. Como o objetivo é fazer com que o usuário responda todas as perguntas, a casa só estará completa quando todas as palavras forem respondidas corretamente. O jogo não repete palavras que já foram completadas de modo correto.

Figura 4.2 Tela do jogo *Laça Palavras*.

O desenvolvimento e aplicação de um algoritmo de inteligência artificial no jogo objetiva ajustar dinamicamente a dificuldade do mesmo, para que se torne mais interessante e divertido, assunto abordado na seção 4.2. O jogo possui quatro níveis, sendo que sua dificuldade aumenta a cada nível, pois cada nível possui um conjunto de palavras da sua classe e uma porcentagem de palavras dos níveis anteriores. O algoritmo de inteligência artificial determinará a transição de níveis durante o jogo.

Os níveis contêm palavras com letras ou dígrafos concorrentes, que representem o mesmo som, no mesmo lugar, e que não existam regras de escrita para essas palavras, segue abaixo o conjunto de letras e dígrafos utilizados:

- Nível 1: Palavras que possuem o dígrafo “SS” e a letra “Ç” concorrentes;
- Nível 2: Conjunto composto por 90% de palavras com as letras “U” e “L” concorrentes, e 10% de palavras do nível 1;
- Nível 3: Conjunto composto por 80% de palavras com as letras “G” e “J” concorrentes, 10% de palavras do nível 2 e 10% de palavras do nível 1;
- Nível 4: Conjunto composto por 70% de palavras com as letras “S” e “Z” concorrentes, 10% de palavras do nível 3, 10% do nível 2 e 10% do nível 1;

4.1.2 Plataforma Escolhida

A plataforma Android foi estabelecida para o desenvolvimento do *Laça Palavras*, seu destaque frente as outras plataformas foi pelo fato de ser gratuita, ou seja, não é necessário investimentos para desenvolver projetos e, principalmente, por ser amplamente utilizada e aceita pelos usuários de tecnologias móveis, a subseção 2.2.1 apresenta detalhes desta plataforma.

Os requisitos necessários para que o *Laça Palavras* funcione no aparelho do usuário é que o mesmo utilize uma plataforma Android com a versão 2.3.1 ou superior, pois versões anteriores não executam o jogo, e possuir no mínimo 256MB de memória de armazenamento. O requisito desejado é que possua uma memória RAM de no mínimo 512MB.

4.1.3 Estrutura do Jogo Digital Ortográfico

A janela *Project* do Unity 3D é uma *interface* para estruturar os diversos recursos que são utilizados no jogo. Esses recursos são chamados de *Assets*, como exemplo, as animações, áudios, *scenes*, *scripts*, etc. O desenvolvedor pode criar uma pasta para cada tipo de recurso do jogo dentro da pasta *Assets*, de forma organizada e de fácil manipulação. A figura 4.3 apresenta a janela *Project* com a estruturação dos recursos na pasta *Assets*.

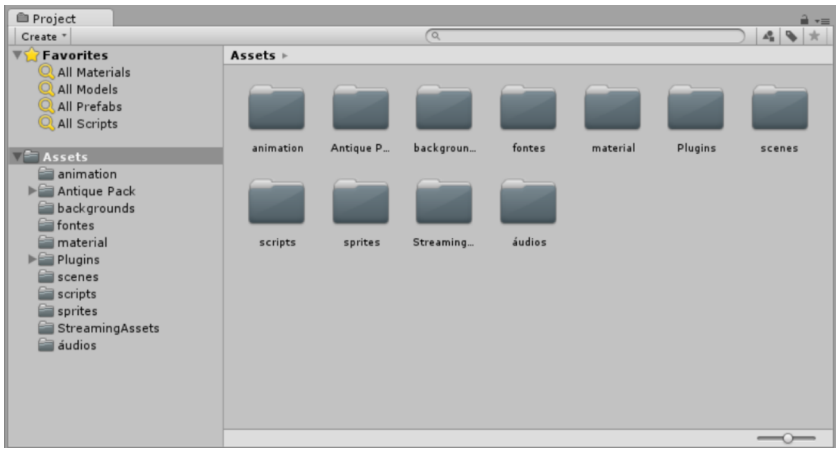


Figura 4.3 Estrutura do projeto *Laça Palavras*.

Para que um *assets* faça parte de determinada cena do jogo o desenvolvedor tem que clicar no recurso desejado e arrastar para a *Hierarchy*. Cada cena possui uma *Hierarchy* que vai conter e exibir todos os seus elementos, através dessa janela o desenvolvedor pode visualizar

e organizar a hierarquia dos elementos. A figura 4.4 apresenta a *Hierarchy* da cena inicial do *Laça Palavras*.

Figura 4.4 Aba *Hierarchy*.

4.1.3.1 Animations

Cada animação do *Laça Palavras* foi desenvolvida através de diversas imagens agrupadas em sequência, os chamados *sprites*. A animação do pássaro, por exemplo, foi desenvolvida através de *sprites* do pássaro com as asas em posições diferentes. Essas imagens são colocadas em sequência transformando em um *loop* dando a impressão que o pássaro está batendo as asas. A figura 4.5 ilustra a pasta *Animation* do jogo, essa pasta possui todas as animações desenvolvidas e essas animações podem fazer parte de qualquer cena do jogo, basta o desenvolvedor clicar e arrastar para a *Hierarchy* da cena desejada.

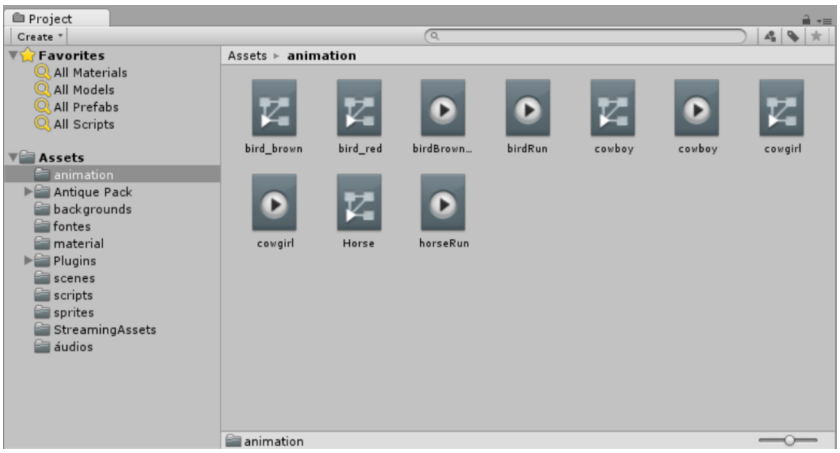


Figura 4.5 Pasta Animation.

Para controlar as animações é necessário sua manipulação através de *scripts*, a figura 4.6 exemplifica a implementação da função *FixedUpdate()* na cena inicial do *Laça Palavras* com objetivo de controlar os movimentos das nuvens e do pássaro. A *FixedUpdate()* é uma função que é chamada a cada *frame* por segundo.

```
// Vai ser chamada a cada frame por segundo!
void FixedUpdate () {
    // limite do movimento dos objetos!
    if (xCloud1 >= 9.52f) xCloud1 = -12.52f;
    if (xCloud2 >= 9.52f) xCloud2 = -12.52f;
    if (xBird >= 9.52f) xBird = 15.00f;

    // Velocidade dos movimentos dos objetos!
    xBird += 0.015f;
    xCloud1 += 0.009f;
    xCloud2 += 0.009f;

    //Setando nos objetos!
    cloud1.transform.position = new Vector2(xCloud1, yCloud1);
    cloud2.transform.position = new Vector2(xCloud2, yCloud2);
    bird.transform.position = new Vector2(xBird, yBird);
}
```

Figura 4.6 Função *FixedUpdate()*.

4.1.3.2 Backgrounds

A pasta *backgrounds* é utilizada para armazenar todas as imagens que fazem parte do jogo, como ilustra a figura 4.7. A imagem *fim de jogo* por exemplo, está armazenada na pasta *backgrounds* e é mostrada quando o jogador responde todas as perguntas do jogo corretamente.

Figura 4.7 Pasta Backgrounds.

4.1.3.3 Fontes

Como demonstra a figura 4.8, as fontes tipográficas utilizadas no jogo estão armazenadas na pasta *fontes*. Para que fossem adicionadas no jogo o arquivo de fontes teve que ser adicionado dentro de *assets*, a pasta *fontes* é utilizada apenas para organizar os arquivos. Somente as fontes do tipo *OpenType* e *TrueType* são suportadas pelo Unity.

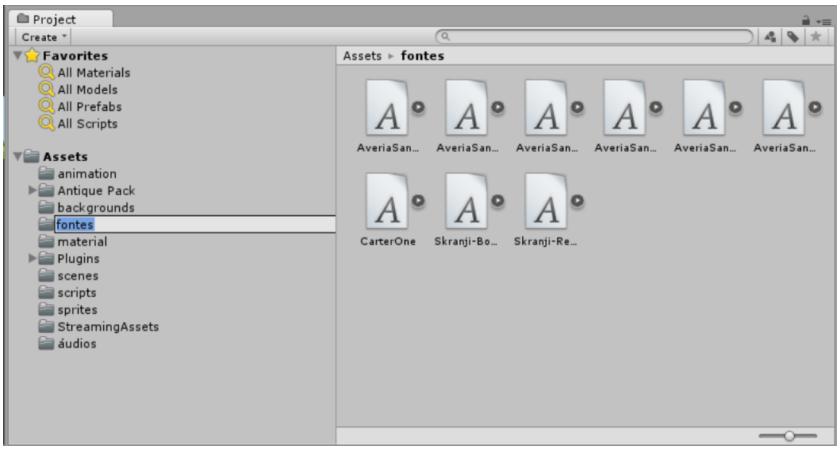


Figura 4.8 Pasta Fontes.

4.1.3.4 Materials

Os *materials* são os elementos gráficos que estão compostos nas cenas de qualquer projeto desenvolvido no Unity. Baseiam-se nos *shaders* e texturas, o *shader* é o formato da imagem no *material* e a textura é o desenho em sua superfície. Os elementos de fundo das cenas do *"Laça Palavras"* foram desenvolvidos através dos *materials*.

A pasta *material* armazena todos os *materials* do jogo, como demonstra a figura 4.9.

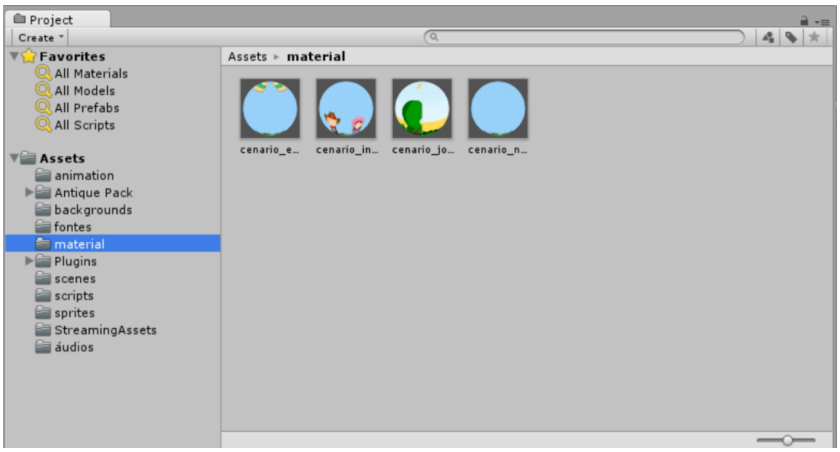


Figura 4.9 Pasta Materials.

4.1.3.5 Plugins

A maioria das funcionalidades implementadas em um projeto do Unity são desenvolvidas através dos *scripts*, mas além das suas funcionalidades próprias o Unity também disponibiliza a opção do desenvolvedor adicionar bibliotecas externas ao projeto com o objetivo de acrescentar mais funções e recursos externos, os chamados *plugins*. O banco de dados SQLite abordado na seção 3.3 é um exemplo de *plugin* adicionado ao *Laça Palavras*. O SQLite não requer muita memória em tempo de execução e sua utilização foi necessária para que o jogo ficasse mais rápido e leve nos dispositivos móveis dos usuários, salvando os dados de maneira eficiente e eficaz.

A figura 4.10 demonstra a pasta *plugins* criada somente para armazenar *plugins* utilizados no jogo.

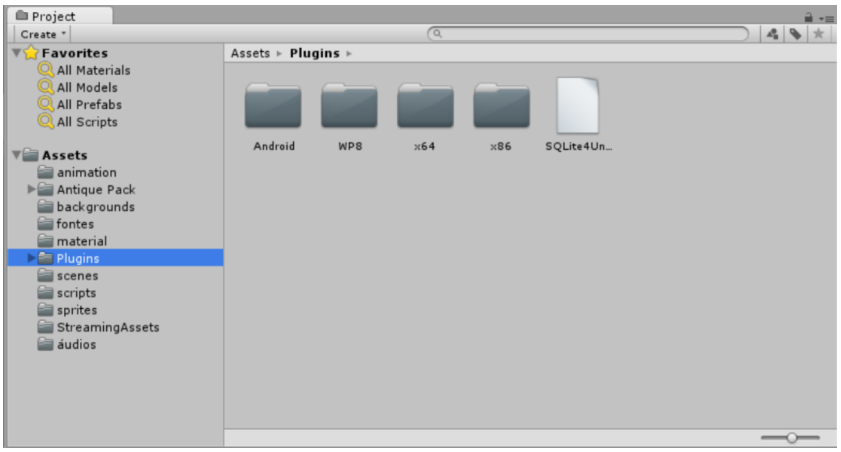


Figura 4.10 Pasta Plugins.

4.1.3.6 Scenes

Como ilustra a figura 4.11, a pasta *scenes* é utilizada para armazenar todas as cenas desenvolvidas para o jogo. O *Laça Palavras* possui cinco cenas para possibilitar o desenvolvimento do jogo em “pedaços” e cada uma delas possuem seus próprios ambientes, decorações e funcionalidades.

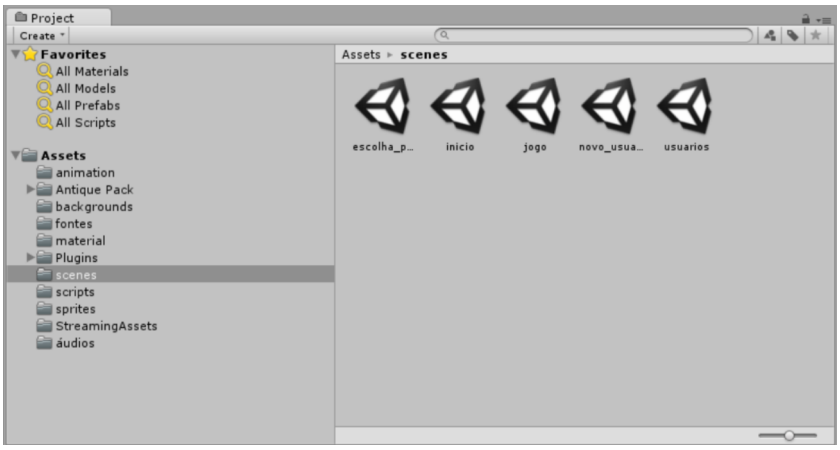


Figura 4.11 Pasta Scenes.

Para possibilitar a transição de cenas foi necessário implementar a função *Application.LoadLevel()* nos *scripts* das cenas do jogo, essa função é utilizada para determinar qual a próxima cena a ser carregada.

A figura 4.12 demonstra a implementação da função no *script* da tela do jogo, quando o usuário apertar o botão *menu* e escolher a opção *menu inicial*, o jogo irá salvar todos os dados do usuário e através da função *Application.LoadLevel()* a cena é alterada para a do *Menu Inicial*.

```
// Ir para o menu inicial!
public void btnClick_MenuInicial()
{
    Time.timeScale = 1f;

    dadosJogo.Instance.currentUser.reforco_atual1 = dadosJogo.Instance.vetor_reforco_atual[0];
    dadosJogo.Instance.currentUser.reforco_atual2 = dadosJogo.Instance.vetor_reforco_atual[1];
    dadosJogo.Instance.currentUser.reforco_atual3 = dadosJogo.Instance.vetor_reforco_atual[2];
    dadosJogo.Instance.currentUser.reforco_atual4 = dadosJogo.Instance.vetor_reforco_atual[3];

    dadosJogo.Instance.salvar_dados();
    Application.LoadLevel("inicio");
}
```

Figura 4.12 Utilização da função *Application.LoadLevel()*.

4.1.3.7 Scripts

Como aborda a subseção 3.1.1, um *script* é um conjunto de instruções desenvolvidas através de alguma linguagem de programação com o objetivo de possibilitar a realização de determinadas ações em um projeto.

Para desenvolver os *scripts* no *Laça Palavras* foi utilizada a linguagem de programação C# disponibilizada gratuitamente pelo Unity, a sua escolha foi pelo fato de ser uma linguagem bastante poderosa e de fácil aprendizagem. Os *scripts* são vinculados nas cenas do jogo para que torne possível aplicar a programação das funcionalidades e ações desenvolvidas para seus componentes. A figura 4.13 ilustra a pasta onde são armazenados todos os *scripts* desenvolvidos.

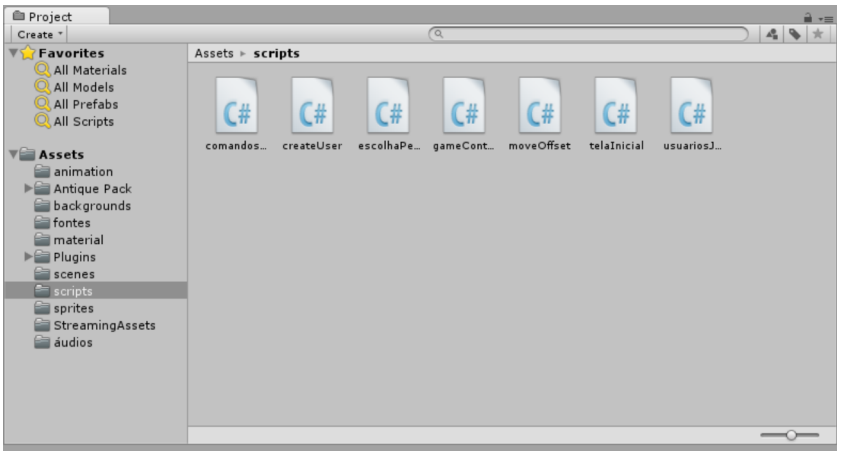


Figura 4.13 Pasta Scripts.

4.1.3.8 Sprites

Para deixar o projeto mais organizado os *sprites* foram armazenados em uma pasta separada das imagens normais, como demonstra a figura 4.14.

Figura 4.14 Pasta Sprites.

4.1.3.9 StreamingAssets

A pasta *StreamingAssets* é utilizada para armazenar o arquivo de banco de dados SQLite implementado, como demonstra a figura 4.15. Todos os arquivos inseridos nessa pasta são copiados para alguma pasta específica no dispositivo móvel do usuário.

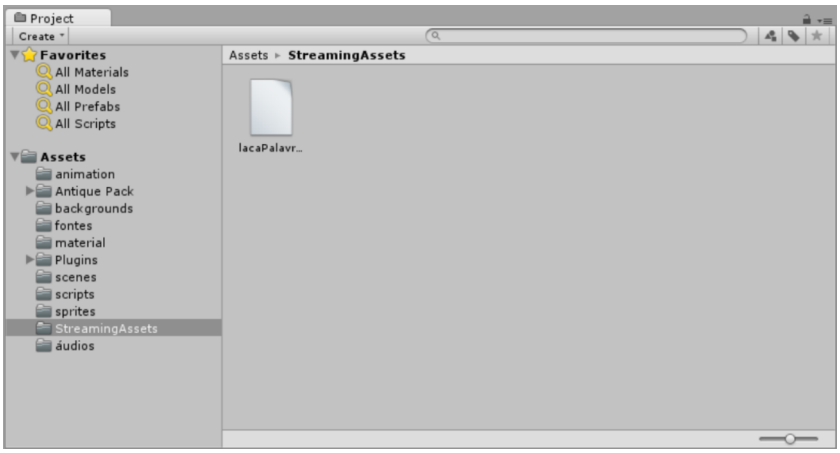


Figura 4.15 Pasta StreamingAssets.

4.1.3.10 Áudios

Todos os áudios utilizados no jogo estão armazenados na pasta *Áudios*, como ilustra a figura 4.16. Um áudio está ligado a um *GameObject* para reproduzir os sons. O Unity

disponibiliza diversas manipulações simples de sons para serem utilizadas e para manipulações mais específicas é necessário a implementação de funções nos *scripts* do jogo. O áudio do cavalo, por exemplo, na aba *Audio Sorce* do Unity foi utilizada uma opção que transforma o som do galope do cavalo em um *loop* infinito, mas para pausar o som quando o usuário desejasse foi necessário desenvolver uma função no *script* da tela do jogo determinando seu bloqueio quando solicitado.

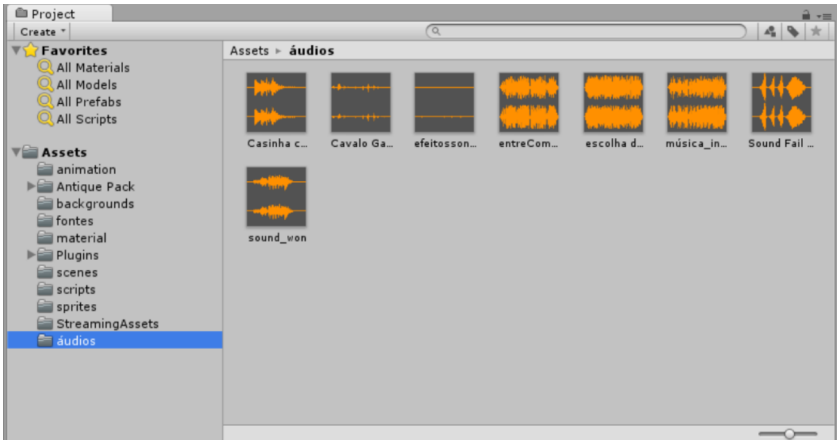


Figura 4.16 Pasta Audios.

A figura 4.17 demonstra uma função implementada no *script* da tela inicial do jogo que manipula os áudios utilizados na cena. O usuário ao apertar o botão com formato de alto-falante disponibilizado na tela inicial todos os áudios são bloqueados através da condição do *if com-paudio.isPlaying*, quando o botão de retomar for acionado todos os áudios são desbloqueados dentro da condição do *else*.

```
    public void click_Sound()
    {
        // Função que bloqueia e ativa o som!
        var compaudio = somOn.GetComponent<AudioSource>();

        if (compaudio.isPlaying)
        {
            somOn.SetActive(false);
            somOn.GetComponent<AudioSource>().Pause();
            bird.GetComponent<AudioSource>().Pause();
            somOff.SetActive(true);
        }
        else
        {
            somOn.SetActive(true);
            somOn.GetComponent<AudioSource>().UnPause();
            bird.GetComponent<AudioSource>().UnPause();
            somOff.SetActive(false);
        }

    }
}
```

Figura 4.17 Função desenvolvida para manipulação de áudios.

4.1.4 Cenas do Jogo Digital Ortográfico

Esta subseção tem como objetivo abordar o funcionamento e o *layout* das cenas desenvolvidas para o *Laça Palavras*.

4.1.4.1 Cena: Menu Inicial

Inicialmente, o usuário é apresentado ao *Menu Inicial* do jogo com as opções de criar novo jogo, continuar um jogo existente ou sair. Para possibilitar a transição de telas, quando o usuário escolher alguma das opções contidas no *menu*, foi utilizada a função *Application.LoadLevel()* no *script* da cena *Menu Inicial*. As animações existentes na cena são controladas pela função *FixedUpdate()*.

O botão com formato de alto-falante, quando apertado, exerce a função de bloquear todos os áudios da cena e em seu lugar aparecerá o botão de alto-falante mudo, que é utilizado para desbloquear os áudios. Todas as cenas explicadas nas subseções posteriores também

possuem o botão bloquear e desbloquear áudios. A figura 4.18 ilustra a cena inicial do *Laça Palavras*.

Figura 4.18 Cena: Menu Inicial.

4.1.4.2 Cena: Criar Usuário

Para criar um novo usuário é necessário primeiramente que a opção *Novo Jogo* da cena *Menu Inicial* seja escolhida. Como ilustra a figura 4.19, o jogador iniciante deve digitar um nome de identificação no campo em branco para criar seu usuário. Não é permitido escolher nomes de usuários já existentes.

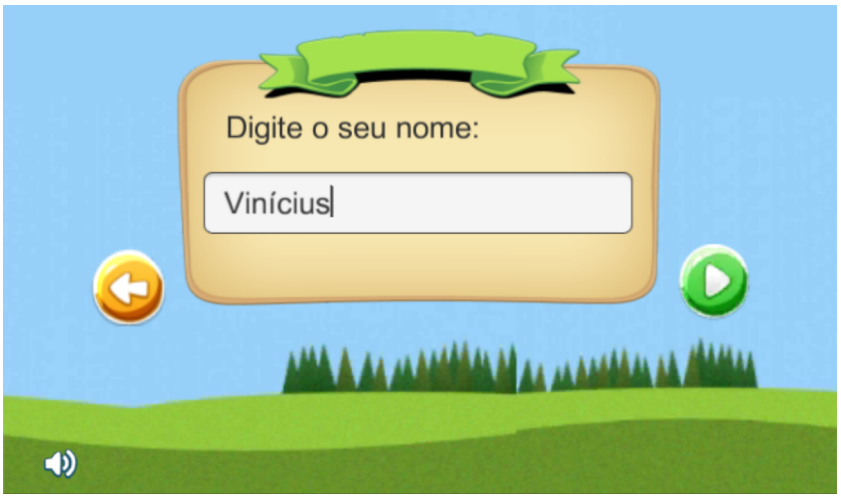


Figura 4.19 Cena: Criar Usuário.

A figura 4.20 demonstra uma parte do *script* da cena *Criar Usuário*. As atribuições dentro do *foreach()* objetiva "setar" os dados do usuário para serem utilizados no jogo e posteriormente a função *Application.LoadLevel()* é utilizada para mudar a cena atual para a próxima cena.

```
    foreach (var user in users)
    {
        dadosJogo.Instance.currentUser = new user
        {
            Id = user.Id,
            Name = user.Name,
            Score = user.Score,
            Nivel = user.Nivel
        };
    }
    Application.LoadLevel("escolha_personagem");
}
```

Figura 4.20 Trecho do código referente a parte de obtenção dos dados do usuário. A última linha permite a mudança para cena onde serão selecionados os personagens do jogo.

4.1.4.3 Cena: Escolher Personagem

Após ter efetuado seu cadastro, o usuário tem que escolher um personagem para representá-lo no jogo. A cena *Escolher Personagem* disponibiliza um menina caracterizada de *cowgirl* e um menino caracterizado de *cowboy* como opções. Estes irão exercer a mesma função de laçar as caixas selecionadas pelo usuário ao decorrer do jogo.

A figura 4.21 ilustra o *layout* da cena *Escolher Personagem*.

Figura 4.21 Cena: Escolher Personagem.

4.1.4.4 Cena: Escolher Usuário

Ao pensar na possibilidade de um usuário ter que parar de jogar em um determinado momento e interessar em voltar a jogar posteriormente de onde parou, foi desenvolvida a cena *Escolher Usuário* objetivando possibilitar o acesso as informações do usuário, que são salvas todas as vezes que o mesmo sair do jogo, responder 10 palavras ou restante de palavras do nível ou voltar para o *Menu Inicial*. Esta cena é ilustrada na figura 4.22.

O botão do lado esquerdo representado por uma seta tem a função de voltar para o *Menu Inicial* e o botão do lado direito vai avançar para a escolha de personagem, o usuário tem a possibilidade de escolher outro personagem mesmo utilizando um cadastro já existente. Para exclusão de algum usuário existente, basta selecioná-lo e apertar o botão *Remover usuário* representado por um “X”.

Figura 4.22 Cena: Escolher Usuário.

A parte da implementação do *script* ilustrada na figura 4.23, é responsável por buscar todos os usuários cadastrados no banco de dados e retornar para o *select* da cena *Escolher Usuário*.

```
IEnumerable<user> users = data._connection.Table<user>();
foreach (var user in users)
{
    Dropdown.OptionData a = new Dropdown.OptionData();
    a.text = user.Name;

    Users.options.Insert(index, a);
    ++index;
}
Debug.Log("Palavras User: " + words.Count() + " opções:" + words2.Count() + " Users: " +index);
```

Figura 4.23 Buscar usuários existentes.

4.1.4.5 Cena: Jogo

A cena *Jogo* exibe um *layout* que transmite ao usuário a sensação de estar no velho oeste, possuindo todas suas imagens e animações com características direcionadas ao público infantil. O cenário principal possui as animações das nuvens, pássaros, cavalos, caixas, personagens, e também possui as imagens do sol, arco-íris, árvore, frutas, dentre outros. Tornando um cenário bastante atrativo e interativo, como pode ser observado através da figura 4.24.

A função da animação do cavalo no jogo é carregar uma caixa e soltar no meio da cena,

são duas caixas e cada uma possui uma letra como opção de resposta para completar a palavra apresentada na parte superior ao centro. Para o usuário escolher uma opção basta dar um toque em cima da caixa. Nesse momento, o personagem que fica no meio da cena vai laçar a caixa, o nome do jogo "Laça Palavras" originou-se dessa ação.

Caso o usuário queira acessar o *menu* do jogo, basta clicar no botão superior do lado direito, nesse momento o jogo pausa e um quadro com três botões é apresentado na tela, o botão "Retornar" tem a função de retornar ao jogo, o "Menu Inicial" salva todas as informações do jogo e a transição para a cena inicial é feita através da função *Application.LoadLevel()*, e por último, o botão "Sair do Jogo" salva todas as informações do jogo e fecha o *Laça Palavras*.

Figura 4.24 Personagem *cowboy*.

No início, são apresentadas somente palavras do primeiro nível para o usuário, porém, ao decorrer do jogo, o algoritmo de balanceamento dinâmico de dificuldade é que vai determinar o avanço ou retrocesso de nível. Se o usuário mantiver um número alto de acertos o jogo avança dinamicamente e, da mesma forma, se manter um número alto de erros o jogo retrocede dinamicamente.

Quando o usuário escolhe uma opção de resposta, automaticamente aparece uma frase na tela informando se a escolha foi correta ou não e a palavra é completada corretamente na parte superior ao centro, como ilustra a figura 4.25.

Figura 4.25 Personagem *cowgirl*.

A classe *gameController* implementada no *script* "gameController.cs", é responsável por manipular todas as funcionalidades existentes na cena. A classe contém o método *Start()*, que é chamado no início da execução do *script* inicializando todos os dados da tela e do usuário corrente, todos os *scripts* devem possuir uma classe *Start()*. Para melhor apresentação do método foi desenvolvido um pseudocódigo, ilustrado na figura 4.26. O trecho do código referente ao método *Start()* se encontra disponível no Apêndice A.

```
1: {Método Start()}
2: Receber do banco de dados a pontuação do usuário;
3:      se usuário está iniciando o jogo então
4:          Mensagem de ajuda na tela;
5:      fim se
6:      senão
7:          Executar a função de calcular porcentagem da casa;
8:      fim senão
9:      se cowgirl for selecionada então
10:         Ativar cowgirl na cena;
11:     fim se
12:     senão
13:         Ativar cowboy na cena;
14:     fim senão
15:     {...}
16:     Inicializar posições de todas as animações;
17:     {...}
18:     Carregar número de acertos do personagem;
19:     Carregar do banco de dados as informações do usuário;
20:     Selecionar palavra aleatória correspondente ao nível do usuário;
```

Figura 4.26 Pseudocódigo do método *Start()*.

O método *FixedUpdate()* exerce a função principal na cena *Jogo*. Esse método é responsável por manipular todo o andamento do jogo, controlando as animações, analisando as opções escolhidas, chamando funções implementadas em outros *scripts*, dentre outros. A figura 4.27 demonstra um pseudocódigo desenvolvido para facilitar o entendimento do método *FixedUpdate()* da cena atual. O trecho do código referente ao método *FixedUpdate()* se encontra disponível no Apêndice B.

```
1: {Método FixedUpdate()}
2: enquanto usuário continuar jogando faça
3:                    {...}
4:                    Manipulação das animações;
5:                    {...}
6:          se usuário escolher uma opção então
7:                    Manipulação do personagem para laçar a opção;
8:                    se acertar então
9:                              {...}
10:                             Apresentar uma mensagem de acerto na cena;
11:                             Calcular porcentagem de acerto para oferecer recompensa a cada 10%;
12:                             {...}
13:                   fim se
14:                   senão
15:                             {...}
16:                             Apresentar mensagem de erro na cena;
17:                             {...}
18:                   fim senão
19:                   se palavras respondidas então
20:                             Executar a função do algoritmo de aprendizagem por reforço SARSA;
21:                             Executar a função selecionar níveis;
22:                             Executar a função salvar dados do usuário corrente;
23:                             Atribuir zero a variável palavras respondidas;
24:                   fim se
25:         fim se
26: fim enquanto
```

Figura 4.27 Pseudocódigo do método *FixedUpdate()*.

4.2 Implementação do SARSA no Jogo Digital Ortográfico

O algoritmo de inteligência artificial desenvolvido para o *Laça Palavras* foi baseado nos conceitos da técnica de aprendizagem por reforço SARSA, sua utilização objetiva realizar o balanceamento dinâmico de dificuldade no jogo. A seção 3.5 aborda com detalhes sobre o Ajuste Dinâmico de Dificuldade. O SARSA é um algoritmo iterativo e essa iteração é especificada

em cada problema, o problema a ser resolvido no jogo não possui uma grande complexidade, facilitando na definição e implementação da iteração, que é mostrada a seguir em um breve resumo:

1. Apresentar 10 palavras de um determinado nível ao usuário, se o nível possuir menos de 10 palavras a serem respondidas naquele momento, então apresentar restante de palavras;
2. Calcular o desempenho de acerto e erros;
3. Calcular o reforço;
4. Atualizar o reforço na probabilidade de cada nível;
5. Selecionar um nível com maior probabilidade;

O *Laça Palavras* possui 40 palavras diferentes armazenadas no banco de dados que são distribuídas em 4 níveis existentes no jogo. O jogo sempre inicia apresentando 10 palavras do primeiro nível ao usuário. Logo após apresentar essas palavras, o desempenho vai ser calculado de acordo com a quantidade de acertos que o usuário obteve. Como o jogo não repete palavras completadas corretamente, se em determinado momento o usuário estiver em um nível no qual já completou algumas palavras de forma correta anteriormente, o jogo irá apresentar somente as palavras restantes do nível, calculando o desempenho de acordo com a quantidade de palavras restantes que foram apresentadas. Exemplo, o usuário já completou 3 palavras corretamente no nível 2, quando o usuário for jogar no nível 2 novamente, somente 7 palavras vão ser apresentadas e o cálculo de desempenho vai ser calculado de acordo com a quantidade de acertos dentre essas 7 palavras.

O método *FixedUpdate()* implementado na classe *GameController* da cena *Jogo* é responsável por apresentar as palavras. Quando forem apresentadas, a função do algoritmo SARSA é chamada, como demonstra a figura 4.28.

```
inteligencia.Instance.SARSA();
```

Figura 4.28 Chamada a função do algoritmo SARSA.

A função que calcula o desempenho vai ser chamada no início da função do algoritmo SARSA. Sua utilização é de suma importância para o problema em questão, pois é a partir

desse cálculo que será determinado quais são os níveis que receberão reforços positivos e quais são os níveis que receberão reforços negativos. O cálculo é feito a partir das 10 palavras ou das palavras restantes respondidas, se *(Acertos* $\geq$ *60%)*, o algoritmo deve reforçar positivamente os níveis posteriores, o nível atual e os anteriores devem receber reforços negativos, se *(Acertos* $<$ *60%)* todos os níveis anteriores recebem reforços positivos, o nível atual e os níveis posteriores recebem reforços negativos.

O objetivo é fazer com que o algoritmo avance níveis se o usuário estiver acertando muitas palavras e retroceda níveis se o mesmo estiver errando muitas palavras. A figura 4.29 demonstra a função que calcula o desempenho do usuário, a variável resultado vai receber a média entre a quantidade de palavras acertadas pela quantidade total de palavras exibidas.

```
public double DesempenhoUsuario()
{
    double resultado = (double)(bancoPalavras.Instance.acertos) / (bancoPalavras.Instance.qtd_Words);
    return resultado;
}
```

Figura 4.29 Cálculo do desempenho do usuário.

O próximo passo é calcular o reforço. A equação do SARSA $\alpha[r(s_t,a_t) + \gamma\, Q(s_{t+1},a_{t+1})$ - $Q(s_t,a_t)]$ é utilizada como base para o cálculo do reforço. Essa equação foi adaptada para o problema em questão da seguinte maneira:

- α recebe por padrão o valor 1;
- O reforço $r(s_t,a_t)$ recebe o valor do desempenho;
- A taxa de desconto γ recebe por padrão o valor 0.9, com intuito de valorizar as recompensas ao decorrer do tempo;
- O $Q(s_{t+1},a_{t+1})$ é a probabilidade atual do nível subsequente superior ou inferior, dependendo do resultado do desempenho. Se *(Desempenho* $\geq$ *0.6)*, $Q(s_{t+1},a_{t+1})$ recebe a probabilidade atual do nível subsequente superior. Se *(Desempenho* $<$ *0.6)*, $Q(s_{t+1},a_{t+1})$ recebe a probabilidade atual do nível subsequente inferior;
- $Q(s_t,a_t)$ é a probabilidade atual do nível corrente;

O reforço é calculado de duas maneiras diferentes de acordo com o valor do desempenho. Suponha que no primeiro caso o usuário tenha um *(Desempenho* $\geq$ *0.6)* e esteja no nível 3, sendo que as probabilidades dos níveis 1, 2, 3 e 4 sejam 0.5, 0.7, 0.8 e 0.5, respectivamente. O reforço vai ser calculado da seguinte maneira:

- reforço $\leftarrow$ 1x[(Desempenho) + 0.9x0.5 - 0.8]

Note que, como o usuário está no nível 3, então $Q(s_t, a_t)$ recebe o valor 0.8, que é sua probabilidade atual. Como *(Desempenho* $\geq$ *0.6)*, $Q(s_{t+1}, a_{t+1})$ recebe a probabilidade atual do nível 4, que é o nível subsequente superior.

Como no primeiro caso o *(Desempenho* $\geq$ *0.6)*, as probabilidades dos níveis inferiores e do nível atual recebem reforço negativo no momento da atualização, diminuindo a probabilidade de serem selecionados. As probabilidades dos níveis superiores recebem reforço positivo no momento da atualização, aumentando a probabilidade de serem selecionados. A figura 4.30 demonstra o pseudocódigo da implementação do algoritmo no momento em que *(Desempenho* $\geq$ *0.6)*. O trecho do código referente à implementação do cálculo de reforço e atualização de probabilidades quando *(Desempenho* $\geq$ *0.6)* se encontra disponível no Apêndice C.

```
1: {SARSA ()}
2: se (desempenho ≥ 0.6) faça
3:      se Nível 1 faça
4:          Calcular α[r(s_t, a_t) + γQ(s_t+1, a_t+1) - Q(s_t, a_t)];
5:          Probabilidade atual do nível 1 = Probabilidade atual do nível 1 - Reforço;
6:          Probabilidade atual do nível 2 = Probabilidade atual do nível 2 + Reforço;
7:          Probabilidade atual do nível 3 = Probabilidade atual do nível 3 + Reforço;
8:          Probabilidade atual do nível 4 = Probabilidade atual do nível 4 + Reforço;
9:      fim se
10:     se Nível 2 faça
11:         Calcular α[r(s_t, a_t) + γQ(s_t+1, a_t+1) - Q(s_t, a_t)];
12:         Probabilidade atual do nível 1 = Probabilidade atual do nível 1 - Reforço;
13:         Probabilidade atual do nível 2 = Probabilidade atual do nível 2 - Reforço;
14:         Probabilidade atual do nível 3 = Probabilidade atual do nível 3 + Reforço;
15:         Probabilidade atual do nível 4 = Probabilidade atual do nível 4 + Reforço;
16:     fim se
17:     se Nível 3 faça
18:         Calcular α[r(s_t, a_t) + γQ(s_t+1, a_t+1) - Q(s_t, a_t)];
19:         Probabilidade atual do nível 1 = Probabilidade atual do nível 1 - Reforço;
20:         Probabilidade atual do nível 2 = Probabilidade atual do nível 2 - Reforço;
21:         Probabilidade atual do nível 3 = Probabilidade atual do nível 3 - Reforço;
22:         Probabilidade atual do nível 4 = Probabilidade atual do nível 4 + Reforço;
23:     fim se
24:     se Nível 4 faça
25:         Calcular α[r(s_t, a_t) + γQ(s_t+1, a_t+1) - Q(s_t, a_t)];
26:         Probabilidade atual do nível 1 = Probabilidade atual do nível 1 - Reforço;
27:         Probabilidade atual do nível 2 = Probabilidade atual do nível 2 - Reforço;
28:         Probabilidade atual do nível 3 = Probabilidade atual do nível 3 - Reforço;
29:         Probabilidade atual do nível 4 = Probabilidade atual do nível 4 + Reforço;
30:     fim se
31: fim se
```

Figura 4.30 Pseudocódigo da implementação do cálculo de reforço e atualização de probabilidades quando *(Desempenho* $\geq$ *0.6)*.

Suponha agora que no segundo caso o usuário tenha um *(Desempenho < 0.6)* e esteja no nível 2, sendo que as probabilidades dos níveis 1, 2, 3 e 4 sejam 0.2, 0.7, 0.6 e 0.4, respecti-

vamente. O reforço vai ser calculado da seguinte maneira:

- reforço ← 1x[(Desempenho) + 0.9x0.2 - 0.7]

Observe que, como o usuário está no nível 2, então $Q(s_t, a_t)$ recebe o valor 0.7, que é sua probabilidade atual. Como *(Desempenho < 0.6)*, então $Q(s_{t+1}, a_{t+1})$ recebe a probabilidade atual do nível 1, que é o nível subsequente inferior.

Como no segundo caso o *(Desempenho < 0.6)*, as probabilidades dos níveis superiores e do nível atual recebem reforço negativo no momento de atualização, para diminuir a probabilidade de serem selecionados. As probabilidades dos níveis inferiores recebem reforço positivo no momento da atualização, para aumentar a probabilidade de serem selecionados. A figura 4.31 demonstra o pseudocódigo da implementação do algoritmo no momento em que *(Desempenho < 0.6)*. O trecho do código referente a implementação do cálculo de reforço e atualização de probabilidades quando *(Desempenho < 0.6)* se encontra disponível no Apêndice D.

```
1: {SARSA ()}
2: se (desempenho < 0.6) faça
3:      se Nível 1 faça
4:           Calcular α[r(s_t, a_t) + γQ(s_t+1, a_t+1) - Q(s_t, a_t)];
5:           Probabilidade atual do nível 1 = Probabilidade atual do nível 1 + Reforço;
6:           Probabilidade atual do nível 2 = Probabilidade atual do nível 2 - Reforço;
7:           Probabilidade atual do nível 3 = Probabilidade atual do nível 3 - Reforço;
8:           Probabilidade atual do nível 4 = Probabilidade atual do nível 4 - Reforço;
9:      fim se
10:     se Nível 2 faça
11:          Calcular α[r(s_t, a_t) + γQ(s_t+1, a_t+1) - Q(s_t, a_t)];
12:          Probabilidade atual do nível 1 = Probabilidade atual do nível 1 + Reforço;
13:          Probabilidade atual do nível 2 = Probabilidade atual do nível 2 - Reforço;
14:          Probabilidade atual do nível 3 = Probabilidade atual do nível 3 - Reforço;
15:          Probabilidade atual do nível 4 = Probabilidade atual do nível 4 - Reforço;
16:     fim se
17:     se Nível 3 faça
18:          Calcular α[r(s_t, a_t) + γQ(s_t+1, a_t+1) - Q(s_t, a_t)];
19:          Probabilidade atual do nível 1 = Probabilidade atual do nível 1 + Reforço;
20:          Probabilidade atual do nível 2 = Probabilidade atual do nível 2 + Reforço;
21:          Probabilidade atual do nível 3 = Probabilidade atual do nível 3 - Reforço;
22:          Probabilidade atual do nível 4 = Probabilidade atual do nível 4 - Reforço;
23:     fim se
24:     se Nível 4 faça
25:          Calcular α[r(s_t, a_t) + γQ(s_t+1, a_t+1) - Q(s_t, a_t)];
26:          Probabilidade atual do nível 1 = Probabilidade atual do nível 1 + Reforço;
27:          Probabilidade atual do nível 2 = Probabilidade atual do nível 2 + Reforço;
28:          Probabilidade atual do nível 3 = Probabilidade atual do nível 3 + Reforço;
29:          Probabilidade atual do nível 4 = Probabilidade atual do nível 4 - Reforço;
30:     fim se
31: fim se
```

Figura 4.31 Pseudocódigo da implementação do cálculo de reforço e atualização de probabilidades quando *(Desempenho < 0.6)*.

A probabilidade que possuir valor menor que 0 vai receber automaticamente o valor 0 e a probabilidade que possuir valor maior que 1 vai receber automaticamente o valor 1. O objetivo é manter os valores de probabilidade entre o intervalo de 0 a 1.

Por último, o nível é selecionado de acordo com sua probabilidade. A figura 4.32 demonstra o pseudocódigo da função que seleciona nível, a primeira condição definida na linha 3 é utilizada para determinar a transição de níveis se todas as palavras de um determinado nível forem respondidas, a segunda condição definida na linha 10 é utilizada para selecionar o nível que possuir a maior probabilidade, se existir duas probabilidades iguais com o valor maior, o nível de menor dificuldade que conter essa probabilidade vai ser selecionado. O trecho do código referente ao método de selecionar nível se encontra disponível no Apêndice E.

```
1: {seleciona_nivel()}
2: se (Acertos = Quantidade de Palavras no nível) faça
3:         se (Nível Corrente= Nível 4) faça
4:                 Nível = 1;
5:         fim se
6:         senão
7:                 Pular para nível posterior;
8:         fim senão
9: fim se
10: senão
11:        Selecionar nível com maior probabilidade;
12: fim senão
```

Figura 4.32 Função para selecionar nível.

O capítulo 5 demonstra a aplicação dos testes e comparações em sequências de desempenhos de um usuário específico no jogo, com intuito de apresentar o comportamento do algoritmo de inteligência artificial SARSA desenvolvido para o *Laça Palavras*.

CAPÍTULO 5

Testes

O presente capítulo objetiva apresentar os testes aplicados no algoritmo de inteligência artificial desenvolvido para o *Laça Palavras*, que é utilizado para balancear dinamicamente a sua dificuldade. A aplicação dos testes utilizou uma sequência de desempenhos em cada iteração, pré-definida pelo próprio autor, ilustrando as probabilidades em que cada nível recebeu em determinada iteração e também a transição desses níveis de acordo com as probabilidades. Os resultados foram explicados e comparados com os resultados dos mesmos testes aplicados no algoritmo de balanceamento dinâmico de dificuldade baseado no Q-Learning, desenvolvido por (Leal, 2016) para ser utilizado no *Laça Palavras*.

A aplicação de testes em crianças, visando estimar a eficiência e eficácia do algoritmo de balanceamento dinâmico de dificuldade na aprendizagem do conteúdo proposto no jogo, não foi possível por motivo de tempo para submeter à apreciação do Comitê de Ética, sendo de fato, objetivo que pode ser concluído em trabalhos futuros.

5.1 Primeiro Teste

O primeiro teste possui a seguinte sequência de desempenhos a cada iteração: 0.6, 0.2, 1.0, 1.0, 0.7, 0.3, 1.0. Pode-se observar que o desempenho oscila ao decorrer do jogo, sendo que, em alguns momentos o desempenho está alto e em outros momentos o desempenho está baixo.

O gráfico 5.1 demonstra a transição de níveis em cada iteração utilizando o algoritmo baseado no SARSA. Na primeira iteração o desempenho é aceitável, ou seja, *(Desempenho* $\geq$ *0.6)*, observa-se que o jogo aumenta dinamicamente a sua dificuldade, passando a apresentar as palavras do nível 2. Na segunda iteração o desempenho cai drasticamente para 0.2, fazendo com que o jogo diminua dinamicamente a sua dificuldade, voltando a apresentar palavras do nível 1. Na terceira iteração o desempenho volta a subir, fazendo com que o jogo apresente palavras do nível 2 novamente. Nota-se que o algoritmo se comporta de maneira dinâmica

até o fim do jogo, ajustando a dificuldade de forma correta para a sequência de desempenhos propostos no primeiro teste.

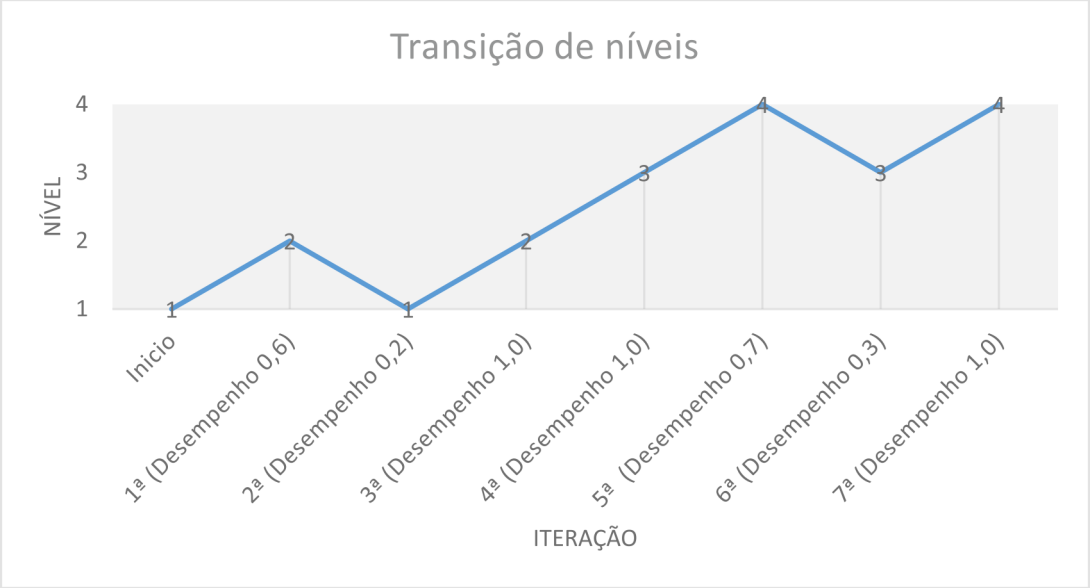
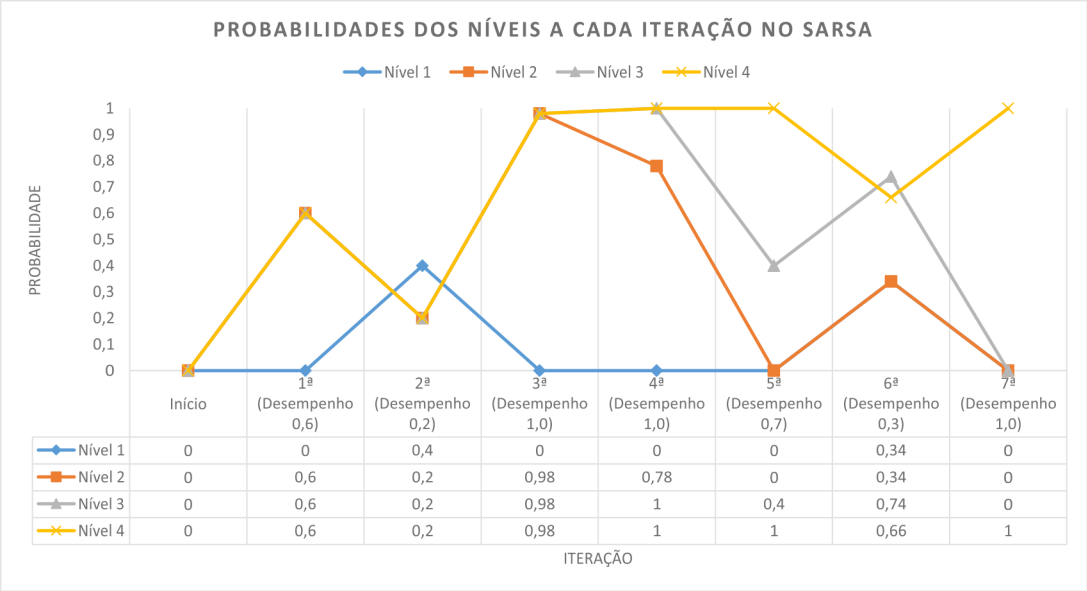


Figura 5.1 Gráfico de linha com as transações de níveis ao decorrer das iterações do SARSA. Fonte: Elaborada pelo autor.

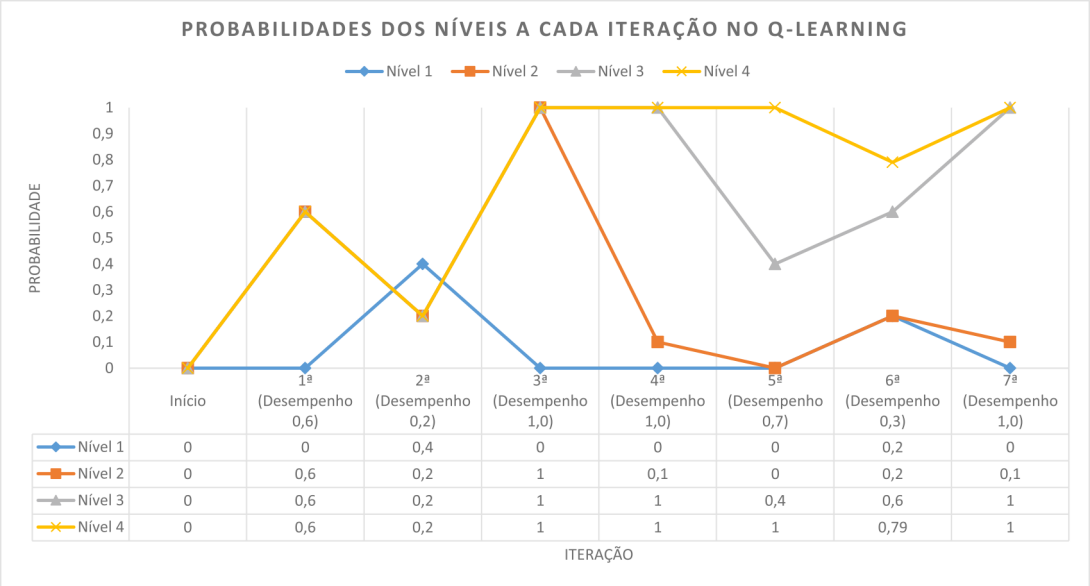
O gráfico 5.2 apresenta a probabilidade de cada nível a cada iteração do algoritmo baseado no SARSA. Observa-se que na primeira iteração, como o *(Desempenho* $\geq$ *0.6)*, os níveis superiores recebem reforços positivos, fazendo com que o nível 2 seja selecionado. Na segunda iteração o *(Desempenho* $<$ *0.6)*, então o nível atual e os níveis superiores recebem reforços negativos e o nível inferior recebe reforço positivo, o nível 1 passa a possuir a maior probabilidade, fazendo com que o mesmo seja selecionados. Nota-se que as probabilidades dos níveis oscilam bastante em função do desempenho a cada iteração. Na sexta iteração, por exemplo, o jogo diminui dinamicamente sua dificuldade, pelo fato de que o valor do desempenho foi muito baixo, fazendo com que os níveis inferiores ao nível atual recebessem reforços positivos.



	Início	1ª (Desempenho 0,6)	2ª (Desempenho 0,2)	3ª (Desempenho 1,0)	4ª (Desempenho 1,0)	5ª (Desempenho 0,7)	6ª (Desempenho 0,3)	7ª (Desempenho 1,0)
Nível 1	0	0	0,4	0	0	0	0,34	0
Nível 2	0	0,6	0,2	0,98	0,78	0	0,34	0
Nível 3	0	0,6	0,2	0,98	1	0,4	0,74	0
Nível 4	0	0,6	0,2	0,98	1	1	0,66	1

Figura 5.2 Gráfico de linha com os resultados das probabilidades de cada nível ao decorrer das iterações do SARSA. Fonte: Elaborada pelo autor.

O gráfico 5.3 apresenta o mesmo teste aplicado no algoritmo baseado no *Q-Learning*. Nota-se que o algoritmo possui um comportamento similar ao comportamento do SARSA. Tanto o Q-Learning quanto o SARSA adaptou-se de maneira esperada à dificuldade proposta no primeiro teste, diminuindo e avançando níveis quando necessário.



	Início	1ª (Desempenho 0,6)	2ª (Desempenho 0,2)	3ª (Desempenho 1,0)	4ª (Desempenho 1,0)	5ª (Desempenho 0,7)	6ª (Desempenho 0,3)	7ª (Desempenho 1,0)
Nível 1	0	0	0,4	0	0	0	0,2	0
Nível 2	0	0,6	0,2	1	0,1	0	0,2	0,1
Nível 3	0	0,6	0,2	1	1	0,4	0,6	1
Nível 4	0	0,6	0,2	1	1	1	0,79	1

Figura 5.3 Gráfico de linha com os resultados das probabilidades de cada nível ao decorrer das iterações do Q-Learning. Fonte: Elaborada pelo autor.

5.2 Segundo Teste

Para o segundo teste foi definida uma sequência de desempenhos a cada iteração diferente dos valores propostos no primeiro teste, são eles: 0.5, 1.0, 1.0, 0.8, 0.1, 1.0.

O gráfico 5.4 demonstra a transição de níveis em cada iteração utilizando o algoritmo baseado no SARSA. Como o desempenho na primeira iteração é mediano, o algoritmo continua no mesmo nível. Na segunda, terceira e quarta iteração o desempenho é satisfatório, fazendo com que o jogo aumente o nível dinamicamente. A quinta iteração o desempenho volta a cair, fazendo com que o jogo diminua o nível dinamicamente. Nota-se que no segundo teste o algoritmo se comporta de maneira diferente ao primeiro teste, mostrando sua adaptatividade a cada problema específico proposto.

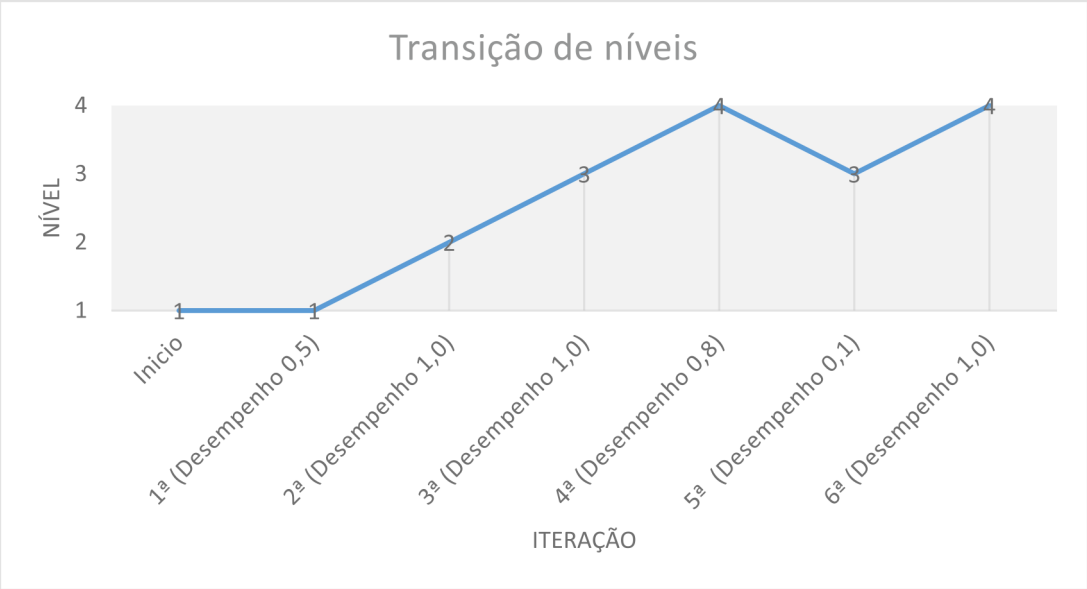
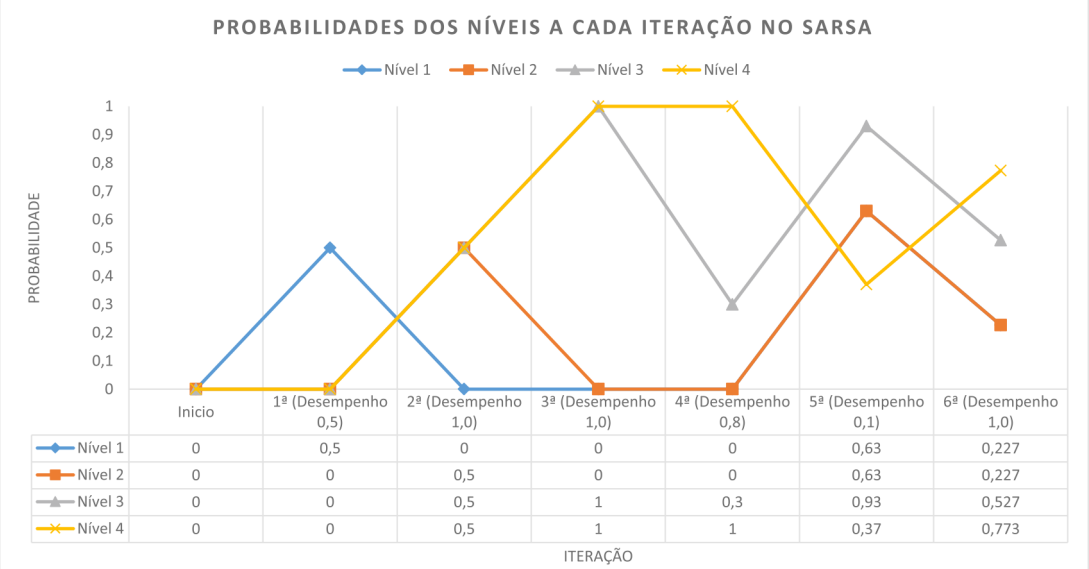


Figura 5.4 Gráfico de linha da transição de níveis do algoritmo SARSA. Fonte: Elaborada pelo autor.

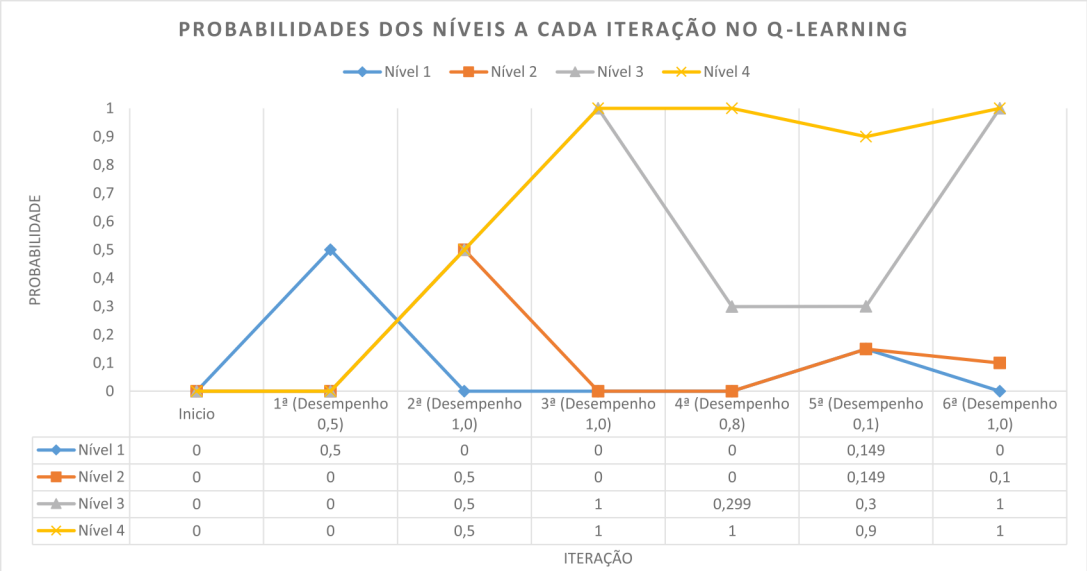
O gráfico 5.5 apresenta a probabilidade de cada nível a cada iteração do algoritmo baseado no SARSA. Pode-se observar através do gráfico que o algoritmo se comporta de acordo com o que foi proposto na seção 4.2, sendo que, sempre que o desempenho for insatisfatório os níveis inferiores recebem reforços positivos e sempre que o desempenho for satisfatório os níveis superiores recebem reforços positivos. As probabilidades do segundo teste tiveram resultados diferentes aos resultados do primeiro teste, pelo fato de que os desempenhos são diferentes, nos dois testes o algoritmo se comporta de maneira correta.



	Inicio	1ª (Desempenho 0,5)	2ª (Desempenho 1,0)	3ª (Desempenho 1,0)	4ª (Desempenho 0,8)	5ª (Desempenho 0,1)	6ª (Desempenho 1,0)
Nível 1	0	0,5	0	0	0	0,63	0,227
Nível 2	0	0	0,5	0	0	0,63	0,227
Nível 3	0	0	0,5	1	0,3	0,93	0,527
Nível 4	0	0	0,5	1	1	0,37	0,773

Figura 5.5 Gráfico de linha com os resultados das probabilidades de cada nível ao decorrer das iterações do SARSA. Fonte: Elaborada pelo autor.

O gráfico 5.6 demonstra o mesmo teste aplicado no algoritmo baseado no Q-Learning. O algoritmo também possui um comportamento similar ao SARSA no segundo teste, demonstrando aumentar e diminuir dinamicamente os níveis quando necessário. Na quinta iteração somente é que o Q-Learning toma uma decisão diferente ao SARSA, mesmo com o desempenho baixo o algoritmo continua no mesmo nível. O que não muda o comportamento esperado de ambos os algoritmos.



	Inicio	1ª (Desempenho 0,5)	2ª (Desempenho 1,0)	3ª (Desempenho 1,0)	4ª (Desempenho 0,8)	5ª (Desempenho 0,1)	6ª (Desempenho 1,0)
Nível 1	0	0,5	0	0	0	0,149	0
Nível 2	0	0	0,5	0	0	0,149	0,1
Nível 3	0	0	0,5	1	0,299	0,3	1
Nível 4	0	0	0,5	1	1	0,9	1

Figura 5.6 Gráfico de linha com os resultados das probabilidades de cada nível ao decorrer das iterações do Q-Learning. Fonte: Elaborada pelo autor.

Nota-se através dos testes que o desenvolvimento do algoritmo para balancear dinamicamente a dificuldade do *Laça Palavras* se comporta de maneira esperada, pode ser observado através dos dois testes diferentes que o algoritmo se adaptou de maneira correta nas dificuldades propostas, sendo assim, sua utilização vai ser importante em qualquer sequência de desempenhos que um usuário qualquer obtiver.

Capítulo 6

Conclusões

Neste trabalho, foram apresentadas a importância das atividades lúdicas no processo de aprendizagem do ser humano e a dificuldade em que crianças possuem na aprendizagem correta da ortografia, principalmente quando se trata de palavras que possuam letras ou dígrafos concorrentes, que representem o mesmo som, no mesmo contexto, e que não existam regularidades de escrita para essas palavras. Dentro desse contexto foi possível constatar a importância do desenvolvimento de um jogo digital ortográfico para atender a este problema.

Com o desenvolvimento de um jogo digital foi observada a necessidade de um algoritmo para balancear dinamicamente a dificuldade desse jogo. A técnica de aprendizagem por reforço SARSA foi escolhida para ser utilizada como base para a implementação do algoritmo no *Laça Palavras*. A escolha do SARSA surgiu como uma ótima ideia para resolver o problema em questão, os testes apresentados no capítulo 5 comprovam o comportamento esperado do algoritmo desenvolvido, atingindo o objetivo que era de desenvolver um algoritmo que apresentasse os próprios desafios para cada usuário.

As ferramentas descritas no capítulo 3 foram de enorme importância para que fosse possível desenvolver o *Laça Palavras* e o algoritmo de inteligência artificial, principalmente a ferramenta Unity 3D, que se destacou pela sua facilidade de utilização e por oferecer uma versão poderosa e gratuita.

Considerando o que foi proposto, esse trabalho cumpriu com todos os objetivos apresentados, o que resultou em um jogo digital ortográfico divertido, interessante e baseado em práticas pedagógicas adequadas. O algoritmo de inteligência artificial desenvolvido fez com que o jogo ficasse mais interessante, sendo indispensável para o *Laça Palavras*.

CAPÍTULO 7

Trabalhos Futuros

Para que o presente trabalho não se torne apenas o final de um projeto, mas também o início de uma jornada, os possíveis trabalhos futuros são apresentados a seguir:

- Submeter o projeto à apreciação do Comitê de Ética em Pesquisa em Seres Humanos para testar a eficiência e eficácia do jogo digital ortográfico no processo de aprendizagem de crianças;
- Desenvolver um módulo que atenda aos problemas ortográficos de jovens e adultos, podendo também testar sua eficiência e eficácia no processo de aprendizagem desses indivíduos.
- Desenvolver uma nova versão do jogo digital ortográfico com um número maior de conteúdos e cenários de jogabilidade, não se limitando somente em um cenário de velho oeste, podendo ter outras recompensas e formas de apresentar as palavras;
- Implementar outras técnicas de inteligência artificial para possibilitar o estudo comparativo entre os algoritmos. Exemplo: Q(λ), Dyna, etc.;

APÊNDICE A

Método Start() da tela do jogo

```
void Start () {

    ScoreInitial = dadosJogo.Instance.currentUser.Score;

    if (ScoreInitial == 0) {

        mensagem_inicial.SetActive(true);
        mensagem_audio_frase.SetActive(true);

    }
    else verifica_porcentagem_casa();

    horse.GetComponent<AudioSource>().Pause();
    horse2.GetComponent<AudioSource>().Pause();
    Score.text = "ACERTOS: " + dadosJogo.Instance.currentUser.Score;

    // IF para saber qual personagem vai ativar!
    if (dadosJogo.Instance.currentPesonagem == 1)
    {
        cowboy_moving_rope.SetActive(false);
```

```
}
else cowgirl_moving_rope.SetActive(false);

// inicializando as posições dos objetos!
x1 = cloud1.transform.position.x;
y1 = cloud1.transform.position.y;

x2 = cloud2.transform.position.x;
y2 = cloud2.transform.position.y;

x3 = cloud3.transform.position.x;
y3 = cloud3.transform.position.y;

xBird1 = bird1.transform.position.x;
yBird1 = bird1.transform.position.y;

xBird2 = bird2.transform.position.x;
yBird2 = bird2.transform.position.y;

xHorseinitial = xHorse = horse.transform.position.x;
yHorseinitial = yHorse = horse.transform.position.y;

xHorse2initial = xHorse2 = horse2.transform.position.x;
yHorse2initial = yHorse2 = horse2.transform.position.y;

xboard_letter_initial = xboard_letter =
 board_letter.transform.position.x;
yboard_letter_initial = yboard_letter =
```

```
    board_letter.transform.position.y;

    xboard2_letter_initial = xboard2_letter =
    board2_letter.transform.position.x;
    yboard2_letter_initial = yboard2_letter =
    board2_letter.transform.position.y;


    bancoPalavras.Instance.carrega_palavras_nivel(dadosJogo.
    Instance.currentUser.Nivel);

    countWords = (bancoPalavras.Instance.qtd_Words);

    idPalavra = Random.Range(1, 9);

    Debug.Log(countWords + " e palavra " + bancoPalavras.Instance.
    palavras[dadosJogo.Instance.currentUser.Nivel][idPalavra].palavra_completa);

    randNum_blocos = Random.Range(1.0f, 2.0f);

    this.setPalavra(dadosJogo.Instance.currentUser.Nivel, idPalavra);
}
```

APÊNDICE B

Método FixedUpdate() da tela do jogo

```
void FixedUpdate () {

 // limite do movimento dos objetos!
 if (xboard_letter >= -2f) xboard_letter -= 0.065f;
 else if (yboard_letter <= -1.7f)
 {
     yboard_letter += 0.090f;
     yHorse -= 0.06f;

 }

 if (xboard2_letter >= 0f) xboard2_letter -= 0.065f;
 else if (yboard2_letter <= -1.7f)
 {
     yboard2_letter += 0.090f;
     yHorse2 -= 0.06f;

 }
 if (x1 >= 9.52f) x1 = -9.52f;
 if (x2 >= 9.52f) x2 = -9.52f;
 if (x3 >= 9.52f) x3 = -9.52f;
```

```
if (xBird1 <= -9.52f) xBird1 = 16.52f;
if (xBird2 <= -9.52f) xBird2 = 23.52f;
if (xHorse <= -9.52f)
{
    horse.GetComponent<AudioSource>().Pause();
    xHorse = 30f;

}

if (xHorse < 9.52f && respondido == false) horse.
GetComponent<AudioSource>().UnPause();

if (xHorse2 <= -9.52f)
{
    horse2.GetComponent<AudioSource>().Pause();
    xHorse2 = 30f;
}

if (xHorse2 < 9.52f && respondido == false) horse2.
GetComponent<AudioSource>().UnPause();

// Velocidade dos movimentos dos objetos!
x1 = x1 + 0.002f;
x2 = x2 + 0.002f;
x3 = x3 + 0.002f;
xBird1 -= 0.009f;
xBird2 -= 0.009f;
xHorse -= 0.065f;
```

```
xHorse2 -= 0.065f;



// IF para verificar se usuário escolheu a opção!
if (respondido)
{

        xHorse = xHorseinitial;
        yHorse = yHorseinitial;
        xHorse2 = xHorse2initial;
        yHorse2 = yHorse2initial;

        if (aux <= 9.52f && xboard_letter < 9f)
        {
            aux += 0.09f;
            //yboard_letter += 0.09f;
            //yboard2_letter += 0.09f;
            //board_letter.transform.position =
            new Vector2(xboard_letter, yboard_letter);
            //board2_letter.transform.position =
            new Vector2(xboard2_letter, yboard2_letter);

        }
        else
        {

            xboard_letter = xboard_letter_initial;
            yboard_letter = yboard_letter_initial;
```

```
    aux = 0;
    xboard2_letter = xboard2_letter_initial;
    yboard2_letter = yboard2_letter_initial;
    respondido = false;
    message_hit.SetActive(false);
    message_error.SetActive(false);

if (dadosJogo.Instance.currentPesonagem == 2)
{
    cowboy_lacando1.SetActive(false);
    cowboy_lacando2.SetActive(false);
    cowboy_moving_rope.SetActive(true);

}
else
{
    cowgirl_lacando1.SetActive(false);
    cowgirl_lacando2.SetActive(false);
    cowgirl_moving_rope.SetActive(true);

}

    idPalavra = countWords;
    randNum_blocos = Random.Range(1.0f, 2.0f);



if (countWords == -1 )
```

```
{
    if (dadosJogo.Instance.currentUser.Score == bancoPalavras.
    Instance.total_palavras)
    {

        mensagem_fim_jogo.SetActive(true);
        horse.SetActive(false);
        horse2.SetActive(false);
        board_letter.SetActive(false);
        board2_letter.SetActive(false);
        cowboy_moving_rope.SetActive(false);
        cowgirl_moving_rope.SetActive(false);
        fimJogo = true;
        return;

    }



    inteligencia.Instance.SARSA();
    inteligencia.Instance.seleciona_nivel();
    dadosJogo.Instance.salvar_dados();
    ScoreInitial = dadosJogo.Instance.currentUser.Score;
    bancoPalavras.Instance.carrega_palavras_nivel(dadosJogo.
    Instance.currentUser.Nivel);
    countWords = (bancoPalavras.Instance.qtd_Words - 1);
    idPalavra = countWords;

}
```

```
            this.setPalavra(dadosJogo.Instance.currentUser.Nivel, countWords);
        }

    }

    cloud1.transform.position = new Vector2(x1, y1);
    cloud2.transform.position = new Vector2(x2, y2);
    cloud3.transform.position = new Vector2(x3, y3);
    bird1.transform.position = new Vector2(xBird1, yBird1);
    bird2.transform.position = new Vector2(xBird2, yBird2);
    horse.transform.position = new Vector2(xHorse,yHorse);
    horse2.transform.position = new Vector2(xHorse2, yHorse2);
    board_letter.transform.position = new Vector2(xboard_letter, yboard_letter);
    board2_letter.transform.position = new Vector2(xboard2_letter, yboard2_letter);

}
```

APÊNDICE C

Código do SARSA quando *(Desempenho $\geq$ 0.6)*

```
if (dadosJogo.Instance.ultimo_desempenho >= 0.6)
 {
     if (dadosJogo.Instance.currentUser.Nivel == 0)
     {
         reforco = dadosJogo.Instance.ultimo_desempenho +
         (desconto * dadosJogo.Instance.vetor_reforco_atual[1])
          - dadosJogo.Instance.vetor_reforco_atual[0];
         if (reforco < 0)
         {
             reforco *= (-1);
         }

         dadosJogo.Instance.vetor_reforco_atual[0] = ((dadosJogo.Instance.vetor_reforco_atual[0] -
          reforco) >= 0.0d) ? (dadosJogo.Instance.vetor_reforco_atual[0]-reforco) : 0;

         dadosJogo.Instance.vetor_reforco_atual[1] = ((dadosJogo.Instance.vetor_reforco_atual[1] +
          reforco) <= 1.0d) ? (dadosJogo.Instance.vetor_reforco_atual[1]+reforco) : 1;

         dadosJogo.Instance.vetor_reforco_atual[2] = ((dadosJogo.Instance.vetor_reforco_atual[2] +
          reforco) <= 1.0d) ? (dadosJogo.Instance.vetor_reforco_atual[2]+reforco) : 1;
```

```
   dadosJogo.Instance.vetor_reforco_atual[3] = ((dadosJogo.Instance.vetor_reforco_atual[3] +
     reforco) <= 1.0d) ? (dadosJogo.Instance.vetor_reforco_atual[3]+reforco) : 1;



}

if (dadosJogo.Instance.currentUser.Nivel == 1)
{
    reforco = dadosJogo.Instance.ultimo_desempenho + (desconto *
     dadosJogo.Instance.vetor_reforco_atual[2]) -
     dadosJogo.Instance.vetor_reforco_atual[1];
    if (reforco < 0)
    {
        reforco *= (-1);
    }
    dadosJogo.Instance.vetor_reforco_atual[0] = ((dadosJogo.Instance.vetor_reforco_atual[0] -
     reforco) >= 0.0d) ? (dadosJogo.Instance.vetor_reforco_atual[0]-reforco) : 0;

    dadosJogo.Instance.vetor_reforco_atual[1] = ((dadosJogo.Instance.vetor_reforco_atual[1] -
    reforco) >= 0.0d) ? (dadosJogo.Instance.vetor_reforco_atual[1]-reforco) : 0;

    dadosJogo.Instance.vetor_reforco_atual[2] = ((dadosJogo.Instance.vetor_reforco_atual[2] +
    reforco) <= 1.0d) ? (dadosJogo.Instance.vetor_reforco_atual[2]+reforco) : 1;

    dadosJogo.Instance.vetor_reforco_atual[3] = ((dadosJogo.Instance.vetor_reforco_atual[3] +
    reforco) <= 1.0d) ? (dadosJogo.Instance.vetor_reforco_atual[3]+reforco) : 1;

}
```

```
if (dadosJogo.Instance.currentUser.Nivel == 2)
{
    reforco = dadosJogo.Instance.ultimo_desempenho + (desconto *
    dadosJogo.Instance.vetor_reforco_atual[3]) -
    dadosJogo.Instance.vetor_reforco_atual[2];
    if (reforco < 0)
    {
        reforco *= (-1);
    }
    dadosJogo.Instance.vetor_reforco_atual[0] = ((dadosJogo.Instance.vetor_reforco_atual[0] -
    reforco) >= 0.0d) ? (dadosJogo.Instance.vetor_reforco_atual[0]-reforco) : 0;

    dadosJogo.Instance.vetor_reforco_atual[1] = ((dadosJogo.Instance.vetor_reforco_atual[1] -
     reforco) >= 0.0d) ? (dadosJogo.Instance.vetor_reforco_atual[1]-reforco) : 0;

    dadosJogo.Instance.vetor_reforco_atual[2] = ((dadosJogo.Instance.vetor_reforco_atual[2] -
    reforco) >= 0.0d) ? (dadosJogo.Instance.vetor_reforco_atual[2]-reforco) : 0;

    dadosJogo.Instance.vetor_reforco_atual[3] = ((dadosJogo.Instance.vetor_reforco_atual[3] +
    reforco) <= 1.0d) ? (dadosJogo.Instance.vetor_reforco_atual[3]+reforco) : 1;


}

if (dadosJogo.Instance.currentUser.Nivel == 3)
{
    reforco = dadosJogo.Instance.ultimo_desempenho + (desconto *
```

```
         dadosJogo.Instance.vetor_reforco_atual[3]) -
          dadosJogo.Instance.vetor_reforco_atual[3];
        if (reforco < 0)
        {
            reforco *= (-1);
        }
        dadosJogo.Instance.vetor_reforco_atual[0] = ((dadosJogo.Instance.vetor_reforco_atual[0] -
        reforco) >= 0.0d) ? (dadosJogo.Instance.vetor_reforco_atual[0] - reforco) : 0;

        dadosJogo.Instance.vetor_reforco_atual[1] = ((dadosJogo.Instance.vetor_reforco_atual[1] -
        reforco) >= 0.0d) ? (dadosJogo.Instance.vetor_reforco_atual[1] - reforco) : 0;

        dadosJogo.Instance.vetor_reforco_atual[2] = ((dadosJogo.Instance.vetor_reforco_atual[2] -
        reforco) >= 0.0d) ? (dadosJogo.Instance.vetor_reforco_atual[2] - reforco) : 0;

        dadosJogo.Instance.vetor_reforco_atual[3] = ((dadosJogo.Instance.vetor_reforco_atual[3] +
        reforco) <= 1.0d) ? (dadosJogo.Instance.vetor_reforco_atual[3] + reforco) : 1;


    }

 }
}
```

APÊNDICE D

Código do SARSA quando *(Desempenho < 0.6)*

```
if ( dadosJogo.Instance.ultimo_desempenho < 0.6)
 {



     if (dadosJogo.Instance.currentUser.Nivel == 0)
     {
         reforco = dadosJogo.Instance.ultimo_desempenho +
         (desconto * dadosJogo.Instance.vetor_reforco_atual[0])
          - dadosJogo.Instance.vetor_reforco_atual[0];
         if  (reforco < 0)
         {
             reforco *=(-1);
         }
         dadosJogo.Instance.vetor_reforco_atual[0]= ((dadosJogo.Instance.vetor_reforco_atual[0]
         + reforco) <= 1.0d) ? (dadosJogo.Instance.vetor_reforco_atual[0]+reforco) : 1;


         dadosJogo.Instance.vetor_reforco_atual[1]= ((dadosJogo.Instance.vetor_reforco_atual[1]


          - reforco) >= 0.0d) ? (dadosJogo.Instance.vetor_reforco_atual[1]-reforco) : 0;


         dadosJogo.Instance.vetor_reforco_atual[2]= ((dadosJogo.Instance.vetor_reforco_atual[2]
```

```
    - reforco) >= 0.0d) ? (dadosJogo.Instance.vetor_reforco_atual[2]-reforco) : 0;

    dadosJogo.Instance.vetor_reforco_atual[3]= ((dadosJogo.Instance.vetor_reforco_atual[3]
     - reforco) >= 0.0d) ? (dadosJogo.Instance.vetor_reforco_atual[3]-reforco) : 0;


}

if (dadosJogo.Instance.currentUser.Nivel == 1)
{
    reforco = dadosJogo.Instance.ultimo_desempenho +
     (desconto * dadosJogo.Instance.vetor_reforco_atual[0])
     - dadosJogo.Instance.vetor_reforco_atual[1];
    if (reforco < 0)
    {
        reforco *= (-1);
    }
    dadosJogo.Instance.vetor_reforco_atual[0] = ((dadosJogo.Instance.vetor_reforco_atual[0]
     + reforco) <=1.0d) ? (dadosJogo.Instance.vetor_reforco_atual[0] + reforco) : 1;

    dadosJogo.Instance.vetor_reforco_atual[1] = ((dadosJogo.Instance.vetor_reforco_atual[1]
    - reforco) >= 0.0d) ? (dadosJogo.Instance.vetor_reforco_atual[1] - reforco) : 0;

    dadosJogo.Instance.vetor_reforco_atual[2] = ((dadosJogo.Instance.vetor_reforco_atual[2]
     - reforco) >=0.0d) ? (dadosJogo.Instance.vetor_reforco_atual[2] - reforco) : 0;

    dadosJogo.Instance.vetor_reforco_atual[3] = ((dadosJogo.Instance.vetor_reforco_atual[3]
     - reforco) >=0.0d) ? (dadosJogo.Instance.vetor_reforco_atual[3] - reforco) : 0;
```

```
}
```

```
if (dadosJogo.Instance.currentUser.Nivel == 2)
{
    reforco = dadosJogo.Instance.ultimo_desempenho +
     (desconto * dadosJogo.Instance.vetor_reforco_atual[1])
     - dadosJogo.Instance.vetor_reforco_atual[2];
    if (reforco < 0)
    {
        reforco *= (-1);
    }
    dadosJogo.Instance.vetor_reforco_atual[0]= ((dadosJogo.Instance.vetor_reforco_atual[0]
    + reforco) <=1.0d) ? (dadosJogo.Instance.vetor_reforco_atual[0]+reforco) : 1;

    dadosJogo.Instance.vetor_reforco_atual[1]= ((dadosJogo.Instance.vetor_reforco_atual[1] +
    reforco) <= 1.0d) ? (dadosJogo.Instance.vetor_reforco_atual[1]+reforco) : 1;

    dadosJogo.Instance.vetor_reforco_atual[2]= ((dadosJogo.Instance.vetor_reforco_atual[2] -
     reforco) >=0.0d) ? (dadosJogo.Instance.vetor_reforco_atual[2]-reforco) : 0;

    dadosJogo.Instance.vetor_reforco_atual[3]= ((dadosJogo.Instance.vetor_reforco_atual[3] -
     reforco) >= 0.0d) ? (dadosJogo.Instance.vetor_reforco_atual[3]-reforco) : 0;
```

```
}

if (dadosJogo.Instance.currentUser.Nivel == 3)
{
    reforco = dadosJogo.Instance.ultimo_desempenho + (desconto *
    dadosJogo.Instance.vetor_reforco_atual[2])- dadosJogo.Instance.vetor_reforco_atual[3];
    if (reforco < 0)
    {
       reforco *= (-1);
    }
    dadosJogo.Instance.vetor_reforco_atual[0]= ((dadosJogo.Instance.vetor_reforco_atual[0] +
    reforco) <=1.0d) ? (dadosJogo.Instance.vetor_reforco_atual[0]+reforco) : 1;

    dadosJogo.Instance.vetor_reforco_atual[1]= ((dadosJogo.Instance.vetor_reforco_atual[1] +
    reforco)<= 1.0d) ? (dadosJogo.Instance.vetor_reforco_atual[1]+reforco) : 1;

    dadosJogo.Instance.vetor_reforco_atual[2]=
    ((dadosJogo.Instance.vetor_reforco_atual[2] + reforco) <= 1.0d) ?
    (dadosJogo.Instance.vetor_reforco_atual[2]+reforco) : 1;

    dadosJogo.Instance.vetor_reforco_atual[3]= ((dadosJogo.Instance.vetor_reforco_atual[3] -
    reforco) >=0.0d) ?(dadosJogo.Instance.vetor_reforco_atual[3]-reforco) : 0;


       }

   }
```

APÊNDICE E

Método seleciona_nivel()

```
public void seleciona_nivel()
{



 if (bancoPalavras.Instance.acertos == bancoPalavras.Instance.qtd_Words)
 {
     if (dadosJogo.Instance.currentUser.Nivel == 3) dadosJogo.Instance.currentUser.Nivel = 0;
     else
         ++dadosJogo.Instance.currentUser.Nivel;
 }
 else
 {
     double maior = 0;
     for (int i = 0; i < 4; ++i)
     {
         if (dadosJogo.Instance.vetor_reforco_atual[i] > maior)
         {
             maior = dadosJogo.Instance.vetor_reforco_atual[i];
             dadosJogo.Instance.currentUser.Nivel = i;
         }
     }
```

```
  }
}
```

APÊNDICE F

Palavras e Frases do Laça Palavras

Nível 1:

- Palavra: Engraçado. Frase: “O palhaço é muito engraçado!”;
- Palavra: Dança. Frase: “Eu adoro a aula de dança!”;
- Palavra: Assunto. Frase: “Esse assunto é chato!”;
- Palavra: Tosse. Frase: “Ele tosse muito?”;
- Palavra: Assustar. Frase: “Assustar as pessoas é chato!”;
- Palavra: Pássaro. Frase: “O pássaro voa bem alto.”;
- Palavra: Professora. Frase: “A professora entregou a tarefa aos alunos.”;
- Palavra: Girassol. Frase: “Gosto de girassol.”;
- Palavra: Osso. Frase: “O osso do meu cachorro é grande!”;
- Palavra: Assado. Frase: “O frango assado da minha mãe é delicioso!”;

Nível 2:

- Palavra: Papel. Frase: “O papel molhou?”;
- Palavra: Desagradável. Frase: “Isso é desagradável!”;
- Palavra: Legal. Frase: “Isso é legal, adoro o carnaval.”;
- Palavra: Caracol. Frase: “O caracol é nojento! Eca!”;
- Palavra: Anel. Frase: “Nossa! Que anel lindo!”;

- Palavra: Álcool. Frase: “Que cheiro do álcool ruim!”;
- Palavra: Gol. Frase: “E é gol do Atlético Mineiro!”;
- Palavra: Sol. Frase: “A Terra gira em torno do Sol.”;
- Palavra: Canal. Frase: “Mudei de canal para ver o filme.”;
- Palavra: Passos. Frase: “Os meus passos são grandes.”;

Nível 3:

- Palavra: Girafa. Frase: “Eu vi uma girafa”;
- Palavra: Hoje. Frase: “Hoje está super gelado. Brrrrrr!”;
- Palavra: Majestade. Frase: “Vossa majestade, tu mandas em tudo? Podes ordenar um pôr do Sol?”;
- Palavra: Mágica. Frase: “A festa de ontem foi mágica.”;
- Palavra: Zoológico. Frase: “O zoológico é maneiro!”;
- Palavra: Página. Frase: “Essa página está rasgada.”;
- Palavra: Gente. Frase: “Ei, gente! Vocês não vão dançar? Isso é um baile.”;
- Palavra: Jiló. Frase: “Eu comi Jiló”;
- Palavra: Poço. Frase: “Vamos procurar um poço?”;
- Palavra: Sol. Frase: “A Terra gira em torno do Sol.”;

Nível 4:

- Palavra: Treze. Frase: “Eu tenho treze anos.”;
- Palavra: Blusa. Frase: “Essa blusa é sua?”;
- Palavra: Doze. Frase: “Você vai fazer doze anos amanhã?”;
- Palavra: Duzentos. Frase: “Eu quero duzentos balões para a festa.”;

- Palavra: Asa. Frase: “A asa do passarinho quebrou!”;
- Palavra: Rosa. Frase: “Vamos colocar a rosa no lugar dela.”;
- Palavra: Lousa. Frase: “A lousa é branca?”;
- Palavra: Assoprar. Frase: “Eu vou assoprar a sua casinha de palha!”;
- Palavra: Álcool. Frase: “Que cheiro do álcool ruim!”;
- Palavra: Página. Frase: “Essa página está rasgada.”;

Referências Bibliográficas

Alves(2010) Daniela Pereira Alves. Modelagem de aprendizagem por reforço e controle em nível meta para melhorar a performance da comunicação em gerência de tráfego aéreo. Citado na pág. 31

Andrade *et al.*(2006) Gustavo Danzi de Andrade, Lisboa Ramalho, *et al.* Balanceamento dinâmico de jogos: uma abordagem baseada em aprendizagem por reforço. URL http://repositorio.ufpe.br/bitstream/handle/123456789/2604/arquivo5292_1.pdf?sequence=1&isAllowed=y. Acessado em dezembro de 2015. Citado na pág. xii, 26

Barto e Sutton(2012) G. Barto e A. Sutton. *Reinforcement Learning: An Introduction*. Bradford Book. Citado na pág. 28, 29, 31, 32

Bittencourt e Giraffa(2003) João Ricardo Bittencourt e Lucia Maria Giraffa. Modelando ambientes de aprendizagem virtuais utilizando role-playing games. *XIV Simpósio Brasileiro de Informática na Educação. Rio de Janeiro: SBC*, 2003:718–727. URL http://www.nce.ufrj.br/sbie2003/publicacoes/paper71.pdf. Acessado em agosto de 2015. Citado na pág. 20

Caillois(1990) Roger Caillois. Os jogos e os homens, trad. *José Garcez Palha. Lisboa: Edições Cotovia*. Citado na pág. 5

Canuto e Moita(2011) Érika CA Canuto e Filomena Mª GSC Moita. Os jogos digitais e a aprendizagem: interrelações entre o ensino e os estilos dos alunos. *X SBGames*, páginas 1–10. URL http://www.sbgames.org/sbgames2011/proceedings/sbgames/papers/cult/full/92224_1.pdf. Acessado em maio de 2015. Citado na pág. 6

Coelho e Prado(2015) Taiany Coelho e Giuliano Prado. Analise comparativa para avaliacao de tecnologias de banco de dados para dispositivos moveis. *Caderno de Estudos em Sistemas de Informação*, 1(1). URL http://seer.cesjf.br/index.php/cesi/article/view/128/48. Acessado em novembro de 2015. Citado na pág. 23

Comachio(2012) Vanderson Comachio. Funcionamento de banco de dados em android: um estudo experimental utilizando sqlite. URL http://repositorio.roca.utfpr.edu.br:8080/

jspui/bitstream/1/537/1/MD_COADS_2011_2_07.pdf. Acessado em dezembro de 2015. Citado na pág. 24

Falkembach(2006) Gilse A Morgental Falkembach. O lúdico e os jogos educacionais. *Mídias na Educação*, 16. URL http://penta3.ufrgs.br/midiasedu/modulo13/etapa1/leituras/arquivos/Leitura_1.pdf. Acessado em fevereiro de 2016. Citado na pág. 10

Fedoce e Squirra(2011) Rosângela Spagnol Fedoce e Sebastião Carlos Squirra. A tecnologia móvel e os potenciais da comunicação na educação. *LOGOS, Comunicação e Universidade*, páginas 267–278. URL http://ibict.metodista.br/tedeSimplificado/tde_arquivos/5/TDE-2011-02-18T134242Z-912/Publico/Rosangela/%20Fedoce.pdf. Acessado em maio de 2015. Citado na pág. 7

Fleury *et al.***(2006)** André Leme Fleury, AZ DAHMER, e G SCHWARTZ. Da inclusão à emancipação digital: novos modelos de produção e distribuição de conteúdo digital, 2006. URL http://www.abepro.org.br/biblioteca/ENEGEP2006_TR560372_8245.pdf. Acessado em maio de 2015. Citado na pág.

Galvão *et al.***(2009)** Aline de Oliveira Galvão, Osni Barbosa Chagas, Simone Bello Kaminski Aires, e João Paulo Aires. Alternativas para o desenvolvimento de software, sem custo, para micro e pequenas empresas. URL http://www.admpg.com.br/revista2009/v2/artigos/a14.pdf. Acessado em março de 2016. Citado na pág. 24

GLOBALWEBINDEX(2014) GLOBALWEBINDEX. 15 trends for 2015: Android excels, 2014. URL http://www.globalwebindex.net/blog/15-for-15-android-excels. Acessado em junho de 2015. Citado na pág. 8

Henke *et al.***(2011)** Márcia Henke, Clayton Santos, Eduardo Nunan, Eduardo Feitosa, Eulanda dos Santos, e Eduardo Souto. Aprendizagem de máquina para segurança em redes de computadores: Métodos e aplicações. *Minicursos do XI Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg 2011)*, 1:53–103. URL http://www.peotta.com/sbseg2011/resources/downloads/minicursos/91555.pdf. Acessado em dezembro de 2015. Citado na pág. 28

Huizinga(1971) Johan Huizinga. *Homo ludens: o jogo como elemento da cultura*. Editora da Universidade de S. Paulo, Editora Perspectiva. Citado na pág. 2

IBGE(2013) IBGE. Acesso a internet e posse de telefone móvel celular para uso pessoal, 2013. URL http://www.ibge.gov.br/home/presidencia/noticias/imprensa/ppts/00000012962305122013234016242127.pdf. Acessado em maio de 2015. Citado na pág. 7

Kaelbling *et al.*(**1996**) Leslie Pack Kaelbling, Michael L Littman, e Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, páginas 237–285. URL http://www.jair.org/media/301/live-301-1562-jair.pdf. Acessado em janeiro de 2015. Citado na pág. 29

Kuang(2012) Alex Kuang. *Dynamic Difficulty Adjustment*. Tese de Doutorado, Worcester Polytechnic Institute. URL https://www.wpi.edu/Pubs/E-project/Available/E-project-011112-185356/unrestricted/mqp_report_final.pdf. Acessado em dezembro de 2015. Citado na pág. 25

Leal(2016) Ademir Santos Leal. Aplicação do método de aprendizagem por reforço q-learning na adaptatividade dinâmica de dificuldade de um jogo digital ortográfico. march 2016. Citado na pág. 33, 63

Lecheta(2010) Ricardo R. Lecheta. *Google Android: aprenda a criar aplicações para dispositivos móveis com Android SDK*. Novatec Editora Ltda. Citado na pág. 8

Lee *et al.*(**2005**) Valentino Lee, Heather Schneider, e Robbie Schell. *Aplicações móveis: arquitetura, projeto e desenvolvimento*. Pearson Makron Books. Citado na pág. 2

Lemle(2009) Miriam Lemle. Guia teórico do alfabetizador. Citado na pág. 9

Lemos(1997) André Lemos. Anjos interativos e retribaliação do mundo. sobre interatividade e interfaces digitais. *Tendências XXI*. URL http://www.facom.ufba.br/ciberpesquisa/lemos/interac.html. Acessado em abril de 2015. Citado na pág. 5

Liberty(2005) Jesse Liberty. *Programming C#: Building. NET Applications with C#*. "O'Reilly Media, Inc.". Citado na pág. 22

Lima e Reis(2002) Edwin Lima e Eugênio Reis. C# e. net–guia do desenvolvedor. *Rio de Janeiro: Campus*. URL http://www.etelg.com.br/paginaete/downloads/informatica/apostila2.pdf. Acessado em dezembro de 2015. Citado na pág. 23

Mitchell(1997) Tom M Mitchell. Does machine learning really work? *AI magazine*, 18(3):11. URL http://www.aaai.org/ojs/index.php/aimagazine/article/viewArticle/1303. Acessado em dezembro de 2015. Citado na pág. 28

Monteiro(2009) João Gabriel Gadelha Xavier Monteiro. Propostas para o balanceamento dinâmico de jogos com ajuste de dificuldade e payoff lentos. URL http://www.cin.ufpe.br/~tg/2009-2/jggxm.pdf. Acessado em fevereiro de 2016. Citado na pág. 26

Monteiro(2002) Sildomar Takahashi Monteiro. *Estudo de desempenho de algoritmos de aprendizagem sob condições de ambiguidade sensorial*. Tese de Doutorado, Brasil. Ministério da Defesa. Comando-Geral de Tecnologia Aeroespacial (CTA). Instituto Tecnológico de Aeronáutica (ITA). Citado na pág. xii, 31

Monteiro e Ribeiro(2004) Sildomar Takahashi Monteiro e Carlos HC Ribeiro. Desempenho de algoritmos de aprendizagem por reforço sob condições de ambiguidade sensorial em robótica móvel. *Sba: Controle & Automação Sociedade Brasileira de Automatica*, 15(3): 320–338. URL http://www.scielo.br/scielo.php?pid=S0103-17592004000300008&script=sci_arttext&tlng=pt. Acessado em fevereiro de 2016. Citado na pág. 30

Morais(2007) Artur Gomes de Morais. A norma ortográfica do português: o que é? para que serve? como está organizada? *Ortografia na sala de aula*, página 11. URL http://www.lugaresesaberes.com.br/materialApoio/Ortografia%20e%20sala%20de%20aula.pdf#page=12. Acessado em fevereiro de 2016. Citado na pág. 9

Nobile e Barrera(2009) Gislaine Gasparin Nobile e Sylvia Domingos Barrera. Análise de erros ortográficos em alunos do ensino público fundamental que apresentam dificuldades na escrita. *Psicologia em Revista*, 15(2):36–55. URL http://pepsic.bvsalud.org/scielo.php?script=sci_arttext&pid=S1677-11682009000200004. Acessado em junho de 2015. Citado na pág. 9

Nunes(1998) Paulo Nunes. Educação lúdica: Técnicas e jogos pedagógicos. *Editora Loyola, São Paulo*. Citado na pág.

Pereira e Silva(2009) Lucio Camilo Oliva Pereira e Michel Lourenço da Silva. *Android para desenvolvedores*. Brasport. Citado na pág. 8

Pereira(1988) Luís Moniz Pereira. Inteligência artificial: mito e ciência. *Revista Colóquio-Ciências*, 3:1–13. URL http://www.egov.ufsc.br/portal/sites/default/files/anexos/6511-6510-1-PB.pdf. Acessado em dezembro de 2015. Citado na pág. 27

Pinheiro(2007) Cristiano Max Pereira Pinheiro. *para uma aproximação entre jogos digitais e comunicação.* Tese de Doutorado, Pontifícia Universidade Católica do Rio Grande do Sul. URL http://xa.yimg.com/kq/groups/17172435/1930471227/name/397758.pdf. Acessado em abril de 2015. Citado na pág. 5

Prauchner(2014) Darlinton Carlos Krupp Prauchner. Aplicação do algorítmo de inteligência artificial sarsa à robótica na tarefa de coleta de lixo. Citado na pág. 32

Rummery e Niranjan(1994) Gavin A Rummery e Mahesan Niranjan. On-line q-learning using connectionist systems. Citado na pág. 30

Russel e Norvig(2004) Stuart Russel e Peter Norvig. Inteligência artificial traduçao da 2a. edição, 2004. Citado na pág. xii, 27, 28, 29

Sampaio(2012) Maria Nobre Sampaio. Desempenho ortográfico de escolares do ensino fundamental: elaboração e aplicação de um instrumento de intervenção. URL http://repositorio.unesp.br/bitstream/handle/11449/91216/sampaio_mn_me_mar.pdf?sequence=1&isAllowed=y. Acessado em junho de 2015. Citado na pág. 9

Scattone e Masini(2007) Cristiane Scattone e Elcie FS Masini. O software educativo no processo de ensino-aprendizagem: um estudo de opinião de alunos de uma quarta série do ensino fundamental. *Revista Psicopedagogia*, 24(75):240–250. URL http://pepsic.bvsalud.org/scielo.php?script=sci_arttext&pid=S0103-84862007000300004. Acessado em maio de 2015. Citado na pág.

Silva *et al.*(2009) Maycon Prado Rocha Silva, Paula Dornhofer Paro Costa, Paulo Sérgio Prampero, e Vera Aparecida de Figueiredo. Jogos digitais: definições, classificações e avaliação. *Tópicos em*. URL http://www.dca.fee.unicamp.br/~martino/disciplinas/ia369/trabalhos/t1g1.pdf. Acessado em maio de 2015. Citado na pág. 6

Sprague e Ballard(2003) Nathan Sprague e Dana Ballard. Multiple-goal reinforcement learning with modular sarsa (0). Em *IJCAI*, páginas 1445–1447. Citeseer. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.9.7282&rep=rep1&type=pdf. Acessado em fevereiro de 2016. Citado na pág.

Tatai(2003) Victor Kazuo Tatai. Técnicas de sistemas inteligentes aplicadas ao desenvolvimento de jogos de computador. *Proposta de Tese de Mestrado–Processo FAPESP 00/07631-6*. URL ftp://ftp.dca.fee.unicamp.br/pub/docs/gudwin/publications/TeseTatai.pdf. Acessado em dezembro de 2015. Citado na pág. 27

Tiellet *et al.*(2007) Cláudio Afonso Tiellet, Gilse Antoninha Morgental Falkembach, Nires Metilde Colleto, Larisa Rosa dos Santos, e Patric da Silva Ribeiro. Atividades digitais: seu uso para o desenvolvimento de habilidades cognitivas. *RENOTE*, 5(1). URL http://www.seer.ufrgs.br/renote/article/download/14152/8087. Acessado em maio de 2015. Citado na pág. 6

Unity(2015) Unity. O *software* líder global da indústria de jogos, 2015. URL https://unity3d.com/pt/public-relations. Acessado em outubro de 2015. Citado na pág. xii, 21, 22

Unity3D(2015) Unity3D. Scripting, 2015. URL http://docs.unity3d.com/Manual/ScriptingSection.html. Acessado em novembro de 2015. Citado na pág. 22

Unity3D(2016) Unity3D. Obtenha o unity, 2016. URL https://unity3d.com/pt/get-unity. Acessado em fevereiro de 2016. Citado na pág. 21

Zorzi e Ciasca(2009) Jaime Luiz Zorzi e Sylvia Maria Ciasca. Análise de erros ortográficos em diferentes problemas de aprendizagem. *Revista CEFAC*, 11(3):406–416. URL http://www.scielo.br/pdf/rcefac/v11n3/a07v11n3.pdf. Acessado em junho de 2015. Citado na pág. 9