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

Bacharelado em Sistemas de Informação

Aplicação do método de aprendizagem por reforço Q-Learning na adaptatividade dinâmica de dificuldade de um jogo digital ortográfico

Ademir Santos Leal

Diamantina, MG

Março de 2016

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

Ademir Santos Leal

Aplicação do método de aprendizagem por reforço Q-Learning na adaptatividade dinâmica 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.

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

Diamantina, MG

Março de 2016

Ademir Santos Leal

Aplicação do método de aprendizagem por reforço Q-Learning na adaptatividade dinâmica 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.

Aprovada em _____ de _____ de _____.

BANCA EXAMINADORA:

Luciana Pereira de Assis

_____

Alessandro Vivas Andrade

_____

Adriana Bodolay

_____

À minha família e amigos.

Agradecimentos

Desejo expressar minha sincera gratidão a todos que me apoiaram. Agradeço primeiramente a Deus pela saúde e energia durante essa longa caminhada enfrentando todos os obstáculos. Agradeço a minha família pelo apoio e incentivo em todos os momentos, em especial meus pais, Ilda Oliveira Santos Leal e Sebastião Delgracie Campos Leal e todos os meus irmãos. Agradeço à minha orientadora Prof. Luciana Pereira de Assis e a Prof. Adriana Bodolay pela grande ajuda, paciência e compromisso durante toda a elaboração do trabalho. Agradeço também a todos os meus amigos e colegas.

A natureza é um enorme jogo de xadrez disputado por deuses, e que temos o privilégio de observar. As regras do jogo são o que chamamos de física fundamental, e compreender essas regras é a nossa meta.

Richard Feynman

Resumo

Com os avanços da tecnologia, através de meios tecnológicos, são grandes as possibilidades de representação, interação, comunicação e processamento da informação. Essa nova realidade está tornando o mundo cada vez mais diferente, inclusive em características comportamentais. A presença cada vez mais forte dos dispositivos móveis e dos jogos digitais é um fato que se torna cada vez mais desejável a utilização dessas tecnologias da informação na área da educação, com o objetivo de auxiliar na aprendizagem de conteúdos escolares. Nos últimos anos, pesquisadores descobriram que os jogos educacionais podem ser excelentes ferramentas para auxiliar no processo de aprendizagem, principalmente na aprendizagem de crianças. Este trabalho aborda os benefícios que os jogos digitais educacionais promovem no auxílio da aprendizagem, principalmente os jogos digitais ortográficos. Apresenta a importância da presença de métodos de inteligência artificial nesses jogos como forma de melhorar a eficácia dos mesmos na aprendizagem, tornando-os mais dinâmicos, menos previsíveis, reforçando o aprendizado do usuário nas maiores dificuldades. Este trabalho teve como foco o desenvolvimento de um jogo digital ortográfico denominado *Laça Palavras*, que foi desenvolvido para ser executado em dispositivos móveis Android. O *Laça Palavras* tem como objetivo o auxílio nos treinos ortográficos de palavras da língua portuguesa que possuem a característica da concorrência, que consiste em que duas letras estão aptas a representar o mesmo som, no mesmo lugar. Ainda como foco principal apresenta a implementação de um método de aprendizagem por reforço baseado no algoritmo Q-Learning com o objetivo de mostrar de maneira prática a implementação de um método de inteligência artificial em um jogo digital educacional e principalmente realizar o ajuste dinâmico de dificuldade, tornando o jogo mais eficaz e divertido.

Palavras Chave: Jogos Digitais. Jogos Digitais Educacionais. Jogo Digital Ortográfico. Dispositivos Móveis. Android. Inteligência Artificial. Aprendizagem por Reforço. Q-Learning.

Abstract

With advances in technology, through technological means, are great the possibilities of representation, interaction, communication and information processing. This new reality is making the world increasingly different, including behavioral characteristics. The increasingly strong presence of mobile devices and digital games in people's lives is a fact, which becomes the most desirable use of these information technologies in education in order to assist in the learning of school subjects. In the last years, researchers discovered that educational games can be excellent tools for assistance in the learning process, mainly in the learning of children. This work discusses the benefits that digital educational games promote learning in aid, mainly orthographic digital games. It shows the importance of the presence of artificial intelligence methods in these games as a way to improve their effectiveness in learning, making them more dynamic, less predictable and more effective. This work focused on the development of an orthographic digital game called *Laça Palavras*, which was designed to run on Android mobile devices. The *Laça Palavras* aims to aid in spelling practice Portuguese language words that have the characteristic of competition, consisting of two letters that are able to represent the same sound at the same place. Still the main focus is to present the implementation of a reinforcement learning method based on the algorithm Q-Learning in order to show a practical way to implement a method of artificial intelligence in a digital educational game and mainly perform the Dynamic Difficulty Adjustment, making it the most effective and entertaining game.

Keywords: Digital Games. Educational Digital Games. Orthographic Digital Games. Mobile Devices. Android. Artificial Intelligence. Reinforcement Learning. Q-Learning.

Sumário

Sumário

Introdução 9

1.1 Apresentação 9

1.2 Motivação 11

1.3 Objetivos 12

1.4 Estrutura do Trabalho 13

Referencial Teórico 14

2.1 Definição de Jogo 14

2.2 Jogos Digitais 16

2.3 Os Jogos Educativos 17

2.3.1 Jogos Digitais Educativos 18

2.3.2 Alfabetização 19

2.3.3 Jogos Digitais Ortográficos 20

2.4 Dispositivos Móveis 21

2.5 Inteligência Artificial em Jogos Digitais Educacionais 22

2.5.1 Inteligência Artificial 23

2.5.2 Adaptatividade em Jogos 29

Android OS e Ferramentas Utilizadas 31

3.1 Android OS 31

3.1.1 Versões do Android OS 32

3.1.2 Funcionamento do Android 34

3.1.3 Bibliotecas, Frameworks e Aplicação 36

3.2 Unity 3D 37

3.3 Linguagem C# 39

3.4 SQLite 40

3.5 CorelDRAW 41

Desenvolvimento 43

4.1 Plataforma Escolhida 43

4.2 Objetivo do Jogo 43

4.3 Estrutura do Jogo 44

4.3.1 Pasta animation 45

4.3.2 Pasta backgrounds 45

4.3.3 Pasta fontes 46

4.3.4 Pasta material 47

4.3.5 Pasta Plugins 47

4.3.6 Pasta scenes 48

4.3.7 Pasta scripts 49
4.3.8 Pasta sprites 50
4.3.9 Pasta StreamingAssets 50
4.3.10 Pasta áudios 51

4.4 Funcionamento do Jogo 52
4.4.1 Tela Inicial 52
4.4.2 Tela de criação de novo usuário 55
4.4.3 Tela usuários 57
4.4.4 Tela de escolha do personagem 60
4.4.5 Tela principal do jogo 62

4.5 Algoritmo de Inteligência Artificial 73
4.5.1 Aprendizado de Máquina 73
4.5.2 Aprendizagem por Reforço 74
4.5.3 Algoritmo Q-Learning 75
4.5.4 Descrição do Algoritmo 78

Testes e comportamento do Laça Palavras 86

Conclusão e Trabalhos Futuros 93

Referências 94

APÊNDICE A – Conjunto de palavras do jogo Laça Palavras 99

Capítulo 1

Introdução

Esta monografia faz uma abordagem sobre a aplicação de métodos de inteligência artificial em jogos digitais educacionais para dispositivos móveis. Essa abordagem consiste em propor um jogo digital educacional, bem como inserir um método de Inteligência Artificial nesse jogo. Este trabalho apresenta uma implementação de um mecanismo de inteligência artificial baseado na aprendizagem por reforço, mais especificamente no Q-Learning em um jogo digital educacional chamado *Laça Palavras* desenvolvido para a plataforma Android. O jogo tem como objetivo auxiliar o aprendizado da ortografia de palavras da língua portuguesa que possuem a característica da concorrência entre letras, que consiste em que duas letras estão aptas a representar o mesmo som, no mesmo contexto. Este capítulo tem como objetivo contextualizar o problema, apresentar a motivação e os objetivos a serem alcançados e descrever a organização deste trabalho.

1.1 Apresentação

A escola é um espaço que direciona o aprendizado, principalmente nos níveis iniciais de estudo. Esse ambiente é composto por diversas habilidades a serem desenvolvidas e deve providenciar formas para que as mesmas sejam exploradas. DIAS (2009) afirma que, através de relatos de professores que convivem diariamente com alunos que cursam esses níveis iniciais, é verificado que há determinadas dificuldades no ensino da ortografia da língua portuguesa e torna-se necessário pensar em estratégias para ensiná-las.

Atualmente, a tecnologia está bem diferente de algumas décadas atrás, através de meios tecnológicos são grandiosas as possibilidades de representação, interação, comunicação e processamento da informação. Essa nova realidade está tornando o mundo cada vez mais diferente, inclusive em características comportamentais. Esta revolução traz como consequência transformações econômicas, sociais, políticas e culturais numa sociedade denominada sociedade informática. Segundo GAMBIM (2013, p. 1):

> *Observa-se, principalmente nos últimos 10 anos, um interesse e uma preocupação cada vez maiores por parte dos educadores a respeito da presença do computador na escola. O interesse parece justificar-se pela possibilidade de novos caminhos para se alcançar uma melhoria da qualidade de ensino; a preocupação talvez fique por conta de desconhecimento da maioria dos professores de como se utilizar dessa tecnologia sem ser substituído por ela.*

Observando essa realidade, como ferramentas de auxílio para ajudar a sanar dificuldades do ensino da ortografia da língua portuguesa, estão os jogos digitais educacionais em dispositivos móveis. Através de jogos se desenvolvem muitas habilidades e conhecimentos e, ainda, aprender de forma lúdica é muito mais prazerosa. Segundo TAROUCO (2004, p. 1):

> *Os jogos divertem os jogadores enquanto motivam, ao mesmo tempo que facilitam o aprendizado e aumentam a capacidade de retenção do que foi ensinado. Os jogos permitem ainda o reconhecimento e entendimento de regras, identificação dos contextos que estas são aplicadas e invenção de novos contextos para a modificação das mesmas.*

KENSKI (1998, p. 60): “as velozes transformações tecnológicas da atualidade impõem novos ritmos e dimensões à tarefa de ensinar e aprender. É preciso que se esteja em permanente estado de aprendizagem e de adaptação ao novo”.

Observando esse contexto, é desejável a utilização dos jogos digitais educacionais no auxílio da aprendizagem.

PEREIRA (2005 *apud* GAMBI, 2013, p. 2):

> *Nos dias atuais, com o avanço da capacidade de processamento dos computadores, pode-se perceber como se tornou eficaz a busca de informações através dos sistemas computacionais. Juntamente com este forte avanço surge a motivação para elaboração de pesquisas e estudos em aplicações que utilizam IA para facilitar o processo de ensino e aprendizagem.*

POZZEBON (2003, p. 2):

> *A partir de meados do século XX, o desenvolvimento da Inteligência Artificial (IA) esteve profundamente ligado à evolução dos computadores. Através deles, tornou-se possível simular vários aspectos da inteligência humana, o que levou o homem a questionar se as máquinas seriam inteligentes (como os seres humanos) e capazes de aprender.*

Portanto, é bastante desejável atualmente que os jogos digitais educacionais utilizem mecanismos de Inteligência Artificial para torná-los cada vez mais eficazes e eficientes no auxílio da aprendizagem. A Inteligência Artificial pode ser utilizada com o intuito de se obter vários objetivos através de várias formas e técnicas. Um conceito bastante potencial de utilização da IA nesse contexto é o uso de experiências passadas como um guia para a solução ou interpretação de novos problemas ou situações. Outro benefício esperado refere-se à possibilidade de reutilizar experiências passadas para guiar o processo de avaliação dos alunos. Com a utilização dessas experiências, pode-se estabelecer os pontos onde cada aluno necessitaria de maior aprofundamento no conteúdo estudado.

Diante desse contexto, este trabalho consiste em apresentar um jogo digital educacional para dispositivos móveis Android com ajuste dinâmico de dificuldade. O mesmo tem como objetivo auxiliar no aprendizado da ortografia de determinadas palavras da língua portuguesa.

Contextualizando a problemática deste trabalho, DIAS (2009) apresenta em sua obra que, ao lidar com alunos que cursam o Ensino Fundamental no brasil, é verificado que há determinadas dificuldades no ensino da ortografia de palavras da língua portuguesa. LEMLE (1998) reforça também sobre a dificuldade do entendimento sobre as relações entre sons e letras na ortografia da língua portuguesa. LEMLE (1998) ainda afirma que um dos casos mais difíceis para a aprendizagem da língua escrita é a denominada concorrência, que consiste em

que duas letras ou mais estão aptas a representar o mesmo som no mesmo contexto. Por exemplo, a palavra *CASA* se enquadra no caso da concorrência, pelo fato de tanto a letra “S”, quanto a letra “Z”, serem aptas a representar o mesmo som no mesmo lugar, porém, na palavra *CASA* especificamente, é correto apenas utilizar-se a letra “S”, por motivo etimológico (origem) da palavra. Esse problema pode ser minimizado através do frequente contato do aprendiz com tais palavras, de forma que os mesmos consigam memorizá-las em sua forma ortográfica correta.

O jogo digital educacional *Laça Palavras*, desenvolvido neste trabalho, visa ajudar a minimizar esse problema, que consiste em um desafio em que várias palavras (que se enquadram no caso da concorrência) devem ser completadas envolvendo um enredo, uma interface amigável e destinado a executar em dispositivos móveis, provendo maior facilidade em sua utilização em diferentes lugares e de maneira mais acessível. O jogo ainda possui um algoritmo de inteligência artificial baseado na aprendizagem por reforço, visando ajustar dinamicamente sua dificuldade durante a execução, para torná-lo mais eficaz no seu papel como auxílio da aprendizagem e mais divertido para o usuário.

1.2 Motivação

A sociedade atual vive uma revolução na forma de se comunicar, interagir, representar e processar informações. O desenvolvimento de novas tecnologias e a disponibilização destas no mercado já alcança todas as áreas da sociedade. As mudanças são constantes, em espaços curtos de tempo novas tecnologias vão surgindo tomando o lugar de outras. Considerando essa realidade tecnológica em todos os espaços de nossa sociedade, há de se pensar nas mudanças provocadas por esse desenvolvimento nas formas de comunicação, antes mediadas de pessoa por pessoa, hoje mediadas também pelas tecnologias.

PERY (2010, p. 107) diz: “E o que pensar das crianças, o que aqui podemos chamar como “nativos digitais”, “experts” no uso dessas tecnologias. Tais mudanças interferem diretamente em um campo essencial, o da educação, fundamental para a transformação da sociedade. ”

Segundo PERY (2010, p. 107):

> *As tecnologias de informação e comunicação (TICs), agora já presentes nos cenários escolares, vêm transformando a vida humana e interferindo na maneira como nos relacionamos em sociedade e como adquirimos conhecimento, instituindo um novo estado de cultura, a “cibercultura”, que se caracteriza pela ampliação dos lugares em que nos informamos e nos quais de alguma forma aprendemos a viver, a sentir e a pensar sobre nós mesmos. Institui-se um novo estado de cultura e se torna necessário que o papel da escola seja ampliado, pois as diferentes tecnologias influenciam os espaços de educação, seja nos discursos, nas atividades e nas maneiras de aprender e ensinar. Integrar a cultura tecnológica ao currículo escolar torna-se desejável.*

Partindo desse cenário, atualmente existem duas tecnologias em conjunto que se tornaram bastante eficazes no auxílio da aprendizagem dessa nova geração que já nasce inserida nessa nova cultura: os jogos digitais educacionais e os dispositivos móveis.

Muitos jovens, seduzidos pelos jogos digitais, permanecem por longos períodos de tempo jogando. Eles são totalmente empenhados nos desafios que são dados buscando superá-los para obter satisfação, dando a impressão de que são imunes a distrações e que nada é capaz de desconcentrá-los. Conseguir desviar a atenção que esses jovens têm pelos jogos e fazê-los ter maior interesse nos estudos não é tarefa fácil. Por isso, cada vez mais o número de pesquisas que une o ensino e diversão nos jogos digitais educacionais tem aumentado. Esses tipos de jogos promovem um ensino de maneira mais interativa, dinâmica e motivadora por parte do aprendiz. Assim, os jogos educacionais se tornam auxiliares importantes do processo de ensino e aprendizagem (SAVI, 2008).

Mas para serem utilizados com fins educacionais, os jogos precisam ter objetivos de aprendizagem bem definidos e ensinar conteúdo das disciplinas aos usuários, ou então, promover o desenvolvimento de habilidades importantes para ampliar a capacidade cognitiva e intelectual dos alunos (GROS, 2003).

O que GROS (2003) afirma em sua obra é de extrema importância: em que medida a Inteligência Artificial pode ajudar na construção de um jogo educativo que desenvolva as estratégias e as habilidades do usuário aprendiz, promovendo a ampliação da capacidade cognitiva e intelectual. Através da Inteligência Artificial torna-se possível simular vários aspectos da inteligência humana. Assim, o jogo fica mais dinâmico e menos previsível, o uso de experiências passadas como um guia para a solução ou interpretação de novos problemas ou situações, a possibilidade de guiar o processo de avaliação dos usuários e reforçar o aprendizado nas maiores dificuldades apresentadas pelo mesmo.

O aumento no desenvolvimento de jogos educacionais já é significativo, porém ainda são pouco difundidos, principalmente os jogos educacionais que seguem metodologias para incorporarem métodos inteligentes. LUSTOSA (2004, p. 9) aponta: “O ensino, por ser uma área que depende da interação e da adaptação, características tipicamente humanas, apresenta muitas dificuldades na implementação de soluções computacionais inteligentes”.

1.3 Objetivos

O objetivo geral desta monografia é propor um jogo digital educacional para plataforma móvel Android com objetivo de auxiliar a aprendizagem da ortografia de palavras da língua portuguesa que possuem a característica da concorrência, que consiste em que duas letras ou mais estão aptas a representar o mesmo som no mesmo lugar. Mostrar a importância da presença de métodos de Inteligência Artificial nos jogos digitais educacionais com o uso de um algoritmo de Inteligência Artificial baseado no Q-Learning para adapatatividade dinâmica de dificuldade do jogo, tornando-o mais dinâmico, mais divertido e reforçando o aprendizado nas maiores dificuldades do usuário.

Objetivos específicos:

- Com base na revisão da literatura, apresentar os benefícios que os jogos digitais educacionais promovem como ferramentas de auxílio no aprendizado de conteúdos escolares para a nova geração de alunos;

- Estudar os métodos de Inteligência Artificial nos jogos digitais educacionais como forma de melhorar a eficácia dos mesmos na aprendizagem, tornando-os mais dinâmicos, menos previsíveis, reforçando o aprendizado do usuário nas maiores dificuldades.

- Desenvolver um jogo digital educacional para plataforma móvel Android com objetivo de auxiliar a aprendizagem da ortografia de palavras da língua portuguesa que possuem a característica da concorrência, consistindo em que duas letras ou mais estão aptas a representar o mesmo som no mesmo lugar. O público-alvo do jogo digital educacional são crianças a partir de 7 anos de idade que dominem o sistema alfabético e que se encontrem no nível de desenvolvimento da escrita ortográfica.

- Implementar um método de Inteligência Artificial, baseado no algoritmo de aprendizagem por reforço denominado Q-Learning para adaptar os níveis de dificuldade do jogo dinamicamente, tornando-o mais eficaz, dinâmico e divertido.

1.4 Estrutura do Trabalho

No capítulo 2, é apresentado um referencial teórico englobando alguns conceitos sobre jogos, jogos digitais, jogos educativos, jogos digitais educativos, alfabetização, jogos digitais ortográficos, dispositivos móveis, inteligência artificial, ajuste dinâmico de dificuldade e sua utilização em jogos digitais e jogos digitais educacionais.

O capítulo 3 apresenta a plataforma Android e as ferramentas utilizadas.

O capítulo 4 apresenta o desenvolvimento do jogo digital ortográfico *Laça Palavras* e a implementação do algoritmo de inteligência artificial.

O capítulo 5 apresenta os testes e o comportamento do jogo *Laça Palavras*.

O capítulo 6 apresenta a conclusão e trabalhos futuros.

Capítulo 2

Referencial Teórico

Este capítulo tem como objetivo explicar os conceitos fundamentais relacionados ao tema para proporcionar um maior entendimento sobre o assunto. A Seção 2.1 é apresentada a definição do que é um jogo, na seção 2.2 mostra os conceitos sobre jogos digitais, na seção 2.3 sobre jogos educativos, jogos digitais educativos, alfabetização e jogos digitais ortográficos, na Seção 2.4 é apresentado conceitos sobre dispositivos móveis e, na Seção 2.5 é apresentado os conceitos de Inteligência Artificial, ajuste dinâmico de dificuldade e sua utilização em jogos digitais e em jogos digitais educacionais.

2.1 Definição de Jogo

Segundo SILVA (2009), é difícil definir corretamente os jogos por causa de alguns obstáculos, principalmente pelo fato do uso liberal do termo “jogo”, que promove uma percepção exagerada da compreensão do mesmo gerando ambiguidades na definição do conceito. SILVA (2009) exemplifica que o conceito jogo é aplicado a qualquer coisa, jogo como brincadeira, jogo como desafio, jogo como elemento do cinismo, jogo como competição, dentre outros.

Segundo WEISZFLOG (2007 *apud* SILVA, 2009) jogo pode ser definido como: brincadeira, divertimento ou folguedo; passatempo; divertimento de crianças em que elas testam suas habilidades, destreza ou astúcia; e um conjunto de regras a ser observado, quando se joga. Assim, essa definição apresenta o jogo como um elemento de divertimento, de atividade prazerosa.

Existem também autores que buscam uma definição mais científica do termo “jogo”, apresentando-o como um elemento de competição.

Segundo SARTINI (2004) o elemento básico de um jogo é o conjunto de participantes (jogadores). Cada jogador tem um conjunto de estratégias e quando cada um escolhe uma determinada estratégia pode-se dizer que tem uma determinada situação em um espaço de situações possíveis. Cada jogador tem interesse ou preferências para cada situação possível no jogo. Falando de forma matemática, cada jogador tem uma função (utilidade) que atribui um valor, que pode ser representado por um número real, que significa o ganho do jogador em cada situação possível no jogo (SARTINI, 2004).

SARTINI (2004) apresenta uma maneira mais formal de representar um jogo definindo os seguintes elementos básicos: existe um conjunto finito de jogadores, que pode ser representado por $G = \{g1, g2, ..., gn\}$. Cada jogador $gi \in G$ possui conjunto finito $Si = \{si1, si2, ..., sim_i\}$ de opções, denominadas “estratégias puras” do jogador gi ($m_i \geq 2$). Um vetor $Si = \{si1, si2, ..., sim_i\}$, onde sij_i é uma estratégia pura para o jogador $gi \in G$, é

denominado um perfil de estratégia pura, logo o conjunto de todos os perfis de estratégia pura formam o produto cartesiano

$$S = \prod_{i=1}^{n} S_i = S_1 \times S_2 \times \ldots \times S_n,$$

denominado espaço de estratégia pura do jogo.

Um pequeno exemplo que SARTINI (2004) aponta, e que é um clássico da teoria dos jogos, pode mostrar de maneira mais clara a definição de jogos e também a dificuldade de se analisar certos tipos de jogos.

O exemplo se dá pelo seguinte contexto: dois ladrões, um chamado Al e outro chamado Bob, são capturados e acusados de um mesmo crime. Ambos estão presos, mas ficam em selas diferentes e não conseguem comunicar entre si. O delegado de plantão sugere a seguinte proposta: cada um pode confessar ou negar o crime, se os dois negarem o crime ambos serão penalizados com 1 ano de prisão. Se os dois confessarem o crime ambos serão penalizados com 5 anos de prisão. Agora se um deles negar o crime e o outro confessar, o ladrão que negou o crime será penalizado com 10 anos de prisão, já o ladrão que confessou o crime será solto imediatamente. Assim, as opções que os acusados têm, ou seja, confessar e negar, são as "estratégias puras" de cada acusado (jogador). E todas as possíveis opções que os dois acusados podem escolher dentro do contexto é denominado espaço de estratégia pura do jogo.

Entretanto, um jogo também pode ter uma característica diferente, como por exemplo, um desafio envolvendo apenas um jogador e um conjunto de regras. SILVA (2009) afirma que as regras definidas em jogos, principalmente jogos digitais, os jogadores ou o jogador se deparam em uma posição que usualmente ele não poderia se tornar, envolvendo desafios e empenhando habilidades baseadas em conhecimentos ou experiências de aprendizado. O jogo desenvolvido neste trabalho se enquadra nessa definição, não envolvendo mais de um jogador ao mesmo tempo e apresentando se como um desafio para o usuário envolvendo um conjunto de regras.

Segundo o dicionário Aurélio, "jogo é atividade física ou mental fundada em um sistema de regras que definem a perda ou o ganho ou simplesmente passatempo". Atualmente sua principal função é divertir as pessoas, mas também pode assumir o papel de um recurso para o auxílio da aprendizagem.

Uma outra ideia bem interessante, é a de HUIZINGA (1951 *apud* LUCCHESE, 2009, p. 2) em que:

> *O jogo pode ser definido como uma atividade lúdica muito mais ampla que um fenômeno físico ou reflexo psicológico, sendo ainda, um ato voluntário concretizado como evasão da vida real, limitado pelo tempo e espaço, criando a ordem através de uma perfeição temporária.*

LUCCHESE (2009, p. 2) ainda afirma que: “o jogo tem uma característica muito importante que o desconhecimento do seu final. Isso se dá pelo fato de existirem em seu ambiente vários fatores, internos e externos”.

2.2 Jogos Digitais

Com o avanço dos computadores e microcomputadores, vários tipos de aplicações têm surgido nas últimas décadas. Com o hardware mais potente e acessível e o aprimoramento das linguagens de programação, as possibilidades que os computadores promovem são grandes. Assim, os computadores tornaram-se máquinas incríveis para criar jogos, principalmente como primeiro objetivo o entretenimento das pessoas.

Segundo SCHUYTEMA *apud* LUCCHESE (2009, p. 8):

> *Schuytema afirma que um jogo eletrônico é uma atividade lúdica formada por ações e decisões que resultam numa condição final. Tais ações e decisões são limitadas por um conjunto de regras e por um universo, que no contexto dos jogos digitais, são regidos por um programa de computador. O universo contextualiza as ações e decisões do jogador, fornecendo a ambientação adequada à narrativa do jogo, enquanto as regras definem o que pode e o que não pode ser realizado, bem como as consequências das ações e decisões do jogador.*

Já BATTAIOLA (2000) menciona que o jogo digital é composto por algumas partes, principalmente três partes. São elas o enredo, motor e interface interativa. O enredo em um jogo digital é composto pelo tema, trama, os objetivos que possui, e a sequência com o qual os acontecimentos surgem. O motor do jogo é o responsável por controlar o ambiente de acordo com as ações e decisões do jogador. E a interface interativa é a parte que permite o jogador comunicar com o motor do jogo de acordo com as entradas possíveis e saídas que são exibidas na tela.

Os jogos digitais também podem ser definidos como jogos que são construídos a partir de bits e bytes, segundo a *Digital Games Research Association* (DIGRA).

Existe uma característica principal que diferencia os jogos digitais dos jogos não-digitais, é a existência de mundos fictícios essencialmente abstratos (JUUL, 2005 *apud* LUCCHESE, 2009). A existência de mundos fictícios se dá pelo fato de os jogos digitais representarem um ambiente lúdico único. LUCCHESE (2009, p. 9) menciona um ponto bastante interessante que sustenta essa ideia: “nos jogos não-digitais acaba surgindo um mundo fictício, mas o mesmo fica limitado ao imaginário de cada participante e não é compartilhado e delimitado como nos jogos digitais”. Isso é um argumento importante que deve ser observado.

2.3 Os Jogos Educativos

Os jogos são utilizados em diversas aplicações, desde a área da economia, política, lazer e até na aprendizagem. Eles fazem parte da nossa vida desde os tempos mais remotos. Os jogos têm uma característica fundamentalmente lúdica que proporciona prazer e satisfação. Segundo HUIZINGA (1951 *apud* AMATE, 2007), os jogos apontam características como:

> *Prazer do jogador, o caráter "não-sério" da ação, a liberdade de ação e sua separação dos fenômenos do cotidiano, a existência de regras, o caráter fictício ou representativo e a limitação no tempo e no espaço para definir o jogo. Além disso, um novo elemento introduzido pelo autor é a natureza improdutiva do jogo. Entende-se que o jogo, por ser uma ação voluntária da criança, um fim em si mesmo, não pode criar nada, não visa um resultado final*.

É muito comum os jogos serem aplicados com intuito de divertir as pessoas, são bastante utilizados como produtos comerciais que tem principalmente como público alvo as crianças. Mas, o que os pesquisadores descobriram é que:

> *Os jogos podem ser ferramentas instrucionais eficientes, pois eles divertem enquanto motivam, facilitam o aprendizado e aumentam a capacidade de retenção do que foi ensinado, exercitando as funções mentais e intelectuais dos jogadores.* (TAROUCO, 2004, p. 1)

Além disso, segundo TAROUCO (2004): "Os jogos também permitem o reconhecimento e o entendimento de regras, identificação dos contextos que elas estão sendo utilizadas e invenção de novos contextos para a modificação das mesmas". TAROUCO (2004) ainda menciona que jogar é participar de um mundo irreal, de mentira, não saber o que está para acontecer e enfrentar desafios em busca de diversão entretenimento e lazer. "Através do jogo se revelam a autonomia, a criatividade, originalidade e a possibilidade de realizar simulações e experimentar situações perigosas e proibidas no nosso cotidiano" (TAROUCO, 2004, p. 1). Assim, os jogos como grande ferramenta motivadora do processo de aprendizagem são denominados jogos educativos ou jogos educacionais.

Existe ainda muitas discussões sobre o que pode ser um jogo educacional, TAROUCO (2004, p. 2) diz: "DEMPSEY, RASMUSSEM e LUCCASSEN (1996) citados por BOTELHO (2004) definem que os jogos educacionais se constituem por qualquer atividade de formato instrucional ou de aprendizagem que envolva competição e que seja regulada por regras e restrições". Assim, consideram-se jogos educacionais qualquer jogo que tenha como objetivo ensinar o jogador algum conteúdo educacional e que estiver pedagogicamente embasado.

2.3.1 Jogos Digitais Educativos

O poder de processamento e possibilidades de criação que os computadores proporcionam é bastante alto. Sem falar nas possibilidades de interação humano-computador através de recursos de interface que estão presentes atualmente.

Dentro desse contexto atual, os jogos digitais estão cada vez mais presentes na sociedade, principalmente com intuito de divertir as pessoas. Segundo ALVES (2008), a comercialização desses jogos está crescendo cada vez mais, apresentando narrativas mais complexas, com melhor jogabilidade, interatividade e realismo dos gráficos, garantindo ao jogador melhor experiência de jogo. Segundo PERY (2010, p. 109): "É interessante observar o fascínio e o deslumbramento dos alunos das séries iniciais em relação aos jogos digitais, pois estes remetem à sensação de prazer e alegria e atendem à necessidade lúdica das crianças".

BATTAIOLA (2000 *apud* AMATE, 2010, p. 35) menciona que:

> *Existem várias classificações de jogos digitais, e cada uma delas obedece a regras que definem características aplicáveis à interface e motor, que influenciam diretamente as técnicas de implementação utilizadas. Os principais tipos de jogos são: Estratégia, Simuladores, Aventura, Passatempo, RPG (Role Playing Game), Esporte e Educativos/Infantil.*

O foco deste trabalho é os jogos educativos. É considerado jogo digital educativo qualquer jogo que tem como objetivo ensinar algum conteúdo educacional, que estiver pedagogicamente embasado e que o jogador interaja com o ambiente do jogo por meio de alguma mídia digital.

Segundo AMATE (2010), os jogos digitais educativos podem conter qualquer uma das características referentes às classificações citadas anteriormente, podem até conter uma combinação delas. Ele dá um exemplo bastante claro:

> *Um jogo de aventura pode envolver enigmas relativos a um templo Maia, ou um castelo medieval e assim tornar-se um jogo educacional sobre história antiga. O que diferencia os jogos digitais educativos dos demais jogos que visam só diversão, é que os primeiros levam em conta critérios didáticos e pedagógicos.* (AMATE, 2010, p. 35)

TAROUCO (2004) expõe uma ideia bem interessante: os jogos digitais educacionais possuem uma abordagem autodirigida em que o usuário aprende por si só, através da descoberta de relações e da interação com o jogo. Isso se dá porque, os níveis de interatividade e jogabilidade em jogos digitais educacionais são bastante significativos. TAROUCO (2004, p. 3) ainda menciona que: "Neste cenário, o professor tem o papel de moderador, mediador do processo, dando orientações e selecionando jogos adequados e condizentes com sua prática pedagógica".

2.3.2 Alfabetização

A alfabetização, segundo o dicionário Aurélio da Língua Portuguesa, é a ação de alfabetizar, de propagar o ensino de leitura. Já SOARES (2003 *apud* PORGOZELSKI, 2010) afirma que alfabetização é um processo muito amplo podendo ser dividido em três principais pontos de vista. O primeiro ponto vê a alfabetização como um processo de aquisição de código escrito e das habilidades de leitura e escrita. Assim:

> *Alfabetizar significa adquirir a habilidade de decodificar a língua oral em língua escrita (escrever) e de decodificar a língua escrita em oral (ler). A alfabetização seria um processo de representação de fonemas em grafemas (escrever) e de grafemas em fonemas (ler).* (SOARES, 2003 *apud* PORGOZELSKI, 2010, p. 9)

Segundo ponto de vista PORGOZELSKI (2010, p. 9) afirma que:

> *A alfabetização seria um processo, mas de compreensão e expressão de significados por meio do código escrito, em que objetivo primordial é a apreensão e compreensão do mundo num sentido mais amplo, podendo ser utilizado para a aquisição de conhecimento como comunicação.*

Finalizando PORGOZELSKI (2010, p. 9) menciona que:

> *O terceiro ponto de vista está relacionado com a conceituação de alfabetização em diferentes sociedades. Nesta perspectiva, a que se levar em consideração a idade da criança, o tipo de alfabetização que será necessário em um determinado grupo social, pois esses aspectos variam de acordo com o contexto em que a criança está inserida.*

Analisando esses três pontos de vista, PORGOZELSKI (2010) observa que o conceito de alfabetização é muito amplo quando o mesmo é pensado num processo que vai além do exercício de codificar e decodificar símbolos escritos, é um conceito que depende de características culturais, econômicas e tecnológicas.

LEMLE (1988) também menciona uma ideia interessante sobre a alfabetização. A autora afirma que o alfabetizando precisa atingir alguns conhecimentos ou capacidades para aprender a ler e a escrever. Primeiramente, é necessário compreender a ligação simbólica entre letras e sons da fala, a segunda capacidade é conseguir fazer distinção entre as letras, a fim de que deixem de ser apenas manchas pretas na página branca. A terceira consiste em adquirir a capacidade de ouvir e ter consciência dos sons da fala, com suas distinções relevantes na língua.

O ponto importante é que o aprendizado da ortografia no período da alfabetização é fundamental e os jogos digitais educativos são ferramentas de grande potencial que podem auxiliar durante o período do aprendizado. É importante mencionar que a fase da alfabetização é um pré-requisito para o aprendizado da ortografia.

2.3.3 Jogos Digitais Ortográficos

Um problema que sempre existiu nas séries iniciais do ensino escolar é a dificuldade que os alunos possuem no aprendizado da ortografia da língua portuguesa. Professores geralmente encontram bastante dificuldade na realização de atividades que envolvem a escrita (DIAS, 2009). Entende-se por ortografia a escrita das palavras de acordo com o dicionário. Os alunos que possuem as maiores dificuldades geralmente são os que recém se apropriaram do código escrito, ou seja, recém-alfabetizados. DIAS (2009) afirma em sua obra que, ao conviver diariamente com alunos que cursam o Ensino Fundamental, foi verificado que há determinadas dificuldades no ensino da ortografia que prejudicam a aprendizagem dos alunos. Segundo LEMLE (1988), o professor das classes de alfabetização é o que enfrenta os maiores problemas linguísticos, consistindo no momento crucial de toda a sequência da vida escolar. MORAIS (2003 *apud* DIAS, 2009, p. 166) também afirma que:

> *A aquisição da ortografia é impulsionada por diversos fatores, tais como a exposição do aprendiz à língua escrita, a frequência de aparecimento das palavras, a regularidade ou não da notação ortográfica.*

LEMLE (1998) menciona também sobre a dificuldade do entendimento sobre as relações entre sons e letras. O caso mais difícil para a aprendizagem da língua escrita é a concorrência, que consiste em que duas letras estão aptas a representar o mesmo som, no mesmo contexto. LEMLE (1998, p. 23) dá alguns exemplos:

> *É o caso da letra S e da letra Z, que são usadas, ora uma, ora outra, para representar o mesmo som [z] entre duas vogais. Temos MESA, mas também REZA. Temos AZAR, mas também CASAR. Do mesmo tipo é a rivalidade entre C-Ç e SS, usados entre vogais para representar aquilo que é sempre o mesmo som [s]: POSSEIRO e ROCEIRO, ASSENTO e ACENTO, PASSO e LAÇO, CAÇADO e CASSADO. Da mesma maneira, o CH e o X competem na representação da fricativa palatal surda (taxa, racha) e o G e o J rivalizam no privilégio de representar a fricativa palatal sonora (jeito, gente, sujeira, bagageiro).*

Como já foi dito em momentos anteriores neste trabalho, os jogos, principalmente jogos digitais, exercem um grande fascínio nas pessoas e são ferramentas instrucionais eficientes. Assim, observando esse contexto, muitos pesquisadores têm se interessado em projetar jogos digitais educativos que tenham por objetivo prover o auxílio no aprendizado da ortografia da língua portuguesa, ou seja, jogos digitais ortográficos. Como é preciso que o aprendiz fique em constante exposição à língua escrita para sanar tais dificuldades que foram mencionadas nos parágrafos anteriores, esses jogos tendem a melhorar bastante a qualidade do ensino por proporcionar aos alunos práticas prazerosas relacionadas ao conteúdo que está sendo aprendido. Os treinos ortográficos se tornam mais dinâmicos a fim de que o estudo não seja feito de forma mecânica, principalmente se o jogo digital ortográfico possui alguma Inteligência Artificial.

O jogo digital ortográfico desenvolvido neste trabalho tem como diferencial realizar o treino ortográfico desse grupo de palavras mencionado por LEMLE (1998). Como é preciso que o aprendiz fique em exposição a tais palavras, o jogo tem como objetivo tornar esse treino mais divertido para o aprendiz.

2.4 Dispositivos Móveis

O crescimento rápido que tem ocorrido nas últimas décadas nas áreas da computação, comunicação celular, redes e serviços via satélite está permitindo que informações e recursos possam ser acessados e utilizados de qualquer lugar a qualquer momento. Segundo MARÇAL (2005), esse novo paradigma computacional é chamado de Computação Móvel. É classificada como a quarta revolução da computação. A primeira grande revolução foi o surgimento dos grandes centros de processamento de dados, seguido logo após pelo surgimento dos terminais na década de setenta, e depois pelo surgimento das redes de computadores e os computadores pessoais na década de oitenta (TONIN, 2012).

A computação móvel passou a ser viável para as pessoas em geral com o surgimento dos dispositivos móveis que, rapidamente, tornaram-se acessíveis por causa do hardware altamente avançado e desenvolvido com componentes pequenos e baratos.

SACCOL (2007) menciona que existem muitas definições que são apresentadas na literatura sobre computação móvel e dispositivos móveis. Às vezes, tais conceitos trazem certas confusões que devem ser evitadas. Serão mostrados logo abaixo alguns conceitos, de modo a esclarecer a questão.

SACCOL (2007) afirma que Dispositivos Móveis ou Tecnologias de Informação Móveis são dispositivos que possuem a característica de mobilidade, que consiste na capacidade de se levar, para qualquer lugar, um dispositivo de Tecnologia de Informação. Assim, ele afirma que um laptop ou um PDA comum, sem a capacidade de acesso a redes sem fio, são tecnologias móveis ou dispositivos móveis. WEILENMANN (2003 *apud* SACCOL, 2007, p. 179) afirma ainda que: “Uma tecnologia móvel ou dispositivo móvel é aquela que é criada para ser usada enquanto se está em movimento”.

Outro conceito que SACCOL (2007) apresenta é as Tecnologias de Informação Sem Fio, que consiste no uso de dispositivos conectados a uma rede ou a outro dispositivo por links de comunicação sem fio. Como exemplo, ele cita as redes de telefonia celular, a transmissão de dados via satélite, Wireless LAN e Bluetooth.

É importante destacar que atualmente a maioria dos dispositivos móveis possuem tecnologia sem fio, ou seja, são dispositivos que possuem a característica da mobilidade e que ao mesmo tempo acessam a redes sem fio. Por exemplo, os smartphones e tablets.

A grande vantagem dos dispositivos móveis é a comodidade que os mesmos proporcionam para os usuários. Os usuários podem utilizá-los em ambientes diferenciados que não poderiam utilizar, por exemplo, um desktop (computador pessoal de mesa). Sem falar na facilidade de serem carregados e estarem sempre disponíveis com um consumo baixo de energia elétrica.

Atualmente, os dispositivos móveis têm se tornado cada vez mais ferramentas de muita utilidade em várias áreas e para vários propósitos. São bastante utilizados para comunicação, para o trabalho, para o lazer e diversão e também para o aprendizado. A sua utilização como ferramenta para o auxílio da aprendizagem é um estudo muito recente, mas que possui bastante potencialidade e otimismo.

AHONEN (2003); SYVÄNEN (2003) *apud* MARÇAL (2005, p. 1) afirma:

> *A utilização de dispositivos móveis na educação criou um novo conceito, o chamado Mobile Learning ou m-Learning. Seu grande potencial encontra-se na utilização da tecnologia móvel como parte de um modelo de aprendizado integrado, caracterizado pelo uso de dispositivos de comunicação sem fio, de forma transparente e com alto grau de mobilidade.*

Diante desse contexto, com os dispositivos móveis cada vez mais acessíveis, o jogo digital educacional proposto neste trabalho foi construído para executar em dispositivos móveis Android. Essa escolha se deu por facilitar o uso do jogo educacional em diferentes contextos e realidades. Os dispositivos móveis podem ser utilizados nos lugares mais remotos, sendo de fácil transporte.

O jogo digital educacional proposto neste trabalho requer um dispositivo móvel acessível (de baixo custo financeiro) para ser instalado e executado. Sua execução não necessita de conexão com a Internet. No capítulo 3, será abordada com mais detalhes sobre a plataforma Android.

2.5 Inteligência Artificial em Jogos Digitais Educacionais

Segundo GAMBI (2013), nos últimos 10 anos, há um grande interesse e uma preocupação cada vez maior por parte dos educadores a respeito do uso das tecnologias da informação na aprendizagem. O interesse parece justificar-se pela possibilidade de novos caminhos e novas formas que a tecnologia proporciona para alcançar uma melhoria na qualidade do ensino (GAMBI, 2013). Mas essa abordagem ainda não é uma realidade, principalmente no Brasil. Uma ideia interessante que GAMBI (2013, p. 2) ressalta é "a preocupação talvez fique por conta de desconhecimento da maioria dos professores de como se utilizar dessa tecnologia sem ser substituído por ela". KENSKI (1998 *apud* GAMBI, 2013) afirma que as rápidas transformações tecnológicas presentes na atualidade impõem novos ritmos e dimensões à tarefa de aprender e ensinar.

Segundo POZZEBON (2003), uma tecnologia existente desde o século XX que trouxe uma grande evolução dos computadores foi a Inteligência Artificial (IA). "Através deles, tornou-se possível simular vários aspectos da inteligência humana, o que levou o homem a questionar se as máquinas seriam inteligentes (como os seres humanos) e capazes de aprender" (POZZEBON, 2003, p. 2).

Segundo PEREIRA (2005 *apud* GAMBI, 2013, p. 2):

> *Com o avanço da capacidade de processamento dos computadores, pode-se perceber como se tornou eficaz a busca de informações através dos sistemas computacionais". Juntamente com este forte avanço surge a motivação para a elaboração de pesquisas e estudos em aplicações que utilizam IA para facilitar o processo de ensino e aprendizagem.*

2.5.1 Inteligência Artificial

Inteligência Artificial (IA) possui definições de pesquisadores importantes da área. Segundo BOURG e SEEMAN (2004 *apud* GALDINO, 2007, p. 6): "IA é o comportamento inteligente demonstrado pelo computador ou máquina que foi criada, ou talvez o cérebro artificial por trás daquele comportamento inteligente". Eles ainda afirmam que: "outros estudiosos acreditam que o estudo de IA não necessariamente tem o propósito de criar máquinas inteligentes, mas o propósito de conseguir o melhor entendimento da natureza da inteligência humana".

Segundo FERNANDES (2003 *apud* DA SILVA, 2015, p. 2):

> *A palavra inteligência vem do latim que se divide em inter (entre) e legere (escolher), inteligência é aquilo que o homem pode escolher entre uma coisa e outra, sendo que a inteligência é o modo de resolver problemas, de realizar tarefas. Então se considera inteligência artificial um tipo de inteligência produzida pelo homem para beneficiar as máquinas de algum tipo de habilidade que simula a inteligência natural do homem.*

Existem outros autores que definem a inteligência artificial de maneira diferente.

FEIGENBAUM (1981 *apud* GAMBI, 2013, p. 5) afirma que:

> *Inteligência artificial é a parte da ciência da computação voltada para o desenvolvimento de sistemas de computadores inteligentes, ou seja, sistemas que exibem características, as quais se relacionam com a inteligência no comportamento do homem. Pode-se citar como exemplo: compreensão da linguagem, aprendizado, raciocínio, resolução do problema.*

Segundo RUSSEL e NORVIG (2004), pesquisas e trabalhos envolvendo IA começaram a ser notados após a Segunda Guerra Mundial, mas o termo foi realmente cunhado em 1956. O primeiro trabalho que foi notadamente reconhecido dentro da área de IA foi realizado por Warren McCulloch e Walter Pitts em 1943. Eles desenvolveram um modelo de neurônios artificiais, onde cada neurônio continha dois estados, o estado de ligado e desligado. RUSSEL e NORVIG (2004) explicam que um neurônio mudava para o estado "ligado" em resposta ao estímulo de um número suficiente de neurônios vizinhos.

Já em 1949, RUSSEL e NORVIG (2004) afirmam que: "Donald Hebb demonstrou uma regra de atualização simples para modificar as intensidades de conexão entre neurônios. Essa regra, batizada de aprendizagem de Hebb, continua a ser um modelo influente até hoje".

> *Vários trabalhos que podem ser caracterizados como IA surgiram desde então, mas, Alan Turing, ao publicar em 1950 o seu artigo "Computing Machinery and Intelligency", articulou uma visão completa da IA, onde apresentou o teste de Turing, aprendizagem por reforço, aprendizagem de máquina e algoritmos genéticos.* (RUSSEL e NORVIG, 2004)

Nos jogos digitais, os projetistas sempre estão interessados em colocar características da inteligência humana em personagens controlados pelo computador para melhorar assim a qualidade do jogo e atraírem mais consumidores. (GALDINO, 2007).

Assim, são utilizadas diversas técnicas de IA, segundo KARLSSON (2005 *apud* GALDINO, 2007) exemplo:

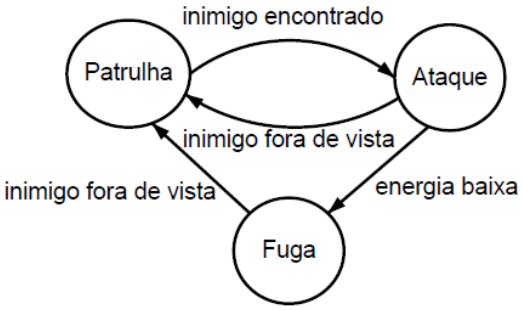
- **Máquinas de Estado Finito:** "são as porcas e parafusos da IA dos jogos" (BOURG e SEEMANN, 2004). Segundo GALDINO (2007, p. 7):

> *As máquinas de estado finito são compostas de um conjunto de estados predefinidos e um conjunto de regras que definem a transição entre os estados, geralmente os estados são alguma ação ou tipo de comportamento que o personagem deve executar e podem também representar algum evento a ser executado no ambiente do jogo.*

Segundo GALDINO (2007), as máquinas de Estado Finito em inglês "*Finite State Machines*" – (FSM's), é o método mais comum e utilizado para representar o comportamento dos personagens. As FSM's definem não somente as ações ou comportamentos do jogo como também um conjunto de definições que determinam quando o estado deve mudar (GALDINO, 2007). O estado atual determina como a FSM deve se comportar, o conjunto de estados são as ações possíveis que o personagem pode executar. Para mudar de um estado para o outro existem os pontos de transição, estes são realizados quando a condição de mudança de estado é alcançada (GALDINO, 2007).

A figura 2.1 exemplifica uma máquina de estados para um jogo fictício apresentado por TATAI (2003):

Figura 2.1: Máquina de estados.



Fonte: TATAI (2003)

O jogo fictício apresentado na figura seria um jogo de guerra, por exemplo, onde o personagem representado pela máquina de estados é uma patrulha. Os estados (círculos) da máquina são as ações do personagem e as transições (setas) são condições de mudança de estado (TATAI, 2003).

- **Path-Finding:** Segundo GALDINO (2007, p. 7):

> *É a busca de caminho entre dois pontos do cenário. Para isso utilizam-se algoritmos específicos, como o A* (lê-se a-estrela) que determinam os caminhos a serem percorridos para o desvio de possíveis obstáculos, inimigos etc.*

Exemplo:

Nos jogos digitais geralmente existem os chamados NPC's (Non-Playable Characters) que em português em tradução livre é "Personagens não jogáveis", eles em determinados momentos se movimentam de um ponto ao outro e durante essa movimentação eles têm de se desviar de obstáculos, demonstrando sinais de inteligência. Assim, para determinar qual caminho que o NPC percorrerá usa-se os algoritmos de path-finding (busca de caminho) (GALDINO, 2007).

Segundo KARLSSON (2005 *apud* GALDINO, 2007), a forma como é feita a busca dos caminhos é executando algoritmos de busca sobre os dados do cenário ou estado do jogo para tentar encontrar o melhor caminho entre a posição de origem e posição de destino.

Um exemplo de algoritmo de busca é o A* (lê-se a-estrela), segundo GALDINO (2007, p. 10):

> *Por ser um algoritmo que é fácil de implementar e também por ser bastante rápido para calcular o menor caminho entre dois pontos, o A* é considerado um dos algoritmos de busca mais utilizados no desenvolvimento dos jogos.*

Assim, geralmente os cenários são organizados em formas de nós no grafo em que os objetos e personagens podem ocupá-los, dependendo do tipo de jogo a representação de tais dados podem se dá de maneira diferente (GALDINO, 2007).

- **Padrões de Movimento:** Segundo GALDINO (2007, p. 7):

> *São técnicas que determinam o comportamento de objetos e/ou personagens do jogo durante a movimentação dos mesmos, essas técnicas pertencem ao conjunto de técnicas que são chamadas de A-Life (Artificial Life).*

Nos jogos digitais, os NPC's se movimentam constantemente no ambiente do jogo, através dos algoritmos de busca de caminho conforme foi abordado anteriormente. Mas, durante essa movimentação é necessário que cada NPC tenha um determinado comportamento diante de uma determinada situação, assim, para gerar uma ilusão de um comportamento inteligente durante tal movimentação, são utilizados os padrões de movimento (GALDINO, 2007).

BOURG e SEEMANN (2004 *apud* GALDINO, 2007, p. 11) abordam um exemplo de implementação de padrão de movimento seguinte:

> *É escolhido o padrão desejado e o mesmo é codificado na central de dados dentro de uma matriz ou conjunto de matrizes. A central de dados consiste em instruções específicas de movimento, tais como andar para frente e girar e a intensidade de que o NPC ou objeto deve se mover utilizando aquele padrão. Utilizando esses algoritmos é possível criar diversos tipos de movimentos como movimento em círculo, em forma de um quadrado, ziguezague e qualquer tipo de movimento que o desenvolvedor consiga programar no conjunto de instruções de movimento.*

Uma ideia bastante interessante que KARLSSON (2005 *apud* GALDINO, 2007) menciona é que o uso combinado entre "*path-finding*" e os padrões de movimento permite criar jogos onde os NPC's e objetos tenham uma autonomia para sua movimentação, juntando as duas técnicas é possível tratar problemas de desvio de obstáculos dinâmicos, fazendo com que o jogo fique mais real e interessante.

- **Sistemas baseados em regras (Rule Based Systems - RBS):** Segundo GALDINO (2007, p. 7):

> *São sistemas onde existe um conjunto de parâmetros e um conjunto de regras, que trabalham sobre os parâmetros de modo a processarem essas regras e exibirem uma saída que pode ser uma ação do personagem.*

Essas regras são utilizadas para fazer inferências ou decidir a tomada de ações. Segundo BOURG e SEEMANN (2004), os sistemas baseados em regras (RBS) são considerados os sistemas de IA mais amplamente utilizados tanto nos jogos quanto nas aplicações para solução de problemas reais. O funcionamento de um RBS é simples, GALDINO (2007, p. 11) diz: "ela faz uso de um conjunto de parâmetros que são trabalhados por um conjunto de regras que no final exibem uma saída, geralmente uma ação do personagem ou comportamento que ele ou algum objeto deverá ter." Assim, o agente ou módulo verifica os parâmetros de acordo com as regras para executar uma determinada ação, as regras são moldadas por condições que são verificadas envolvendo um ou mais parâmetros.

KARLSSON (2005 *apud* GALDINO, 2007) faz o uso de um exemplo bem interessante que torna intuitivo o entendimento de um RBS. O exemplo é de um jogo FPS (*First Person Shooter*) em português pode-se dizer Tiro em Primeira Pessoa. Supõe-se que nesse jogo existe um conjunto de parâmetros: {saúde de um determinado personagem, saúde de um inimigo, distância do inimigo, quantidade de munição}. O personagem possui também ações que podem ser realizadas: {atacar o inimigo, fugir, procurar o inimigo}. Dentro de um RBS as ações do personagem que foram citadas serão escolhidas ao se verificar as condições que as regras impõem para a execução. Um exemplo que GALDINO (2007) menciona seria:

SE (distância do inimigo < 50 E saúde do agente > 50)
ENTÃO atacar o inimigo

O exemplo acima seria uma regra no jogo, nota-se que são avaliados determinados parâmetros que fazem parte do conjunto de parâmetros do jogo para realizar uma ação que faz parte do conjunto de ações. As condições da regra podem ser satisfeitas ou não em um determinado momento, caso sejam satisfeitas a ação será executada, caso não sejam satisfeitas então outras regras serão verificadas (GALDINO, 2007).

Uma grande vantagem dessa técnica segundo BOURG e SEEMANN (2004 *apud* GALDINO, 2007, p. 12) é: "a capacidade de eles imitarem o modo como as pessoas tendem a pensar e a razão de um conjunto de fatos conhecidos e os seus conhecimentos sobre o problema específico do domínio. "

- **Lógica Fuzzy:** Segundo GALDINO (2007, p. 7):

> *A lógica fuzzy ou difusa é uma lógica que admite valores intermediários entre o falso e o verdadeiro, como o "talvez". Esse tipo de lógica analisa as sentenças de forma incerta e procura representar as tomadas de decisão humana.*

Existem certos parâmetros ou situações que os seres humanos avaliam de modo impreciso. Exemplo de uma determinada situação seria: "João é pouco confiável" ou "Maria é muito alta". Assim, a lógica difusa permite representar tais situações, essa técnica utiliza valores intermediários entre o que é totalmente verdadeiro e o totalmente falso. Diferentemente da lógica tradicional booleana que admite apenas valores "verdadeiro" ou "falso" (GALDINO, 2007).

Um exemplo que pode mostrar a utilização de lógica difusa seria um suposto jogo que um determinado personagem possui um certo grau de habilidade específica, como por exemplo, velocidade de movimentação. Essa habilidade pode ser especificada entre 0.0 e 1.0.

Uma ideia interessante que KARLSSON (2005 *apud* GALDINO, 2007) defende é que existe um certo tipo de Máquina de Estado Finito (FSM) que utiliza lógica difusa, chamada de FUZZY FSM (FuSM). Essa técnica tenta resolver os problemas de comportamentos previsíveis das FSM's. Assim, nas transições da máquina, são avaliados valores FUZZY (intermediários) para determinar a mudança ou não de estado.

Uma outra técnica que está sendo bastante utilizada nos jogos digitais e que pode ser utilizada nos jogos digitais educacionais é o Aprendizado de Máquina (AM) ou também denominada na literatura em língua inglesa como "*Machine Learning*" (ML).

O aprendizado está diretamente ligado com a inteligência, aprender e exercer aquilo que foi aprendido é uma característica que os seres humanos afirmam como inteligente. SANTOS (2005) dá uma ideia bem interessante sobre aprendizado:

> *Um processo de aprendizagem inclui a aquisição de novas formas de conhecimento: o desenvolvimento motor e a habilidade cognitiva (através de instruções ou prática), a organização do novo conhecimento (representações efetivas) e as descobertas de novos fatos e teorias através da observação e experimentação. Desde o início da era dos computadores, tem sido realizada pesquisas para implantar algumas destas capacidades em computadores. Resolver este problema tem sido o maior desafio para os pesquisadores de inteligência artificial (IA). O estudo e a modelagem de processos de aprendizagem em computadores e suas múltiplas manifestações constituem o objetivo principal do estudo de aprendizado de máquinas.* (SANTOS, 2005, p. 10).

Segundo MONARD (2003, p. 39):

> *Aprendizado de Máquina é uma área de IA cujo objetivo é o desenvolvimento de técnicas computacionais sobre o aprendizado bem como a construção de sistemas capazes de adquirir conhecimento de forma automática. Um sistema*

> *de aprendizado é um programa de computador que toma decisões baseado em experiências acumuladas através da solução bem-sucedida de problemas anteriores. Os diversos sistemas de aprendizado de máquina possuem características particulares e comuns que possibilitam sua classificação quanto à linguagem de descrição, modo, paradigma e forma de aprendizado utilizado.*

Existem vários métodos de aprendizado de máquina. Esses métodos têm como objetivo realizar o aprendizado por experiência que, conforme a tarefa vai sendo executada, através de percepções, o agente aprende a melhor maneira de resolver ou atuar, além de estruturar o conhecimento existente para levar a um entendimento do aprendizado. (SANTOS, 2005).

Alguns dos principais algoritmos de aprendizado de máquina utilizados em jogos digitais são:

- Árvores de Decisão;
- Redes Neurais Artificiais;
- Algoritmos Evolutivos;
- Algoritmos de Aprendizagem por Reforço.

As técnicas de aprendizado de máquina podem ser utilizadas de diversas formas para diversos objetivos nos jogos digitais. Por exemplo, em um jogo digital educacional, essas técnicas poderiam fazer com que um determinado jogo ensinasse algum conteúdo escolar ao usuário de acordo com suas maiores dificuldades através das experiências realizadas e percepções observadas. Também pode-se fazer com que o jogo perceba que o usuário tem melhorado seu nível de conhecimento e assim dificultar os desafios do jogo gradativamente para tornar-se o jogo mais interessante.

Uma ideia que deve ser levada em consideração é que há uma diferença entre a IA aplicada nos jogos digitais e a IA aplicada puramente no meio acadêmico. Enquanto no meio acadêmico o uso da IA busca resolver problemas complexos, como reconhecimento de sinais e imagens, no desenvolvimento dos jogos digitais, principalmente dos jogos educativos, é utilizada para obter um jogo mais interessante e divertido para o usuário.

Segundo PRETZ (2007, p. 4):

> *As técnicas de IA têm se constituído em objeto de crescente investigação por parte dos pesquisadores da área de informática aplicada, principalmente da educação, devido às suas potencialidades, no que se refere ao desenvolvimento de ambientes de ensino-aprendizagem.*

Os jogos digitais educacionais com métodos de IA exercem forte atração, pela necessidade de participação por parte do usuário, o qual se sente valorizado, tornando parte ativa do processo (PRETZ, 2007).

> *São sistemas que, na interação com o aluno, modificam suas bases de conhecimento através das percepções feitas. Possuem a capacidade de aprender e adaptar estratégias de ensino de acordo com o desempenho do aluno, e se caracterizam, sobretudo, por construir um Modelo Cognitivo*

> *desse Aluno, através da interação, da formulação e da comprovação de hipóteses sobre seu conhecimento. Possui, contudo, a capacidade de adequar estratégias de ensino aprendizagem ao aluno e à situação atual*. (VICCARI, 1990, *apud* PRETZ, 2007 p. 4)

Neste trabalho a IA foi utilizada para realizar a adaptatividade ou ajuste dinâmico de dificuldade de um jogo digital educacional, denominado *Laça Palavras*, para dispositivos móveis. O objetivo do ajuste dinâmico de dificuldade é tornar o jogo mais eficaz no seu papel de auxiliar na aprendizagem e deixá-lo mais divertido para os usuários.

2.5.2 Adaptatividade em Jogos

Um conceito bastante interessante e que não é muito recente nos jogos digitais é a adaptatividade, que também pode ser expresso como balanceamento dinâmico em jogos ou também ajuste dinâmico de dificuldade. É um conceito utilizado geralmente com o objetivo de melhorar a experiência do jogador, aumentando seu interesse pelo jogo fazendo com que o mesmo seja divertido (ARAUJO, 2012).

Sabe-se que a principal expectativa de um jogador ao interagir com um jogo é que o mesmo seja capaz de entretê-lo. Segundo TOZOUR (2002) e ANDRADE (2006 *apud* MONTEIRO, 2009, p. 1): "a capacidade de entretenimento influencia diretamente na qualidade do jogo e pode ser definida por uma série de fatores, como enredo atraente, qualidade gráfica ou sonora e jogabilidade". Para que qualquer jogo tenha essa capacidade, um fator importante é que o jogo precisa ter diferentes níveis de dificuldade em seus desafios para que consiga entreter tanto usuários com poucas habilidades e conhecimentos quanto usuários mais habilidosos (MONTEIRO, 2009).

Para KOSTER (2004 *apud* MONTEIRO, 2009, p. 5):

> *O balanceamento ou ajuste de dificuldade em jogos pode ser definido como o processo que visa garantir um bom nível de desafios tentando evitar extremos tais como frustrar o jogador nos casos em que a dificuldade do jogo está muito elevada ou entediá-lo nos casos em que o jogo só apresenta desafios simples.*

ANDRADE (2006 *apud* MONTEIRO, 2009) ainda afirma que ajustar ou balancear não significa somente atuar em um nível mediano de desafio, mas sim é preciso considerar o aumento de habilidades e conhecimentos do usuário ao longo do tempo e adaptar os novos desafios a esse aumento. Também é preciso considerar que as habilidades dos jogadores podem regredir no decorrer do tempo.

Segundo SILVA (2015) balanceamento de dificuldade ou ajuste dinâmico de dificuldade, consiste em fazer modificações aos parâmetros, cenários e comportamentos do jogo em ordem para evitar a frustração durante a interação aos desafios do jogo. Esse ajustamento permite ao jogo adaptar-se a cada jogador, fazendo o usuário ficar entretido ao longo do jogo. Para isso ser possível o ajuste de dificuldade deve satisfazer três requisitos

básicos. Primeiramente, identificar e se adaptar ao nível do jogador o mais rápido possível. Segundo, identificar e registrar as mudanças de desempenho do usuário. E, por último, o processo de ajuste não deve ser explicitamente percebido pelo usuário, mantendo o jogo coerente (SILVA, 2015).

A adaptatividade também pode ser realizada de maneira estática, também conhecida como *off-line*. Neste caso, o ajuste é realizado utilizando dados do jogador antes da jogabilidade estar ativa, geralmente durante o carregamento ou inicialização (ARAUJO, 2012).

A Inteligência Artificial do jogo digital educacional *Laça Palavras*, que foi desenvolvido neste trabalho, foi aplicada justamente na criação da capacidade de ajuste dinâmico de dificuldade. A importância desse ajuste é perceber as dificuldades do usuário e reforçar o treinamento nessas maiores dificuldades para saná-las. Detalhes sobre a Inteligência Artificial do jogo educacional *Laça Palavras* são mencionados no capítulo 4.

Capítulo 3

Android OS e Ferramentas Utilizadas

Este capítulo abordará de maneira breve alguns aspectos sobre o sistema operacional Android e ferramentas utilizadas no desenvolvimento do jogo digital educacional *Laça Palavras*. Dentre elas, o Unity 3D, a linguagem C#, o Corel Draw e o SQLite.

3.1 Android OS

Os dispositivos que estão em crescente uso atualmente são os dispositivos móveis, que consistem geralmente em smartphones e tablets para uso geral. Esse crescente número de usuários se dá principalmente pela mobilidade que esses dispositivos proporcionam em conjunto com várias outras tecnologias, tornando-os bastante úteis em diversos contextos de uso e ao mesmo tempo proporcionando uma boa comodidade aos usuários. Com a ajuda dos avanços das redes de telefonia móvel 3G e 4G, redes *Wireless*[1], sistemas operacionais otimizados para arquitetura ARM[2], NFC[3], *Bluetooth*[4], entre outros, os dispositivos móveis passam a ser não somente úteis para telecomunicações como também para ambientes coorporativos, entretenimento, lazer e no auxílio da aprendizagem.

Um grande avanço que tornou os dispositivos móveis muito mais populares foi a melhoria dos sistemas operacionais. O Android OS é um grande exemplo: atualmente é a plataforma que está mais presente nos dispositivos móveis, segundo a IDC (2015). Existem outras plataformas, por exemplo, iOS desenvolvido pela empresa norte-americana Apple Inc., Windows Phone desenvolvido pela empresa também norte-americana Microsoft, BlackBerry OS desenvolvido pela empresa *Research in Motion* (RIM) e utilizado nos *smatphones* fabricados pela empresa BlackBerry, Symbian desenvolvido pela empresa Nokia, dentre outros (FILHO, 2016).

Segundo TANENBAUM (2000), um sistema operacional pode ser definido utilizando dois pontos de vista. O primeiro ponto de vista é que o sistema operacional tem a função de apresentar ao usuário o equivalente a uma máquina estendida ou máquina virtual que é mais fácil de programar e manipular que o hardware subjacente. O segundo ponto de vista é que o sistema operacional tem a função de gerenciar todas as partes de um sistema complexo, assim, o trabalho do sistema operacional é oferecer uma alocação ordenada e controlada dos processadores, das memórias e dos dispositivos de entrada/saída entre os vários programas que competem por eles (TANENBAUM, 2000).

---

[1] *Wireless:* Em tradução livre para língua portuguesa significa *sem fio*, conexão para transmissão de informações sem o uso de fios ou cabos.

[2] ARM: Advanced RISC Machine, arquitetura de micro-processadores com um conjunto de instruções reduzidas.

[3] NFC (*Near Field Communication*): Tecnologia de conexão sem fio de dispositivos para transmissão de dados por curta distância.

[4] *Bluetooth*: Tecnologia de conexão sem fio de dispositivos para transmissão de dados por curta distância. Porém permite maior distância e maior velocidade de transmissão que o NFC com consumo baixo de energia elétrica.

O Android OS conseguiu promover uma grande mudança, não somente para os usuários como também, para os desenvolvedores de software. O Android OS desfruta atualmente de um papel de destaque no mercado, tanto pela quantidade significativa de dispositivos produzidos como também por oferecer uma *Application Programming Interface* (API), em tradução livre para língua portuguesa Interface de Programação de Aplicativos, rica, disponibilizando fácil acesso a vários recursos de hardware, tais como *Wi-Fi* e *GPS*, além de boas ferramentas para o desenvolvedor como por exemplo, *Android SDK* e o *Android Studio* (MONTEIRO, 2012).

Alguns anos atrás o desenvolvimento para celulares ou dispositivos móveis específicos, antes mesmo do surgimento dos smartphones e tablets, era praticamente restrito aos fabricantes e operadoras que controlavam o desenvolvimento desses aparelhos. Mas com o aprimoramento e a popularização desses dispositivos os fabricantes passaram a disponibilizar um kit de desenvolvimento (SDK) para que engenheiros e programadores pudessem criar aplicações destinadas às suas plataformas para vários propósitos (MONTEIRO, 2012).

Desenvolvido especialmente para dispositivos móveis como smartphones e tablets, Segundo MONTEIRO (2012, p. 3):

> *O Android é uma plataforma composta de um sistema operacional, middlewares e um conjunto de aplicativos principais como os Contatos, Navegador de Internet e o Telefone propriamente dito em caso de smartphones. Baseado no Linux o Android OS teve seu desenvolvimento iniciado por volta do ano de 2003 pela empresa Android Inc. Mais tarde a empresa foi adquirida pela Google que atualmente lidera o desenvolvimento do Android.*

Segundo MONTEIRO (2012), um acontecimento importante que marcou a popularização do Android OS foi a criação de uma aliança chamada *Open Handset Alliance* em 2007. MONTEIRO (2012) afirma que essa aliança foi criada por empresas de software, hardware e telecomunicações em que o propósito foi a criação de uma plataforma especificamente para dispositivos móveis que fosse completa, aberta e gratuita. Ainda em 2007 a primeira versão beta do Android SDK foi lançada (MONTEIRO, 2012). Assim, uma característica interessante do Android OS é que ele é código aberto e distribuído sob licença Apache 2.0 (MONTEIRO, 2012). Quando algum software é distribuído sob licença Apache 2.0 significa que qualquer programador, engenheiro ou estudante tem acesso ao seu código-fonte e pode ajudar a consertar bugs e aprimorá-lo.

3.1.1 Versões do Android OS

O Android OS atualmente já possui várias versões. A cada nova versão lançada, correções e melhorias são inseridas, buscando de maneira constante que a plataforma supra todas as demandas exigidas pelos usuários e que atenda as novas tecnologias.

Segundo ANDROID (2016) o sistema operacional move mais de um bilhão de smartphones e tablets pelo mundo. O próprio projeto Android afirma que o sistema operacional tem tornado os dispositivos móveis e a vida dos usuários mais doce, acreditando nessa ideia, cada versão do Android OS recebe o nome de uma sobremesa.

O primeiro aparelho com Android OS foi o HTC Dream lançado em 22 de outubro de 2008 nos Estados Unidos. O mesmo possuía a versão 1.0 do sistema. Pouco depois, no início de 2009 foi lançada a versão 1.1 e possuía mais funcionalidades, dentre elas, calculadora, alarme, câmera, contatos, mensagem, música, e outras mais (MAYER, 2015).

Em abril de 2009 foi lançada a versão 1.5 batizada de "Cupcake", foi a primeira versão a receber um nome de sobremesa. Nessa versão foi introduzido a funcionalidade de correção automática nos textos, o bluetooth e também os famosos "widgets" que até hoje são umas das melhores funções do sistema operacional (MAYER, 2015).

> *No final de 2009 foi lançada a versão 1.6 o Android Donut, nessa versão foi adicionado o suporte a resoluções de tela maiores (até 480x800 pixels), nova interface para os aplicativos de câmera e recursos de pesquisa por voz. O grande representante dessa versão é o Sony Ericsson Xperia X10.* (CANALTECH, 2016, p. 1)

Ainda em 2009 foi lançada a versão 2.0/2.1 batizada de "Eclair", o motivo do grande salto em relação ao número da versão é que o Android Eclair trouxe uma nova interface desenhada do zero, tendo suporte a câmeras com flash e um novo e mais eficiente aplicativo de gerenciamento de contatos. A grande popularidade do Android foi iniciada por aparelhos que possuíam essa versão, um grande exemplo foi o famoso Nexus One. Dentre outros recursos, ouve uma melhoria na tela, no *multi-touch* e a incorporação de wallpapers animados (CANALTECH, 2016).

Em maio de 2010 a versão 2.2 chamada de Froyo foi lançada, muito mais rápida que as versões anteriores tinha como destaque um motor melhor de *Javascript*, suporte para criar *hotspot* (compartilhamento de conexão via *Wi-Fi*), suporte ao *Adobe Flash 10.1*, possuía teclado multilíngue, suporte a resoluções altíssimas e integração de um "widget guia" que ajuda a conhecer as funções do Android (CANALTECH, 2016).

Em dezembro de 2010 é lançada a versão 2.3, também com nome de sobremesa, foi chamada de Gingerbread, lançada juntamente com o Nexus S, tinha suporte a NFC (*Near Field Communication*), *VoiP* (Voz sobre IP) inclusive com vídeo conferência, suporte a câmeras frontais, interface mais enxuta visando maior simplicidade e velocidade e novo teclado para digitação rápida (CANALTECH, 2016).

Diferentemente das versões anteriores, a versão 3.0 conhecida como Honeycomb lançada em fevereiro de 2011, foi uma versão para tablet otimizada para telas maiores, sendo apresentada pela primeira vez rodando em um Motorola Xoom e em seguida no Samsung Galaxy Tab 10.1. Teve uma grande melhoria no recurso multitarefas, no gerenciamento de notificações, na personalização da *homescreen* e nos *widgets*. Também acrescentou "*tethering*" (*hotspot* da internet móvel 3G) através do *bluetooth* e suporte integrado para transferir facilmente arquivos multimídia para o PC (CANALTECH, 2016).

Para unificar tablets e smartphones existentes em uma versão unificada, foi lançado em novembro de 2011 o Android 4.0 Ice Cream Sandwich. Teve como objetivo descontinuar o desenvolvimento separado para smartphones e tablets. Essa nova versão trouxe uma interface muito bem-acabada, controle de tráfego de internet, para aqueles que possuem plano de dados limitado, edição simples de vídeos e fotos (via *LiveEffects*), Android Beam para compartilhamento de dados via NFC, desbloqueio de tela por reconhecimento facial e muitos outros recursos (CANALTECH, 2016).

Em 27 de junho de 2012 é lançado mais uma versão do Android OS, agora chamada de Jelly Bean ou 4.1, teve como destaque a melhoria na performance, mostrando se mais veloz, mais fluído e mais reativo aos *inputs*. Teve uma melhoria também bastante interessante que foi o suporte a *widgets* redimensionáveis, a incorporação de um assistente pessoal inteligente o Google Now, suporte a fotos panorâmicas e melhoria nas atualizações dos apps. O Jelly Bean teve vários updates de 2012 até 2013 (MAYER, 2015).

Já em 31 de outubro de 2013 o famoso Android OS 4.4 KitKat foi lançado, teve uma grande otimização no gerenciamento de memória e no *touchscreen* para um *multitasking* (multitarefas) mais veloz. Foi incorporado um novo *framework* para as transições na interface de usuário e suporte para a impressão sem fio (MERCATO, 2015).

O Android OS 5.0 também conhecido como Lollipop, foi lançado em 25 de junho de 2014 e teve substanciais mudanças em relação à versão KitKat. Passou a ter suporte a processadores 64 bits, introduziu uma nova diretriz de design do Android, substituição da máquina virtual Dalvik por Android Runtime (ART), permitiu a exibição de notificações na tela de bloqueio, introdução de recurso multiusuário para smartphones e a incorporação de um modo de economia de bateria (MERCATO, 2015).

A mais recente versão do Android OS (até o momento da escrita deste trabalho) é a versão 6.0 também conhecida como Mashmallow, foi lançada 29 de setembro de 2015 primeiramente para os aparelhos Nexus nos Estados Unidos. Trouxe novidades que a versão anterior não tinha, como por exemplo, o recurso Now on Tap que oferece diferentes opções de interação com o usuário a partir de um scanner dos dados que aparecem na tela, personalização das configurações rápidas, controle de permissões de aplicativos, o uso de abas do Chrome em aplicativos, acesso ao Google Now a partir da tela de bloqueio, incorporação do Android Pay (um sistema de pagamento eletrônico criado pelo Android) e o suporte para pagamentos móveis, melhor gerenciamento inteligente de energia, possibilidade de escolher a banda de frequência *Wi-Fi*, dentre outras (MERCATO, 2015) e (MEYER, 2015).

O jogo digital educacional *Laça Palavras* desenvolvido neste trabalho tem como requisito mínimo a versão Android 2.3.1 (Gingerbread), porém, o jogo foi testado em duas versões, no Android 4.4.4 KitKat e no Android 5.1 Lollipop. Em ambas as versões o jogo executou muito bem, sem grandes travamentos ou problemas.

3.1.2 Funcionamento do Android

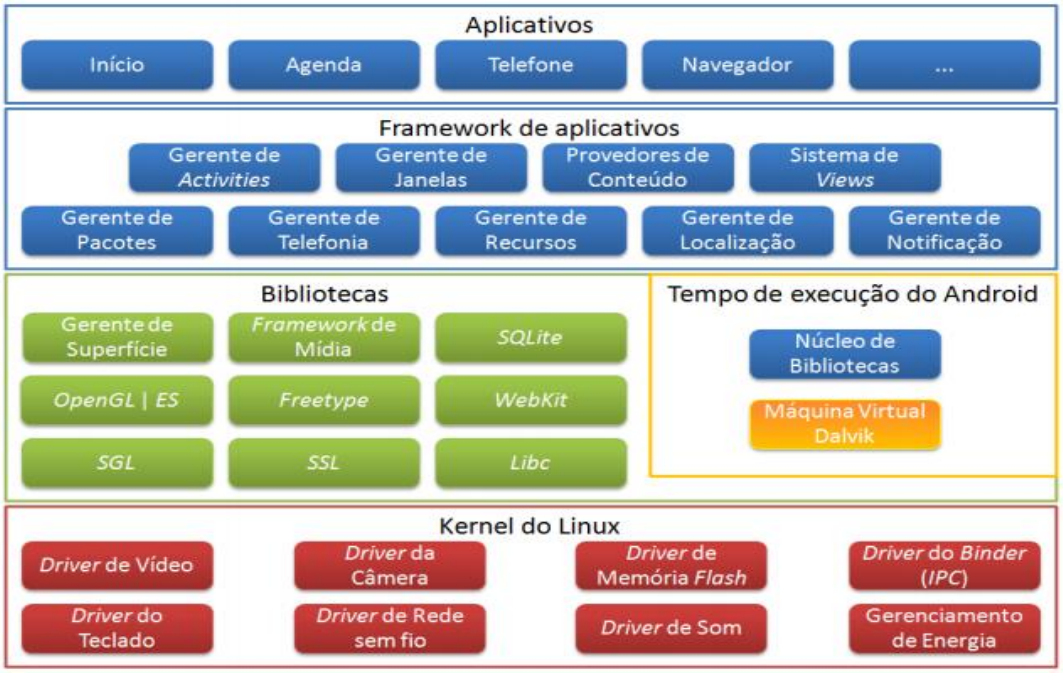
É importante explicar de forma básica os elementos e o funcionamento da plataforma Android para conseguirmos entender como as aplicações funcionam utilizando todos os recursos que a plataforma oferece.

Como já foi dito anteriormente, a plataforma Android consiste basicamente em um sistema operacional, middlewares e uma gama de aplicativos nativos. Geralmente a plataforma é representada em forma de camadas onde as camadas mais abaixo proveem serviços e funções às camadas acima. A camada mais abaixo geralmente é representada pelo kernel 2.6 do sistema operacional Linux, e mais acima uma outra camada composta pelas bibliotecas nativas e Android Runtime (uma espécie de máquina virtual), depois a camada dos *frameworks* de aplicação e por último a camada de aplicação. O gerenciamento dos

processos, *threads*, arquivos, pastas, redes e *drivers* da plataforma são gerenciados pelo kernel do Linux (MONTEIRO, 2012).

A figura 3.1 que exemplifica de maneira básica a arquitetura em camadas da plataforma Android:

Figura 3.1: Arquitetura Android



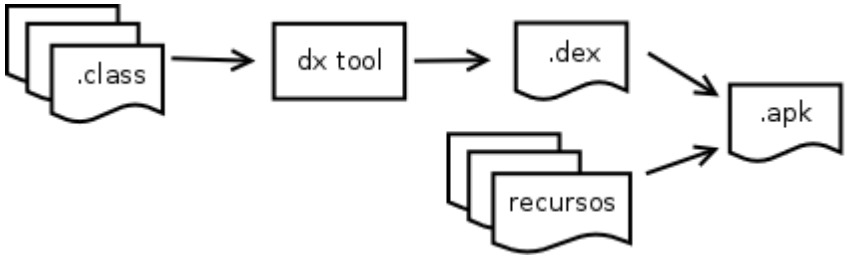
Fonte: http://docplayer.com.br/docs-images/18/753208/images/21-0.png

As aplicações Android são implementadas através da linguagem Java, bastante popular e muito utilizada por vários programadores para diversos fins, é uma linguagem orientada a objetos de fácil aprendizado e que possui um grande número de funções prontas que facilita o desenvolvimento provendo agilidade e produtividade (MONTEIRO, 2012). A execução dessas aplicações é realizada através de uma máquina virtual, baseada em registradores e otimizada para consumir pouca memória, inicialmente foi chamada de Dalvik, mas, atualmente essa máquina virtual sofreu uma modificação, incorporando melhorias e ainda mais otimizada, agora é chamada de Android Runtime ART (MONTEIRO, 2012).

Segundo MONTEIRO (2012), a máquina virtual do Android trabalha um pouco diferente da máquina virtual Java, em vez de executar *bytecodes*, a ART utiliza arquivos no formato ".dex", gerados a partir de classes Java que foram compiladas. Esta conversão ou tradução é feita por uma ferramenta chamada "dx" que está incorporada no Android SDK. O que é feito é o agrupamento de informações duplicadas que encontram-se espalhadas em diversos arquivos ".class" em um arquivo ".dex", com tamanho menor do que os arquivos que o originaram. A ferramenta "dx" faz a conversão de *bytecodes* para um conjunto de instruções específico da máquina virtual ART (MONTEIRO, 2012).

Após ser criado o arquivo “.dex” pela ferramenta “dx”, é feita uma espécie de fusão entre todos os arquivos de recursos da aplicação, como ícones e imagens, e o arquivo “.dex” gerando um único arquivo chamado “.apk” que na verdade é a aplicação Android que pode ser instalada em qualquer dispositivo Android, respeitando os requisitos mínimos, é claro (MONTEIRO, 2012). A figura 3.2 mostra, exemplificando de maneira básica, o processo de geração de uma aplicação Android.

Figura 3.2: Geração de um aplicativo Android



Fonte: MONTEIRO (2012 p. 34).

Até a versão KitKat 4.4 do Android, para cada aplicação é atribuído somente um usuário de sistema e apenas este usuário recebe permissões para acessar os arquivos da aplicação (MONTEIRO, 2012). A partir da versão Lollipop 5.0 o Android passa a ter suporte a múltiplos usuários (MERCATO, 2015). As aplicações são executadas em um processo próprio, ou seja, cada aplicação tem seu próprio processo isoladamente, permitindo a segurança e que nenhuma aplicação seja interferida por outra, assim, cada processo executa uma instância da máquina virtual ART (MONTEIRO, 2012).

3.1.3 Bibliotecas, Frameworks e Aplicação

Continuando o raciocínio de que a plataforma Android possui uma arquitetura em camadas, as aplicações são implementadas utilizando os *frameworks* de aplicação que são ferramentas que simplificam a manipulação de bibliotecas nativas do sistema operacional. Os programadores de aplicações trabalham utilizando esses frameworks tendo acesso a todos os recursos que podem ser incorporados como funcionalidades nas aplicações, assim é por meio dos *frameworks* que o programador acessa a interface gráfica, o armazenamento do dispositivo, GPS e outros mais (STRICKLAND, 2016).

Logo abaixo da camada de *frameworks* de aplicação, estão as bibliotecas nativas do sistema operacional. Geralmente essas bibliotecas nativas são construídas em linguagem C e C++, são elas que fornecem o suporte básico ao sistema, como por exemplo, operações com a interface gráfica, operações com banco de dados, áudio e outros mais. Esse suporte básico é facilitado ainda mais pelos *frameworks*, para que os programadores tenham agilidade e facilidade de criar as aplicações com funcionalidades interessantes (STRICKLAND, 2016).

Por fim, na camada mais alta estão as aplicações, ao serem executadas os usuários conseguem manipular e executar as operações da forma que foram programadas, como por exemplo, tirar uma foto e salvá-la, registrar dados em uma aplicação que controla alguma atividade, utilizar o GPS para obter informações de localizações, dentre outras.

3.2 Unity 3D

O desenvolvimento de um game, seja ele simples ou cheio de efeitos e recursos gráficos, é uma tarefa bastante complexa. Isso se dá pelo fato de que, em sua construção, várias áreas do conhecimento são utilizadas, é preciso diversas habilidades e diversas ferramentas. Atualmente, existem várias ferramentas de qualidade e acessíveis que permitem que projetistas possam construir seus jogos de maneira rápida e com facilidade. Alguns exemplos como, Game Maker, Kodu Game Lab, The Game Factory, Platinum Arts Sandbox 3D Game Maker, CryENGINE 3 Free SDK, Antiryad Gx, RPG Maker VX, Cocos 2D, 3D Rad, Unity 3D, dentre outros. Neste trabalho foi utilizado o Unity 3D.

O Unity 3D é um tipo especial de ferramenta, conhecido como motor de jogos (game engine), é um tipo de ferramenta que foi evoluindo de maneira paralela aos jogos e que hoje são extremamente importantes. Os motores de jogos têm funcionalidades fundamentais como, por exemplo, um sistema de renderização 3D com suporte a *Shaders* programáveis, um sistema de simulação física, uma boa arquitetura para a programação de *scripts*, um editor de cenas integrado e capacidade de importar diretamente modelos 3D, imagens e efeitos de áudio produzidos por ferramentas externas (PASSOS, 2009).

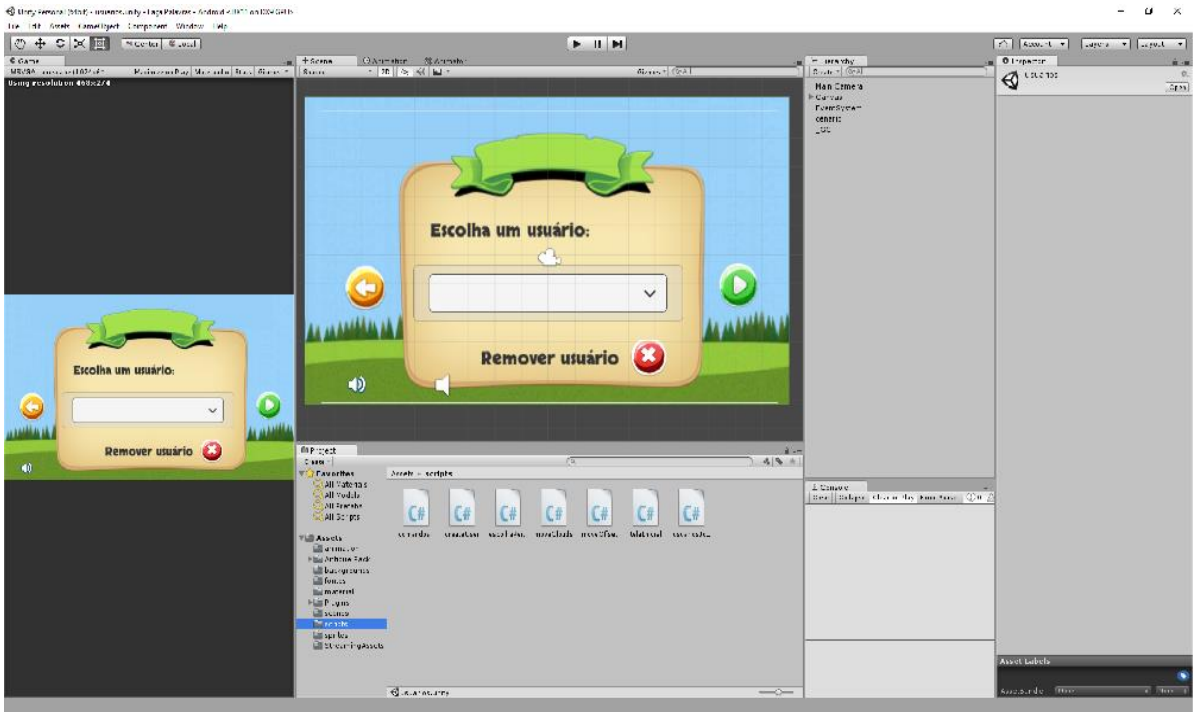
Quando se programa algum software com o intuito de utilizar renderização de gráficos, o projetista precisa utilizar recursos específicos que permita criar esses gráficos para que sejam executados em determinados *hardware* e sistemas operacionais. Atualmente existem dois padrões que são os mais comuns de serem utilizados, o *OpenGL* e o *DirectX*. Esses padrões fornecem APIs (*Application Programming Interfaces*), que são bibliotecas de funções específicas disponibilizadas para a criação e desenvolvimento desses aplicativos em determinadas linguagens de programação (PASSOS, 2009). O Unity 3D utiliza esses dois padrões, porém, no Unity a utilização de tais bibliotecas se torna mais fácil onde o projetista não as manipula diretamente, ou seja, o Unity 3D faz esse papel para o projetista abstraindo vários detalhes (PASSOS, 2009).

Existem versões do Unity 3D, tanto para sistemas operacionais Windows da Microsoft quanto para sistemas operacionais Mac OS da Apple, infelizmente até o momento em que esse trabalho foi escrito, ainda não existe versão do Unity 3D para sistemas operacionais baseados no Linux. Para projetos pessoais e projetos de estudo o Unity 3D possui uma versão gratuita onde a maioria dos seus recursos estão presentes e projetos interessantes podem ser criados. No desenvolvimento do jogo digital educacional *Laça Palavras* realizado neste trabalho, foi utilizada essa versão gratuita.

Uma característica interessante do Unity 3D é que projetistas podem desenvolver tanto jogos 2D como 3D, seu poder é tão grande que pode ser desenvolvido desde jogos 2D simples para dispositivos móveis com baixa capacidade de *hardware*, a até jogos em 3D com detalhes

gráficos pesados para serem executados em *desktops* com as mais modernas placas de vídeo. É importante ressaltar que projetistas podem criar projetos com o Unity 3D para diversas plataformas, como por exemplo, PCs com Windows, Computadores Apple com Mac OS, dispositivos Android, dispositivos IOS e consoles como XBOX e *Play Station*. A figura 3.3 apresenta a interface do Unity 3D:

Figura 3.3: Interface do Unity 3D.



Fonte: Elaborada pelo autor.

Como pode-se observar na figura a interface, do Unity 3D é bem organizada. No topo da interface está o menu principal com todas as opções que envolve a criação e edição de recursos. No lado esquerdo fica uma pré-visualização da cena do jogo que está sendo criada e editada. No meio fica o editor de cenas integrado e logo abaixo fica o acesso aos arquivos do projeto organizados em pastas que podem ser separados em imagens, *scripts* de programação, animações, áudios, dentre outros. Por fim, no lado direito da interface fica a exibição da lista dos nomes referentes aos objetos que fazem parte da cena que está sendo criada, bem como suas informações de propriedades e comportamentos.

3.3 Linguagem C#

Ao construir alguma aplicação, seja ela para executar em desktops, laptops, servidores ou dispositivos móveis, é necessário que o programador utilize alguma linguagem de programação. Existem diversas linguagens que podem ser utilizadas dependendo do foco da aplicação e plataforma, como por exemplo, C, C++, Java, Objective-C, C#, Python, Fortran, Perl, dentre outras. Umas possuem mais recursos que outras, tornando mais fácil a implementação.

No Unity 3D, as aplicações gráficas ou jogos são facilmente criados através da elaboração cenas em um editor integrado que possui opções na interface de fácil acesso onde são manipuladas por comandos ou pelo mouse. Porém, as funcionalidades dos elementos das cenas (personagens, botões, dentre outros) são programadas através de *scripts* nos quais se usa alguma linguagem de programação compatível. O Unity 3D oferece a possibilidade de criar tais *scripts* em duas linguagens, C# ou JavaScript. Assim, o programador pode escolher aquela em que possui maior domínio ou preferência. No projeto do jogo educacional criado neste trabalho os scripts foram programados na linguagem C#.

A linguagem C# foi criada pela Microsoft e segundo o próprio site da empresa MICROSOFT (2016):

> *C# é uma linguagem elegante e de tipos protegidos, orientada a objeto e que permite aos desenvolvedores construírem uma variedade de aplicações seguras e robustas, compatíveis com o .NET Framework. Sua sintaxe é bastante expressiva, mas também é de fácil aprendizado. É facilmente reconhecida por qualquer programador que seja familiarizado com C, C++ ou Java. Os desenvolvedores que sabem qualquer uma dessas linguagens são na maioria das vezes capazes de programar de forma produtiva com C# com pouco tempo de estudo*.

Ainda segundo MICROSOFT (2016):

> *Como uma linguagem orientada a objetos, o C# suporta os conceitos de encapsulamento, herança e polimorfismo. Todas as variáveis e métodos, incluindo o método principal (Main), o ponto de execução de uma aplicação, são encapsuladas em definições de classes. Uma classe derivada pode herdar diretamente somente de uma classe pai, mas pode herdar de qualquer quantidade de interfaces. Métodos da classe derivada que substituem métodos virtuais de uma classe pai exigem a utilização da palavra-chave "override" como forma de evitar a redefinição acidental. Em C#, uma struct (tipo estruturado) é como uma classe simplificada, um tipo alocado em pilha que pode implementar interfaces, mas não suporta herança.*

Diferentemente de algumas outras linguagens de programação, o C# utiliza uma plataforma unificada para sua execução, conhecida como plataforma .Net. Com a ideia um pouco semelhante à plataforma Java, o programador deixa de escrever código para um sistema ou dispositivo específico, e passa a escrever para a plataforma .Net, sua execução é feita sobre um ambiente chamado *Common Language Runtime* - CLR (Ambiente de Execução Independente de Linguagem) interagindo com um Conjunto de Bibliotecas Unificadas

(*framework*). A plataforma .Net é compatível não somente com o C#, mas também com várias outras linguagens (MICROSOFT, 2016).

Seu funcionamento é parecido com Java, o código fonte C# gerado pelo programador é compilado gerando um código intermediário em uma linguagem chamada MSIL (Microsoft Intermediate Language). É esse código em MSIL que é executado pelo ambiente CLR que interpreta para linguagem de máquina, ou seja, código assembly específico da arquitetura do processador (MICROSOFT, 2016).

Em linguagens de programação como C, C++ e Pascal, por exemplo, não requer a presença de ambientes de execução intermediários, pois, o código fonte é compilado para código de máquina específico de uma determinada plataforma e sistema operacional, esse código de máquina específico (*assembly*) será executado e o sistema operacional entenderá perfeitamente. Assim, se um programador quiser que sua aplicação C, C++ ou Pascal rode em várias plataformas e sistemas operacionais ele deverá programá-la e compilá-la para cada uma dessas plataformas e sistemas operacionais.

3.4 SQLite

Quando algum programador, engenheiro ou design cria algum projeto de tecnologia da informação que envolve manipulação dados e que necessita da característica de persistência, ou seja, armazenar esses dados de maneira permanente (não-volátil em um disco rígido ou memória flash), é preciso utilizar sistemas de arquivos de maneira direta ou sistemas de gerenciamento de banco de dados (SGBD). Existem vários sistemas de banco de dados disponíveis que podem ser utilizados, tanto sistemas livres e gratuitos quanto sistemas que cobram algum valor para a aquisição e uso. Alguns exemplos, MySQL, PostgreSQL, Oracle DB, Microsoft SQL Server, dentre outros.

Segundo LAUDON (2011, p. 114), banco de dados pode ser definido como:

> *Um conjunto de arquivos relacionados entre si com registros sobre pessoas, lugares ou coisas. São coleções organizadas de dados que se relacionam de forma a criar algum sentido (Informação) e dar mais eficiência durante uma pesquisa ou estudo.*

LAUDON (2011) ainda busca esclarecer que os sistemas de gerenciamento de banco de dados (SGBDs) não é um banco de dados em si, mas, é um conjunto de softwares responsáveis pelo gerenciamento de um banco de dados. Seu principal objetivo é fazer o papel de gerenciar o acesso, a manipulação e a organização dos dados e deixar que a aplicação cliente faça suas operações com tais dados de maneira bem simples através de uma interface.

No jogo digital educacional *Laça Palavras*, desenvolvido neste trabalho, foi utilizado o SGBD SQLite. Foi necessária sua utilização porque os dados relacionados ao usuário, banco de palavras e dados relacionados a inteligência artificial devem ser salvos para a correta funcionalidade do jogo e possibilitando ao usuário salvar seu jogo e continuar a joga-lo em outra ocasião de onde parou. Mais detalhes sobre o desenvolvimento do *Laça Palavras* no capítulo 4.

Segundo o próprio site oficial SQLITE (2016), o SQLite é uma biblioteca que implementa um banco de dados relacional SQL embutido à aplicação sem necessidade de configuração inicial. É um software livre de domínio público e multiplataforma. Ao utilizar a biblioteca SQLite é possível ter acesso a banco de dados sem executar um processo SGBD separado, o que é bastante recomendável para softwares que são destinados a serem executados em dispositivos com hardware mais modesto, como por exemplo, dispositivos móveis rodando Android. O SQLite lê e escreve diretamente em arquivos de disco comuns.

Por ser bastante leve e compacto obviamente possui algumas limitações quando comparado com outros SGBDs, a sua utilização depende do objetivo da aplicação que está sendo construída, existem alguns exemplos onde é recomendado ou não o uso do SQLite. Segundo SQLITE (2016), é recomendável para aplicações que possuem pouca concorrência, aplicações para sistemas embarcados e dispositivos móveis, aplicações com pouca quantidade de dados, ferramentas de estatística e análise, e sites com poucas requisições por dia. Não é recomendável para aplicações com grande concorrência, aplicações com muita quantidade de dados, aplicações cliente/servidor e sites com muitos acessos. Para aplicações com estas características existem outras opções melhores, como, por exemplo, PostgreSQL, MySQL e Oracle.

Assim sendo, como o jogo digital educacional desenvolvido neste trabalho é destinado a executar em dispositivos móveis Android, o mesmo se enquadra dentre as aplicações que se recomenda o uso do SQLite.

3.5 CorelDRAW

Designs e projetistas ao precisarem criar desenhos gráficos digitais para serem incorporados em seus projetos, necessitam de uma boa ferramenta com bons recursos visando qualidade e produtividade. Existem várias ferramentas disponíveis no mercado que podem ser utilizadas e que produzem resultados bastante interessantes. Alguns exemplos são, Adobe Illustrator, Adobe Photoshop, Inkscape, CorelDRAW, Serif DrawPlus, Xara, Gimp, Skencil, dentre outros. Neste trabalho foi utilizado o CorelDRAW.

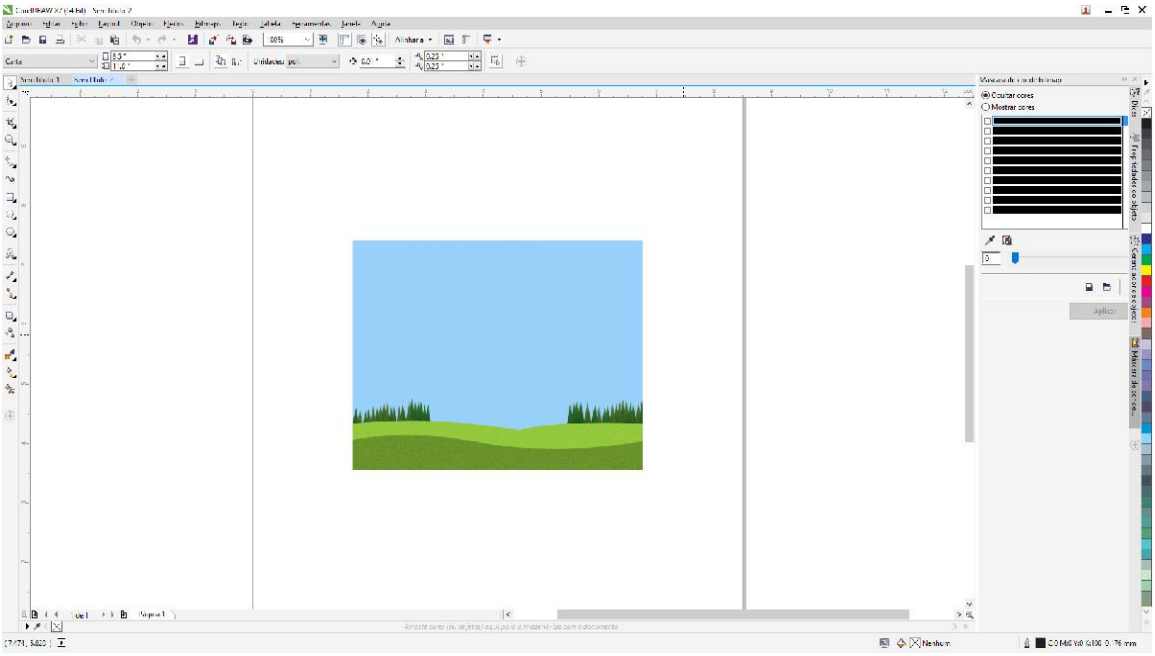
Criado pela empresa canadense Corel Corporation, o CorelDRAW, segundo o próprio site oficial da empresa COREL (2016), é um programa de desenho vetorial bidimensional para design gráfico lançado em 1989 como uma solução gráfica intuitiva que permite que você cause um grande impacto com o seu trabalho artístico. Seja para criar gráficos e layouts, editar fotos ou desenvolver sites, essa suíte completa ajuda você a começar a trabalhar de forma rápida e a manter o rumo.

A Corel tem uma comunidade de mais de 100 milhões de usuários ativos em mais de 75 países e uma rede bem estabelecida de revendedores internacionais, varejistas, fabricantes de equipamentos originais (OEMs), fornecedores on-line e sites globais da Corel. Também possui uma boa documentação que possibilita uma grande facilidade para que novos usuários consigam trabalhar em pouco tempo de estudo (COREL, 2016).

Neste trabalho o CorelDRAW foi utilizado para criar alguns desenhos na interface das cenas do jogo educacional. A versão utilizada foi a versão gratuita de avaliação que possui duração de 30 dias. Esses 30 dias de duração foi suficiente para criar os desenhos necessários.

Uma imagem da interface principal do CorelDRAW Graphics Suite X7 é apresentada na figura 3.4.

Figura 3.4: Interface principal do CorelDRAW Graphics Suite X7.



Fonte: Elaborada pelo autor.

Capítulo 4

Desenvolvimento

O presente capítulo explicará os passos do desenvolvimento do jogo digital educacional *Laça Palavras* e a implementação do algoritmo de inteligência artificial Q-Learning.

4.1 Plataforma Escolhida

Para o desenvolvimento deste jogo educacional foi escolhido que sua execução fosse direcionada aos dispositivos móveis. Conforme foi abordado no capítulo 2 deste trabalho, o sistema operacional Android é uma plataforma livre e gratuita tanto para uso quanto para o desenvolvimento de aplicações. Também é a plataforma que está mais presente nos dispositivos móveis, além de possuir muitos materiais de fácil acesso para estudo. Pela relevância destes fatores e pelo fato de os dispositivos móveis estarem bastante acessíveis atualmente, o *Laça Palavras* foi desenvolvido para ser executado em dispositivos móveis que tem como sistema operacional o Android. É recomendável que o dispositivo móvel possua mais de 512 MB de memória RAM e Android 2.3.1 ou versão superior.

4.2 Objetivo do Jogo

Como foi abordado no capítulo 2, importantes autores da área da alfabetização mencionam a existência de dificuldades que os alunos das séries iniciais do ensino escolar possuem no aprendizado da ortografia da língua portuguesa. Também, autores da área dos jogos digitais explicam como os jogos fascinam as pessoas e são considerados ferramentas instrucionais eficientes. Diante desse contexto, o jogo *Laça Palavras* tem como objetivo principal auxiliar as crianças em fase de alfabetização no aprendizado da ortografia da língua portuguesa. O jogo tem como foco tornar bem mais divertido e menos cansativo os treinos ortográficos das palavras, principalmente palavras que possuem o caso da concorrência (conceito explicado no capítulo 2), que consiste em que existirem duas letras aptas a representar o mesmo som, no mesmo contexto. Essas palavras tendem a gerar muitas confusões nos aprendizes durante a fase inicial do aprendizado.

O jogo foca em quatro grupos de palavras que se enquadram no caso da concorrência. Dentre elas, palavras que contém sílabas com “S” ou “Z”, mas que possui som de [z] (exemplo a palavra “casa”), palavras que contém sílabas com “SS” ou Ç, mas que possui som de [s] (exemplo a palavra “passo”), palavras que contém sílabas com “L” ou “U”, mas que possui som de [u] (exemplo a palavra “alça”) e palavras que contém sílabas com “G” ou “J”, mas que possui som [g] (exemplo a palavra “majestade”). O conjunto de palavras que o *Laça Palavras* possui é apresentado no APÊNDICE A.

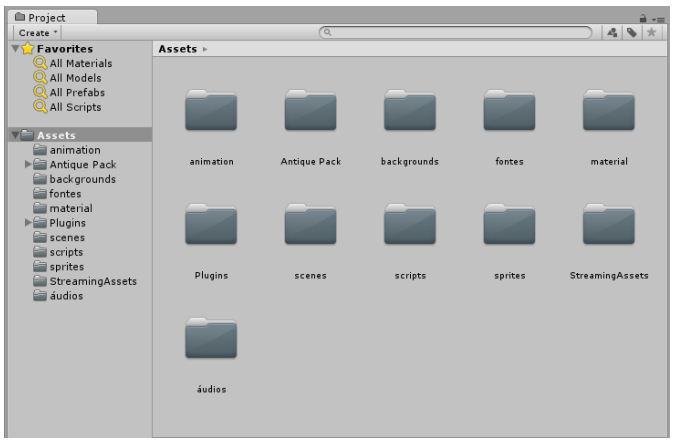
O *Laça Palavras* consiste em apresentar ao usuário diversas palavras referentes aos quatro grupos mencionados para serem completadas corretamente de acordo com as opções existentes, à medida em que o usuário vai acertando o mesmo vai acumulando acertos.

O *Laça Palavras* possui um recurso que permite ao usuário ouvir frases contextualizando as palavras a serem completadas durante a execução do jogo. O usuário pode clicar em um botão na interface e ouvir a frase percebendo os sons da fala envolvendo a palavra. Para cada palavra existe uma frase que pode ser ouvida. Esse recurso faz o *Laça Palavras* ser um diferencial diante de outros jogos ortográficos, uma vez que o usuário tem a possibilidade de perceber o som da fala pronunciando a palavra apresentada na tela. A justificativa desse recurso é fazer com que a criança perceba que ouvir o som da fala pronunciando a palavra não faz diferença na escolha da letra. A criança só aprende a ortografia memorizando a escrita da palavra. O público-alvo do jogo digital educacional são crianças a partir de 7 anos de idade que dominem o sistema alfabético e que se encontrem no nível de desenvolvimento da escrita ortográfica. Mais adiante o jogo será explicado com mais detalhes. O *Laça Palavras* foi desenvolvido com a colaboração de SANTOS (2016).

4.3 Estrutura do Jogo

Como foi mencionado no capítulo 3, o *Laça Palavras* foi desenvolvido no Unity 3D. Todo projeto de desenvolvimento realizado no Unity 3D possui uma estrutura básica que deve ser seguida. Todos os arquivos referentes aos recursos de criação do projeto ficam localizados em um diretório raiz chamado Assets e são separados por subpastas de acordo com cada tipo de arquivo, são arquivos como cenas, imagens, sprites, scripts com códigos, dentre outros. O diretório Assets fica localizado na parte inferior da interface do Unity 3D. Abaixo segue uma imagem mostrando a estrutura de pastas do projeto *Laça Palavras*.

Figura 4.1: Estrutura de pastas do *Laça Palavras*.

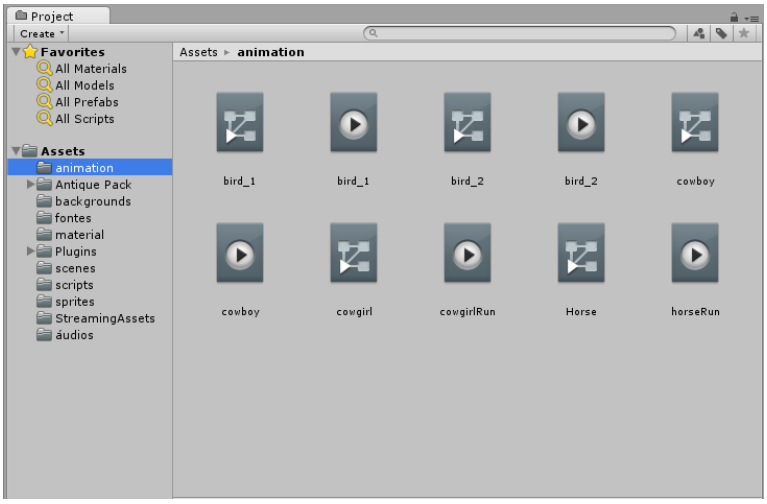


Fonte: Elaborada pelo autor.

4.3.1 Pasta animation

A pasta *animation* contém todas as animações criadas para o jogo. Animações em 2D são facilmente criadas através do mouse ou por comandos no Unity 3D. É necessário inicialmente criar as imagens em uma ferramenta externa e salvá-las em formato PNG ou JPG no diretório de imagens, é mais comum utilizar o formato PNG. Também é bastante comum colocar as imagens que irão compor a animação em forma de Sprite, onde todas as imagens na verdade ficam ordenadas em só uma imagem com fundo transparente. As imagens que compõe uma animação são desenhadas em uma determinada ordem em que quando são postas em transição rapidamente transmitem uma sensação de movimento, um exemplo onde essa técnica é bastante utilizada também é nos desenhos animados. A pasta *animation* é mostrada na figura 4.2.

Figura 4.2: Pasta *animation* do projeto *Laça Palavras.*



Fonte: Elaborada pelo autor.

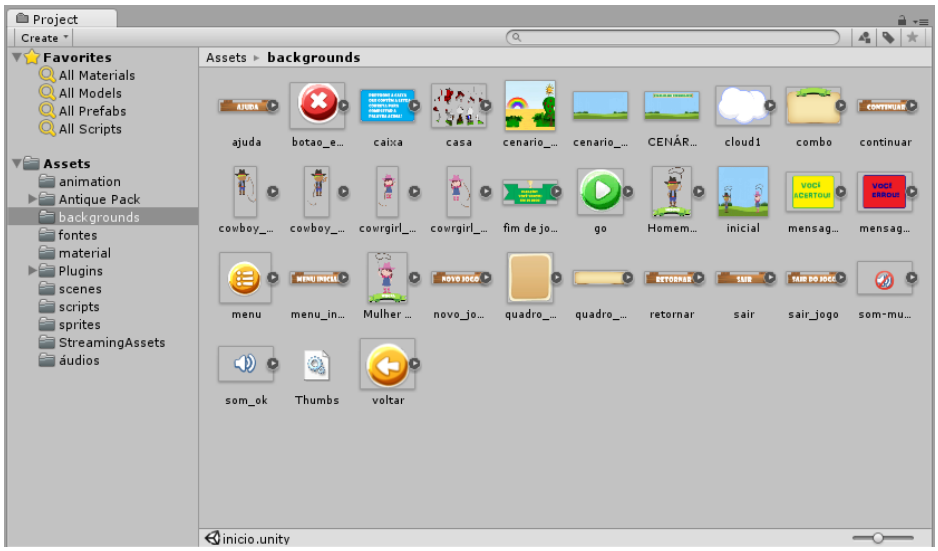
Para cada animação são gerados dois arquivos: um em formato “.controller” e outro em formato “.anim”. As animações são utilizadas ao criar as cenas do jogo, que são referenciadas pelo arquivo “.controller”. Detalhes na criação das cenas serão explicados mais adiante. No projeto *Laça Palavras*, foi necessário criar cinco animações. Duas animações referentes aos pássaros que voam em algumas cenas do jogo, uma animação referente ao personagem principal o “cowboy”, uma animação referente à outra personagem principal a “cowgirl” e uma animação referente aos cavalos que aparecem correndo na cena principal do jogo.

4.3.2 Pasta backgrounds

A pasta *backgrounds* contém todas as imagens de fundo que fazem parte dos cenários das cenas, dos botões, das mensagens, da caixa de apresentação de palavras, dentre outros

elementos do jogo. Todas as imagens utilizadas estão no formato PNG, algumas foram criadas com o auxílio do CorelDRAW e outras foram retiradas em sites da internet que disponibilizam imagens livres para uso. A pasta *backgrounds* é mostrada na imagem abaixo.

Figura 4.3: Pasta *backgrounds* do projeto *Laça Palavras*.

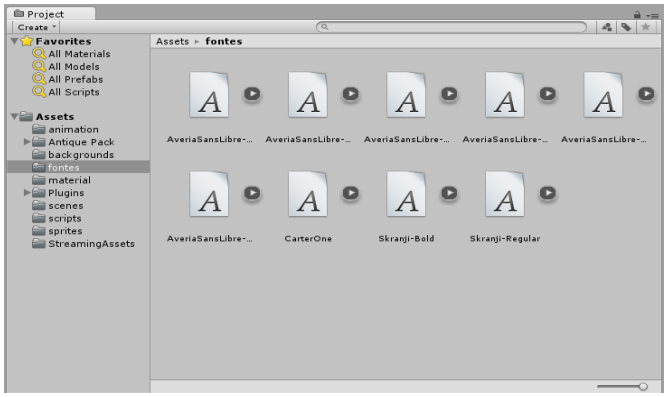


Fonte: Elaborada pelo autor.

4.3.3 Pasta fontes

A pasta fontes contém arquivos de diferentes tipos ou modelos de Fontes (letras) que podem ser utilizados nos elementos do jogo que contém algum texto. O Unity 3D, por padrão, contém apenas o modelo de fonte Arial. Os arquivos de fontes que podem ser utilizados são arquivos em formato TrueType Fonts “.ttf” ou OpenType Fonts “.otf”, basta apenas colocar os arquivos em uma subpasta no diretório raiz Assets que o Unity 3D já as encontra automaticamente. A pasta fontes é mostrada na figura 4.4.

Figura 4.4: Pasta fontes do projeto *Laça Palavras*.

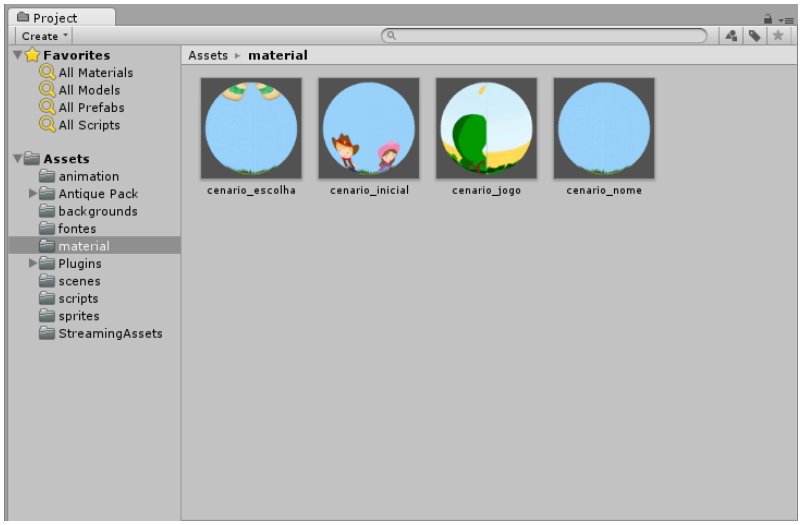


Fonte: Elaborada pelo autor.

4.3.4 Pasta material

A pasta material contém todos *Materials* utilizados pelo jogo. *Materials* são arquivos em formato “.mat” que representam elementos gráficos compostos por texturas e *shaders*: a textura representa o que é desenhado na superfície do material que pode ser uma imagem em formato PNG, enquanto que os *shaders* representam como será desenhado, ou seja, o formato. No projeto *Laça Palavras*, os *Materials* foram utilizados para desenvolver os cenários de fundo das cenas. O interessante de se utilizar os *Materials* é que os mesmos podem ser animados. A pasta material é mostrada na figura 4.5.

Figura 4.5: Pasta material do projeto *Laça Palavras*.

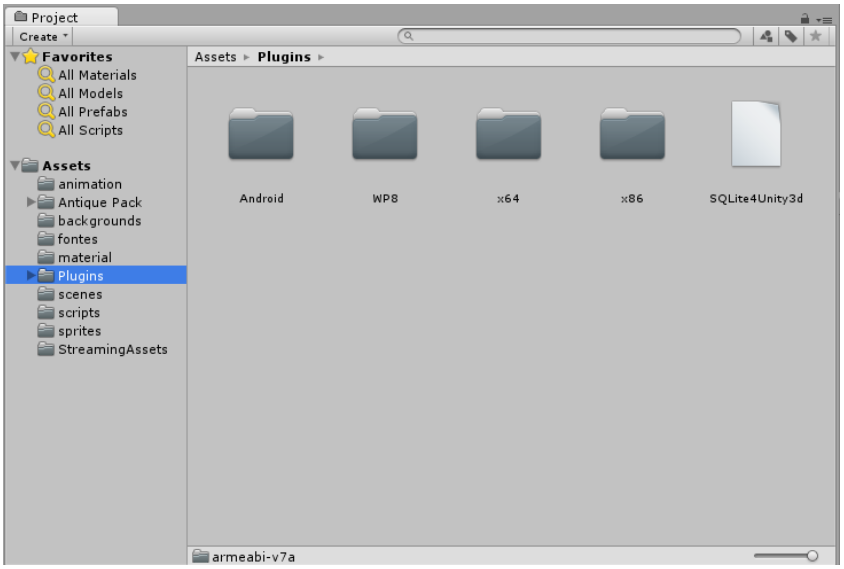


Fonte: Elaborada pelo autor.

4.3.5 Pasta Plugins

A pasta Plugins contém os arquivos de bibliotecas externas ao Unity 3D, ou seja, bibliotecas que implementam funcionalidades e recursos, mas que não existem no Unity 3D inicialmente como padrão. O Unity 3D possui um recurso chamado *Asset Store* que é uma espécie de loja que contém vários plug-ins de novas funcionalidades e recursos que podem ser comprados ou baixados gratuitamente para serem incluídos em seus projetos. Um *plugin* externo que foi utilizado no projeto *Laça Palavras* foi o SQLite. Como foi explicado no capítulo 3, o SQLite é uma biblioteca que implementa um sistema de gerenciamento de banco de dados. A pasta Plugins é mostrada na figura 4.6.

Figura 4.6: Pasta Plugins do projeto *Laça Palavras*.

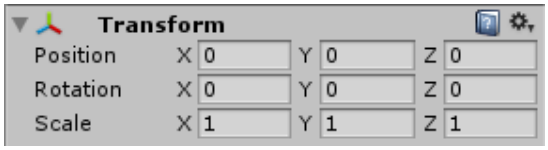


Fonte: Elaborada pelo autor.

4.3.6 Pasta scenes

A pasta *scenes* contém todas as cenas do jogo, é nas cenas que são adicionados e programados todos os recursos através dos *Game Objects*. Os jogos criados no Unity 3D são desenvolvidos através de cenas e cada cena é uma parte do jogo que possui alguma funcionalidade ou representa alguma fase. Qualquer elemento em uma cena, seja algum botão, a imagem de fundo ou o personagem principal animado, é tratado como um *Game Object*. Todos os *Game Objects* são posicionados dentro da cena através de um sistema de coordenadas, seja em 2 ou 3 dimensões e são as unidades fundamentais dentro de qualquer cena. Todos os objetos do jogo podem se movimentar dentro da cena, através da alteração de suas propriedades *Transform*. Cada objeto dentro do jogo no Unity possui um *Transform*, que por sua vez contém as coordenadas para a posição, rotação e escala do objeto. Essas propriedades são alteradas através dos scripts de programação. A figura 4.7 mostra um exemplo das propriedades *Transform* na interface do Unity.

Figura 4.7: Imagem exemplificando o *Transform* na interface do Unity 3D**.**

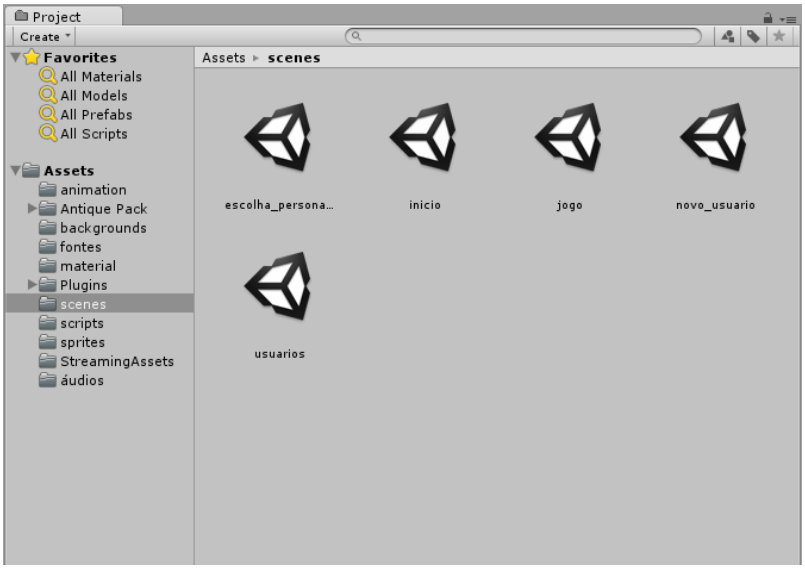


Fonte: Elaborada pelo autor.

O projeto *Laça Palavras* possui um total de cinco cenas onde cada uma possui um propósito diferente, ao serem desenvolvidas são salvas como arquivos do tipo “.unity”. São

elas: inicio; novo_usuario; usuarios; escolha_personagem; e jogo. Cada uma delas serão explicadas mais adiante. A pasta scenes é mostrada na figura 4.8.

Figura 4.8: Pasta *scenes* do projeto *Laça Palavras*.

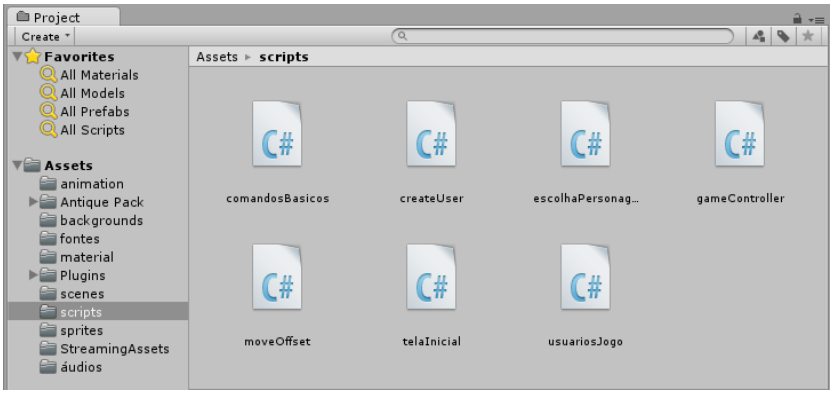


Fonte: Elaborada pelo autor.

4.3.7 Pasta scripts

A pasta *scripts* contém todos os *scripts* de programação em linguagem C#. Os *scripts* são os arquivos onde estão definidas todas as classes com suas respectivas propriedades e métodos que programam as funcionalidades de cada elemento em cada cena do jogo. Os *scripts* são arquivos em formato ".cs" e cada *script* pode ter mais de uma classe implementada. A pasta *scripts* é mostrada na figura 4.9.

Figura 4.9: Pasta scripts do projeto *Laça Palavras*.

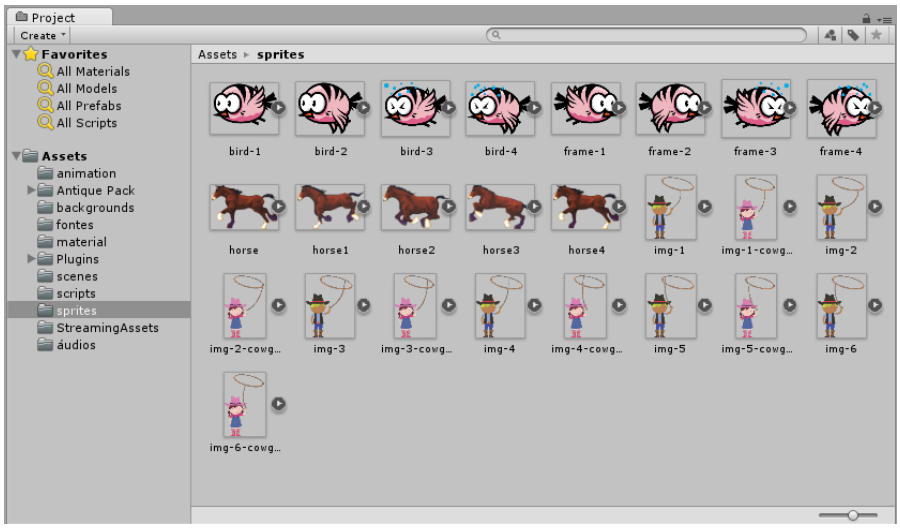


Fonte: Elaborada pelo autor.

4.3.8 Pasta sprites

A pasta *sprites* contém todas as imagens que fazem parte das animações criadas para o jogo. Essas imagens são construídas em uma determinada ordem que ao serem colocadas em transição uma com as outras seguindo uma sequência produz uma sensação de movimento, de animação. É importante deixar as imagens de animações em pasta diferente das outras imagens (imagens de cenários, de botões) para mantê-las de forma mais organizada. A pasta *sprites* é mostrada na figura 4.10.

Figura 4.10: Pasta *sprites* do projeto *Laça Palavras*.

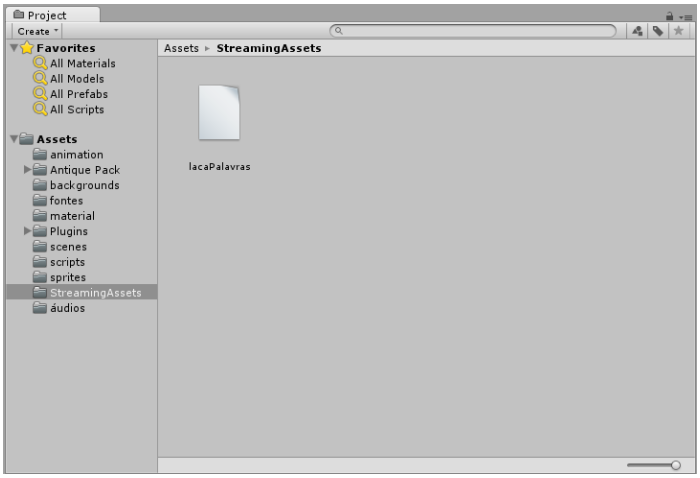


Fonte: Elaborada pelo autor.

4.3.9 Pasta StreamingAssets

A pasta *StreamingAssets* contém o arquivo de banco de dados do projeto. Como foi explicado no capítulo 3, o projeto *Laça Palavras* utiliza o SQLite como gerenciador de banco de dados e este é um SGBD que faz o gerenciamento dos dados diretamente em arquivos. Assim para utilizar um banco de dados já implementado é necessário deixar o arquivo na pasta *StreamingAssets*. No projeto *Laça Palavras* o banco de dados é utilizado para salvar as palavras a serem respondidas, dados do usuário, como pontuação, palavras que já acertou, nível e dados relacionados à inteligência artificial. A figura 4.11 mostra a pasta StreamingAssets.

Figura 4.11: Pasta *StreamingAssets* do projeto *Laça Palavras*.

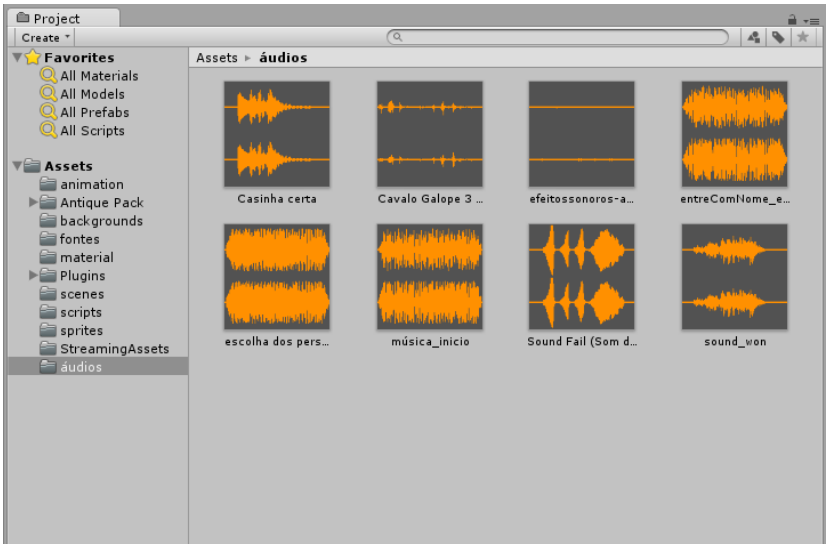


Fonte: Elaborada pelo autor.

4.3.10 Pasta áudios

A pasta áudios contém todos os arquivos de áudio do projeto. Geralmente, os jogos reproduzem áudios durante suas execuções, seja na tela inicial, durante o jogo ou ao realizar alguma ação. No Unity 3D, os áudios podem ser colocados como componentes nos *Game Objects* e suas execuções são controladas através de suas referências nos scripts de programação. O *Laça Palavras* possui áudios em todas as cenas e o formato dos arquivos é ".mp3". A pasta áudios é mostrada na figura 4.12.

Figura 4.12: Pasta áudios do projeto *Laça Palavras*.



Fonte: Elaborada pelo autor.

4.4 Funcionamento do Jogo

Nesta seção será explicado com mais detalhes como é o funcionamento do jogo. Como já foi mencionado anteriormente, o *Laça Palavras* é composto por cinco cenas, onde cada uma representa uma parte do jogo, assim, será explicado o funcionamento de cada cena com sua respectiva implementação.

4.4.1 Tela Inicial

A tela inicial do jogo consiste na primeira interface que é apresentada ao usuário assim que o jogo é executado. Nela contém o nome do jogo com um desenho de fundo e quatro botões. A tela inicial é mostrada na figura 4.13.

Figura 4.13: Interface inicial do *Laça Palavras*

Fonte: Elaborada pelo autor.

Como pode ser observado na figura 4.13, existem os botões “novo jogo”, “continuar” e “sair”. Com exceção do botão “sair” que encerra a execução do jogo. Cada botão leva o usuário para uma outra tela. Também existem algumas nuvens e pássaros que se movimentam no fundo da tela e um botão no canto inferior esquerdo, similar a um alto-falante, que desativa e ativa o áudio. Todos esses elementos gráficos contidos na tela são inseridos através de comandos manipulados pelo próprio mouse na interface do Unity, o usuário cria os elementos e vincula imagens a esses elementos.

O comportamento desses elementos é programado pelos *scripts*. Os *scripts* “telaInicial.cs” e “comandosBasicos.cs” são vinculados à tela inicial e fornecem a programação das funcionalidades. O *script* “comandosBasicos.cs” também é vinculado a todas as outras quatro cenas e implementa classes e funções que são utilizadas por todo o jogo. Alguns conceitos de implementação do Unity serão explicados através da apresentação

do funcionamento da tela inicial e tais conceitos também foram utilizados para a implementação das demais telas.

A classe que controla a tela inicial é denominada “telaInicial” e está implementada no arquivo “telaInicial.cs”. Ela herda propriedades e métodos da classe padrão do Unity chamada *MonoBehaviour* que é a base para a implementação de cenas do jogo. Toda classe que tem o papel de implementar as funções de uma cena no Unity herda da classe *MonoBehaviour*, que possui como padrão as funções “Start()” e “FixedUpdate()” em sua interface para serem implementadas. Quando o jogo executa determinada cena é instanciado um objeto da classe vinculada a tal cena e a função “Start()” é chamada, ou seja, a função “Start()” é o construtor da classe. Já a função “FixedUpdate()” é chamada a cada frame por segundo durante a execução da cena, ou seja, várias vezes por segundo como um *loop*. As propriedades da classe são variáveis que referenciam os *Game Objects* (elementos da cena) e variáveis que são utilizadas para auxiliar na implementação da lógica que controla a cena. A figura 4.14 apresenta a implementação da função “Start()” da tela inicial.

Figura 4.14: Implementação da função “Start()” da tela inicial.

```
// Inicialização da tela início!
void Start () {

    // inicializando as posições das núvens e do pássaro!
    xCloud1 = cloud1.transform.position.x;
    yCloud1 = cloud1.transform.position.y;

    xCloud2 = cloud2.transform.position.x;
    yCloud2 = cloud2.transform.position.y;

    xBird = bird.transform.position.x;
    yBird = bird.transform.position.y;



}
```

Fonte: Elaborada pelo autor.

A função “Start()” da tela inicial mostra, como exemplo de utilização, a inicialização das variáveis relacionadas a posição inicial das nuvens e pássaros da tela ao executar a cena.

Figura 4.15: Implementação da função “FixedUpdate()” da tela inicial.

```
// 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);
}
```

Como foi mencionado anteriormente, a função “FixedUpdate()” é chamada a cada frame por segundo durante a execução da cena. Na tela inicial a função “FixedUpdate()” controla os movimentos de alguns elementos da tela como as nuvens que ficam ao fundo e os pássaros.

As funcionalidades dos botões que aparecem na tela inicial também foram programadas, porém, como a função de cada botão é levar o usuário a uma outra tela ou, no caso do botão “sair”, é encerrar o jogo, essas funções foram implementadas no *script* “comandosBasicos.cs”. Isso se deu pelo fato de tais funções serem aplicáveis a todo o jogo e não somente à tela inicial. Para sair de uma cena e carregar uma nova cena (como é o caso ao pressionar os botões “novo jogo” e “continuar”) é chamada a função “loadScene()” que é mostrada na figura 4.16. Ao pressionar o botão “sair” a execução é executada a função “sair()”, mostrada na figura 4.17.

Figura 4.16: Implementação da função para carregar uma nova cena.

```
// Carregar cena!
public void loadScene(string nameScene)
{

    Application.LoadLevel(nameScene);

}
```

Fonte: Elaborada pelo autor.

Figura 4.17: Implementação da função para encerrar o jogo.

```
public void sair()
{
    Application.Quit();

}
```

Fonte: Elaborada pelo autor.

O botão “novo jogo” leva o usuário para a cena “novo_usuario”, que será explicada na próxima seção. A mesma tem como objetivo criação de um novo usuário. O botão “continuar” leva o usuário para a cena “usuarios” (que também será explicada mais adiante) onde a mesma apresenta todos os usuários já criados e a possibilidade selecionar algum para continuar o jogo salvo. E por último o botão “sair” que encerra a execução do jogo.

4.4.2 Tela de criação de novo usuário

A tela de criação de novo usuário é apresentada quando é pressionado o botão “novo jogo” da tela inicial. O *Laça Palavras* foi desenvolvido com a possibilidade de criação de vários usuários em que cada um pode jogar e salvar seu próprio jogo. A tela contém um campo de texto onde o usuário deve digitar o seu nome. Também existem dois botões em que um deles (o do lado direito) tem como objetivo salvar o usuário para iniciar o jogo e o outro (o do lado esquerdo) para voltar para tela inicial. A tela de criação de novo usuário é mostrada na figura 4.18.

Figura 4.18: Tela de criação de novo usuário.

Fonte: Elaborada pelo autor.

Para fazer o controle da tela foi criado o *script* “createUser.cs” em que foi implementada a classe “createUser”. Esta tem como função principal o método “create()” que é executado quando o usuário pressiona o botão de prosseguir (o botão ao lado direito com

uma seta apontando para direita). O método “create()” salva o novo usuário no banco de dados. A figura 4.19 apresenta a implementação do método “create()”.

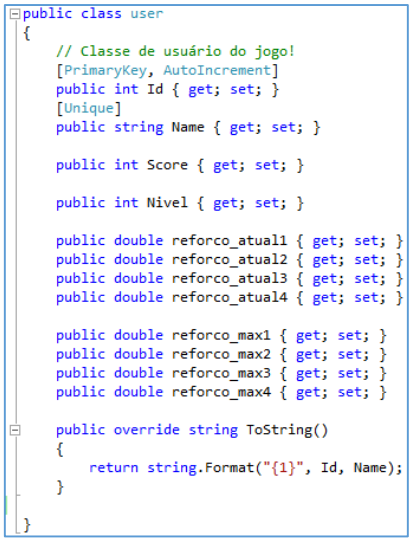
Figura 4.19: Implementação do método “create()”.

```
// Função para criar usuário!
public void create()
{
    // Conexão com o banco de dados lacapalavras!
    DataService data = new DataService("lacapalavras");

    if (nome.text == "") return;

    var users = data._connection.Table<user>().Where(x => x.Name == nome.text);

    if (users.Count() >= 1) return;

    data.CreateUser(nome.text, 0,0);

    users = null;

    users = data._connection.Table<user>().Where(x => x.Name == nome.text);

    // Pega o usuário que ta sendo criado e seta como usuário que ta jogando!
    foreach (var user in users)
    {
        dadosJogo.Instance.currentUser = user;
    }

    Application.LoadLevel("escolha_personagem");

}
```

Fonte: Elaborada pelo autor.

Para realizar a conexão com o banco de dados é criada uma variável do tipo “DataService”. A classe “DataService” foi implementada para realizar a conexão com o banco de dados SQLite. Após a conexão, o método verifica se o campo de texto da tela está vazio ou se já existe algum usuário salvo com o mesmo nome digitado, caso o nome digitado seja válido o usuário é criado e deixa-o como usuário ativo. A classe de persistência que modela o usuário é chamada de “user” e foi implementada no script “usuariosJogo.cs”. A figura 4.20 mostra a classe de persistência “user”.

Figura 4.20: Classe de persistência “user”.



```
public class user
{
    // Classe de usuário do jogo!
    [PrimaryKey, AutoIncrement]
    public int Id { get; set; }
    [Unique]
    public string Name { get; set; }

    public int Score { get; set; }

    public int Nivel { get; set; }

    public double reforco_atual1 { get; set; }
    public double reforco_atual2 { get; set; }
    public double reforco_atual3 { get; set; }
    public double reforco_atual4 { get; set; }

    public double reforco_max1 { get; set; }
    public double reforco_max2 { get; set; }
    public double reforco_max3 { get; set; }
    public double reforco_max4 { get; set; }

    public override string ToString()
    {
        return string.Format("{1}", Id, Name);
    }

}
```

Fonte: Elaborada pelo autor.

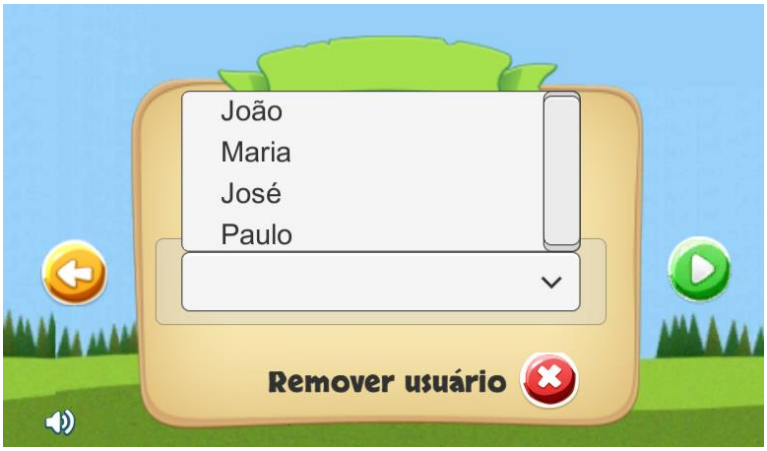
Pode-se observar na figura 4.20 que os dados referentes ao usuário englobam um identificador do tipo inteiro, o nome do usuário, sua pontuação e seu nível. Todas as outras variáveis que começam com “reforco_” são variáveis relacionadas com a inteligência artificial que será explicado mais adiante.

Após criar o usuário e salvá-lo no banco de dados o usuário é redirecionado para a tela de escolha do personagem.

4.4.3 Tela usuários

A tela de usuários tem como objetivo apresentar os usuários que já estão cadastrados no jogo e a possibilidade de continuar a jogar um jogo salvo. É acessada ao pressionar o botão “continuar” da tela inicial. O usuário pode ser selecionado através de um *dropdown* (uma espécie de lista que se expande ao ser clicada ou tocada) e existem as opções de prosseguir com jogo, voltar para tela inicial ou excluir algum usuário existente. A tela de usuários é mostrada na figura 4.21.

Figura 4.21: Tela de usuários.



Fonte: Elaborada pelo autor.

Como pode-se observar na figura 4.21 acima, x o *dropdown* mostra todos os usuários que estão salvos no jogo. Ao pressionar botão do lado esquerdo da tela que está com uma seta para o lado esquerdo, o jogo voltara para tela inicial, o botão com um "X" que está na parte de baixo da tela, exclui o usuário que está selecionado no *dropdown* e o botão que está ao lado direito da tela com uma seta para a direita prossegue com o jogo referente ao usuário que está selecionado no *dropdown*.

A classe que controla a tela de usuários foi definida como "usuariosJogo" e está implementada no *script* "usuariosJogo.cs". A classe possui o método "Start()" que é chamado assim que a tela é carregada. A implementação do método "Start()" é mostrada na figura 4.22.

Figura 4.22: Implementação do método "Start() da classe "usuariosJogo".

```
// Inicialização da tela usuariosJogo!
void Start () {

    data = new DataService("lacaPalavras");

    IEnumerable<user> users = data._connection.Table<user>();

    int index = 0;

    // buscando todos os usuários que estão cadastrados e jogando no select!
    foreach (var user in users)
    {
        Dropdown.OptionData a = new Dropdown.OptionData();
        a.text = user.Name;

        Users.options.Insert(index, a);
        ++index;
    }

}
```

Fonte: Elaborada pelo autor.

O método “Start()” faz uma conexão com banco de dados e realiza uma consulta que retorna todos os usuários já cadastrados no jogo, em seguida preenche o *dropdown* com o resultado, gerando a lista de usuários na tela.

Outro método que foi implementado na classe “usuariosJogo” é o “removeUser()”, que é executado quando um usuário é selecionado no *dropdown* e o botão remover usuário é pressionado. O método tem como objetivo excluir o usuário permanentemente. A implementação do método “removeUser()” é mostrada na figura 4.23.

Figura 4.23: Implementação do método “removeUser()”.

```
// função para remover usuários!
public void removeUser()
{


    var users = data._connection.Table<user>().Where(x => x.Name == Users.captionText.text);


    foreach (var user in users)
    {

        data.removeUser(user.Id);

    }
        Users.options.RemoveAt(Users.value);

        Users.captionText.text = "";


}
```

Fonte: Elaborada pelo autor.

O método “removerUser()” primeiramente realiza uma conexão com banco de dados e faz uma consulta buscando o usuário que foi selecionado no *dropdown*, em seguida exclui o usuário através do deu “Id”, e por fim exclui do *dropdown* o nome do usuário excluído.

Outro método que a classe “usuariosJogo” possui é o método “next()”, que é executado quando o botão de prosseguir (botão que está ao lado direito da tela com uma seta para a direita) é pressionado. O método tem como objetivo carregar todas as informações do usuário, que foi selecionado no *dropdown* e carregar a cena de escolha do personagem. A figura 4.24 apresenta a implementação do método “next()”.

Figura 4.24: Implementação do método “next()”.

```
// Função ir para próxima tela!
public void next()
{

    if (Users.captionText.text == "") return;
    else
    {
        var users = data._connection.Table<user>().Where(x => x.Name == Users.captionText.text);

        foreach(var user in users)
        {

            dadosJogo.Instance.currentUser = user;

        }
        dadosJogo.Instance.vetor_reforco_atual[0] = dadosJogo.Instance.currentUser.reforco_atual1;
        dadosJogo.Instance.vetor_reforco_atual[1] = dadosJogo.Instance.currentUser.reforco_atual2;
        dadosJogo.Instance.vetor_reforco_atual[2] = dadosJogo.Instance.currentUser.reforco_atual3;
        dadosJogo.Instance.vetor_reforco_atual[3] = dadosJogo.Instance.currentUser.reforco_atual4;
        dadosJogo.Instance.vetor_reforco_max[0] = dadosJogo.Instance.currentUser.reforco_max1;
        dadosJogo.Instance.vetor_reforco_max[1] = dadosJogo.Instance.currentUser.reforco_max2;
        dadosJogo.Instance.vetor_reforco_max[2] = dadosJogo.Instance.currentUser.reforco_max3;
        dadosJogo.Instance.vetor_reforco_max[3] = dadosJogo.Instance.currentUser.reforco_max4;


        Application.LoadLevel("escolha_personagem");

    }
}
```

Fonte: Elaborada pelo autor.

O método “next()” primeiramente verifica se algum usuário foi selecionado no *dropdown*. Caso nenhum usuário seja selecionado, o método retorna “vazio” e o jogo não faz nada. Mas, se um usuário for selecionado no *dropdown*, ele faz uma consulta no banco de dados buscando os dados do usuário selecionado e coloca-o como usuário ativo no jogo. Por fim, o método inicializa todas as variáveis referentes à inteligência artificial de acordo com os dados do usuário que foi ativado para serem devidamente utilizadas durante a execução do jogo.

4.4.4 Tela de escolha do personagem

A tela de escolha do personagem é a próxima tela apresentada após a seleção ou a criação de um usuário. A tela possui dois personagens que podem ser escolhidos, “menino” que é o *cowboy* e a “menina” que é a *cowgirl*. Também possui um botão (botão redondo à esquerda da tela com uma seta para a esquerda) que tem a função de retornar para a tela de usuários. A tela é mostrada na figura 4.25.

Figura 4.25: Tela de escolha do personagem.

Fonte: Elaborada pelo autor.

A classe que programa as funções da tela é chamada de “escolhaPersonagem” e foi implementada no *script* “escolhaPersonagem.cs”. A classe possui dois métodos principais, que implementam a escolha de cada personagem. Ao selecionar o personagem “menino” é executado o método “loadSceneBoy()” que basicamente define um valor a uma variável para o jogo “entender” que deve ativar o *cowboy* e leva o usuário para a cena principal do jogo.

Ao selecionar a personagem “menina” é executado o método “loadSceneGirl()”, que também foi implementado de maneira similar ao método “loadSceneBoy()”, porém, define a variável com um valor diferente para o jogo “entender” que deve ativar a *cowgirl*. A implementação do método “loadSceneGirl()” e “loadSceneBoy()” é mostrada na figura 4.26.

Figura 4.26: Implementação dos métodos “loadSceneBoy()” e “loadSceneGirl()” .

```
// Carregar a cena quando selecionar o cowboy!
public void loadSceneBoy(string nameScene)
{

    dadosJogo.Instance.currentPesonagem = 2;

    Application.LoadLevel(nameScene);


}
```

```
// Carregar a cena quando selecionar a cowgirl!
public void loadSceneGirl(string nameScene)
{

    dadosJogo.Instance.currentPesonagem = 1;
    Application.LoadLevel(nameScene);

}
```

Fonte: Elaborada pelo autor.

Percebe-se que, de acordo com a figura 4.26, ao selecionar o personagem “menino” a variável “dadosJogo.Instance.currentPersonagem” recebe o valor numérico 2, enquanto, se a personagem “menina” for selecionada a variável “dadosJogo.Instance.currentPersonagem” recebe o valor numérico 1. Assim, ao carregar a cena principal do jogo é verificada a condição que é mostrada na figura 4.27.

Figura 4.27: Condição que verifica qual personagem deve ser ativado.

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

Fonte: Elaborada pelo autor.

Como pode-se perceber na figura 4.27, se o valor da variável “dadosJogo.Instance.currentPersonagem” for igual a 1 o *cowboy* é desativado, caso “dadosJogo.Instance.currentPersonagem” for diferente de 1 a *cowgirl* é desativada. A condição foi implementada dessa forma pelo fato do Unity deixar os elementos da cena ativados, assim o personagem que for selecionado continua ativo e o outro é desativado.

4.4.5 Tela principal do jogo

A tela principal do jogo consiste na cena apresentada ao usuário durante a partida, ou seja, quando as palavras são apresentadas com as opções para serem respondidas. A tela consiste em um cenário de natureza como uma espécie de fazenda onde o personagem principal deve laçar a letra para completar a palavra corretamente que aparece no quadro, daí o nome *Laça Palavras*. A tela principal do jogo é mostrada na figura 4.28.

Figura 4.28: Tela principal do jogo *Laça Palavras*.

Fonte: Elaborada pelo autor.

Como pode-se observar na imagem acima, o cenário contém nuvens, pássaros, sol, árvore, dentre outros elementos. Um cenário de natureza bastante chamativo com o intuito de atrair as crianças em idade de alfabetização. Na parte superior ao centro, está o quadro onde as palavras são apresentadas e um botão (com um desenho de nota musical) que, ao ser pressionado, executa um áudio que consiste em uma voz pronunciando uma frase contendo a palavra apresentada. Para cada palavra a ser completada existe um áudio de um menino ou uma menina (dependendo do personagem principal) pronunciando uma frase que contém a palavra a ser completada em um determinado contexto. Esse recurso é importante porque a criança tem a possibilidade de perceber que a pronuncia da palavra não ajuda na escolha da letra, sendo necessário memorizar sua escrita. É um diferencial do *Laça Palavras* em relação a outros jogos digitais ortográficos. Na parte inferior contém os cavalos correndo com as caixinhas com as opções que podem ser escolhidas através do toque na tela para completar a palavra. Em todas as palavras são apresentadas duas opções. Essas caixinhas são deixadas pelos cavalos no meio da tela onde são laçadas pelo *cowboy* ou *cowgirl* para completar a palavra no quadro. A figura 4.29 mostra os cavalos derrubando as caixinhas no centro do cenário.

Figura 4.29: Tela principal com os cavalos deixando as caixinhas com as letras.

Fonte: Elaborada pelo autor.

O *cowboy* ou a *cowgirl* fica no meio da tela com uma corda para laçar a caixinha que é selecionada. A figura 4.30 mostra o cowboy laçando a caixinha.

Figura 4.30: Cowboy laçando a caixinha com a letra selecionada.

Fonte: Elaborada pelo autor.

Na parte superior à esquerda está o número de acertos que vai sendo incrementado à medida que o usuário vai acertando as palavras. Já ao lado direito também na parte superior próximo ao sol, contém um botão redondo que ao ser pressionado o jogo é pausado e aparece um menu de opções que o usuário pode selecionar. Como é mostrado na figura 4.31.

Figura 4.31: Tela principal com o menu.

Fonte: Elaborada pelo autor.

O menu possui três botões onde cada um deles faz uma função diferente, o botão “Retornar” fecha o menu e retorna ao jogo, o botão “Menu Inicial” salva o jogo e leva o usuário para a tela inicial e o botão “Sair do Jogo” também salva o jogo e encerra a execução do mesmo.

O jogo possui quatro níveis de dificuldade, em que cada nível possui classes de palavras da língua portuguesa que não possuem regras, ocorrendo o caso da concorrência, em que os sons não contribuem para a escolha da letra ou dígrafo. São quatro classes de palavras, dentre elas, palavras com “S” ou “Z”, mas que possui som de [z] (exemplo a palavra “casa”), palavras com “SS” ou Ç, mas que possui som de [s] (exemplo a palavra “passo”), palavras com “L” ou “U”, mas que possui som de [u] (exemplo a palavra “alça”) e palavras com “G” ou “J”, mas que possui som [g] (exemplo a palavra “majestade”). A ortografia desses grupos de palavras gera muita confusão em crianças durante a fase de alfabetização ou recém-alfabetizadas e precisam ser treinadas para serem memorizadas.

O usuário inicia o jogo no primeiro nível e, de acordo com o seu desempenho, o nível de dificuldade vai sendo ajustado dinamicamente, através da execução de um algoritmo de aprendizado de máquina. Assim, se o usuário durante o jogo está desempenhando um bom número de acertos o jogo tende a aumentar o nível de dificuldade dinamicamente. Da mesma forma, se o usuário estiver errando, o jogo tende a manter ou diminuir o nível dinamicamente. Detalhes sobre a implementação do ajuste dinâmico de dificuldade será explicado mais adiante, na seção 4.5.

No jogo, o usuário tem como objetivo acertar todas as palavras para realizar a construção de uma casinha ao lado esquerdo do cenário. A casinha vai se formando como uma espécie de quebra-cabeça à medida em que o usuário vai acertando as palavras. Isso desperta maior interesse no usuário em continuar o jogo para ver a casinha completa no final. A figura 4.32 mostra como exemplo a casinha se formando no cenário do jogo à medida que o usuário vai acertando as palavras.

Figura 4.32: Construção da casinha no cenário do jogo.

Fonte: Elaborada pelo autor.

A classe que controla as funções da tela principal do jogo é chamada de "gameController" e foi implementada no *script* "gameController.cs". Todos os elementos presentes na tela são referenciados como atributos da classe "gameController", ou seja, são os *Game Objects* que possuem atributos e métodos que podem ser manipulados para programar seus comportamentos. Quando a execução da tela é iniciada o método "Start()" é chamado e tem como objetivo inicializar os elementos da tela de acordo com os dados do usuário. Na figura 4.33 é mostrada as instruções principais que são executadas no método "Start()".

Figura 4.33: Implementação do método "Start()" da tela principal do jogo.

```
ScoreInitial = dadosJogo.Instance.currentUser.Score; // Recebendo do banco de dados o score do usuário corrente!

if (ScoreInitial == 0) mensagem_inicial.SetActive(true);
else verifica_porcentagem_casa();

Score.text = "ACERTOS: " + dadosJogo.Instance.currentUser.Score;

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

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

countWords = (bancoPalavras.Instance.qtd_Words);

idPalavra = Random.Range(1, 9); // Random para colocar palavra inicial aleatória

randNum_blocos = Random.Range(1.0f, 2.0f); // Ordem aleatória de entrada das letras!

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

Fonte: Elaborada pelo autor.

Explicando de maneira simples, o método primeiramente recebe do banco de dados a pontuação atual do usuário corrente e verifica se o usuário está iniciando o jogo pela primeira vez ou já possui um jogo salvo. Em seguida, caso o usuário já tenha um jogo salvo e possui

um número de acertos acima de zero, é verificada a porcentagem da construção da casinha no cenário. Depois o número de acertos do personagem é carregado na tela, é verificado o personagem que foi escolhido, é carregado do banco de dados as palavras do nível atual do usuário e por fim é selecionada a primeira palavra do nível de maneira aleatória e apresentada no quadro para ser respondida.

Durante a execução da tela o método “FixedUpdate()” é chamado a cada frame por segundo. Esse método tem como objetivo fazer as alterações nos elementos da tela e o controle dos mesmos à medida que o jogo vai executando. Como por exemplo, realizar o movimento das nuvens, dos pássaros, dos cavalos correndo com as caixinhas, do personagem principal, verificar se o usuário acertou ou errou a palavra ao responder, selecionar a próxima palavra a ser respondida, modificar o número de acertos na tela, executar o algoritmo de inteligência, mudar o nível do jogo, acrescentar partes na casinha, executar áudios de acordo com determinados acontecimentos, dentre outros.

Como já foi mencionado anteriormente, *Laça Palavras* possui um algoritmo de inteligência artificial que tem como objetivo realizar o ajuste dinâmico de dificuldade de acordo com o desempenho do usuário ao completar as palavras. Esse algoritmo é executado a cada vez que o usuário completa dez palavras de um mesmo nível em sequência ou responde as palavras restantes do nível, a execução desse algoritmo também é controlada pelo método “FixedUpdate()”. Assim, o método “FixedUpdate()” é o método principal da tela que controla toda a execução do jogo. Como a função “FixedUpdate()” em linguagem C# ficou muito extensa, seu pseudocódigo é apresentado no Algoritmo 1.

Algoritmo 1: Algoritmo da tela principal do *Laça Palavras*.

```
Enquanto durar jogo faça
      Controle do movimento das nuvens;
      Controle do movimento e do áudio dos pássaros;
      Controle do movimento e do áudio dos cavalos;
      Controle do movimento das caixinhas com as letras;
    Se o usuário escolheu uma letra
        Cowboy laça a caixa escolhida;
       Se acertou
         Registra o acerto da palavra;
         Incrementa o número de acertos;
         Exibe mensagem de acerto ao usuário;
         Mostra a palavra completa no quadro;
         De acordo com o número de acertos aparece uma parte da casinha;
       Se errou
          Exibe mensagem de erro ao usuário;
          Mostra a palavra completa no quadro;
```

```
            Incrementa o número de erros;
        Fim se errou
        Se acabou de responder 10 palavras
          Executa o algoritmo de inteligência artificial;
          Salva os dados;
          Seleciona o nível;
        Fim se acabou de responder 10 palavras
        Apresenta a próxima palavra no quadro para ser respondida;
    Fim se usuário escolheu uma letra
Fim enquanto
```

A figura 4.34 apresenta o código implementado em linguagem C# que controla os movimentos das nuvens na tela.

Figura 4.34: Controle do movimento das nuvens.

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

x1 = x1 + 0.002f;
x2 = x2 + 0.002f;
x3 = x3 + 0.002f;

cloud1.transform.position = new Vector2(x1, y1);
cloud2.transform.position = new Vector2(x2, y2);
cloud3.transform.position = new Vector2(x3, y3);
```

Fonte: Elaborada pelo autor.

O código apresentado na figura está dentro do método “FixedUpdate()”. Como já foi dito anteriormente, o método “FixedUpdate()” é executado a cada frame por segundo como um *loop*. O que o trecho de código faz é verificar se as posições das nuvens estão no limite da tela voltá-las para a mesma, incrementar suas posições no sentido horizontal utilizando a dimensão “x” do sistema de coordenadas para realizar seu movimento da direita para esquerda, e realizar as alterações nas suas propriedades de posição *Transform*.

A figura 4.35 apresenta o código implementado em linguagem C# que controla o movimento dos pássaros na tela.

Figura 4.35: Controle do movimento dos pássaros.

```
if (xBird1 <= -9.52f) xBird1 = 16.52f;
if (xBird2 <= -9.52f) xBird2 = 23.52f;

xBird1 -= 0.009f;
xBird2 -= 0.009f;

bird1.transform.position = new Vector2(xBird1, yBird1);
bird2.transform.position = new Vector2(xBird2, yBird2);
```

Fonte: Elaborada pelo autor.

De maneira similar ao código da figura 4.34, o trecho de código apresentado na figura 4.35 também é executado a cada frame por segundo. O que o trecho de código faz é, verificar se as posições dos dois pássaros estão no limite da tela voltá-los para a mesma, decrementar suas posições no sentido horizontal utilizando a dimensão “x” do sistema de coordenadas para realizar seu movimento da esquerda para direita, e realizar as alterações nas suas propriedades de posição *Transform*.

A figura 4.36 apresenta o código implementado em linguagem C# que controla o movimento e o áudio dos cavalos que correm na tela.

Figura 4.36: Controle do movimento e do áudio dos cavalos.

```
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();

xHorse -= 0.065f;
xHorse2 -= 0.065f;

horse.transform.position = new Vector2(xHorse, yHorse);
horse2.transform.position = new Vector2(xHorse2, yHorse2);
```

Fonte: Elaborada pelo autor.

O que o trecho de código apresentado na figura 4.36 faz é verificar se as posições dos dois cavalos estão no limite da tela para pausar os áudios do galope e depois voltá-los para a tela novamente e ativar os áudios, decrementar suas posições no sentido horizontal utilizando a dimensão “x” do sistema de coordenadas para realizar seu movimento da esquerda para direita (galopando), e realizar as alterações nas suas propriedades de posição *Transform*.

A figura 4.37 apresenta o código implementado em linguagem C# que controla o movimento das caixinhas com as letras.

Figura 4.37: Controle do movimento das caixinhas com as letras.

```
if (xboard_letter >= -2f) xboard_letter -= 0.065f;
else if (yboard_letter <= -1.5f)
{
    yboard_letter += 0.090f;
    yHorse -= 0.06f;

}

if (xboard2_letter >= 0f) xboard2_letter -= 0.065f;
else if (yboard2_letter <= -1.5f)
{
    yboard2_letter += 0.090f;
    yHorse2 -= 0.06f;

}

board_letter.transform.position = new Vector2(xboard_letter, yboard_letter);
board2_letter.transform.position = new Vector2(xboard2_letter, yboard2_letter);
```

Fonte: Elaborada pelo autor.

O objetivo do código apresentado na figura 4.37 é realizar o movimento das caixinhas com as letras (opções) da direita para esquerda juntamente com os cavalos até serem fixadas no meio da tela, onde o usuário escolhe a caixinha desejada para completar a palavra no quadro. O que o código faz basicamente é, verificar se a posição das caixinhas está no meio da tela a cada frame por segundo (de acordo com suas propriedades de posição “x” e “y”), realizar o movimento das caixinhas da direita para esquerda até que as mesmas cheguem no meio da tela decrementando suas propriedades de posição “x” (“xboard_letter” e “xboard2_letter”).

A figura 4.38 apresenta o código implementado em linguagem C# que faz o *cowboy* ou a *cowgirl* laçar a caixinha que foi selecionada pelo usuário ao escolher a letra para completar a palavra.

Figura 4.38: Código faz o *cowboy* ou a *cowgirl* laçar a caixinha.

```
if (dadosJogo.Instance.currentPesonagem == 2)
{
    cowboy_moving_rope.SetActive(false);
    cowboy_lacando1.SetActive(true);
}
else
{
    cowgirl_moving_rope.SetActive(false);
    cowgirl_lacando1.SetActive(true);
}
```

Fonte: Elaborada pelo autor.

O código da figura 4.38 é executado quando o usuário toca na caixinha selecionando a letra para completar a palavra. O que o código faz é, verificar qual é o personagem principal que está ativo (*cowboy* ou *cowgirl*) e desativar a animação do personagem rodando a corda e ativar a animação do personagem laçando a caixinha escolhida. Essas animações são criadas pela interface gráfica do Unity e são referenciadas no *script* como *Game Objects*.

A figura 4.39 apresenta o código implementado em linguagem C# que é executado quando o usuário seleciona a letra correta para completar a palavra.

Figura 4.39: Código executado quando o usuário seleciona a letra correta.

```
if (yboard2_letter >= -1.7f
    && txtBoardLetter2.text == bancoPalavras.Instance.palavras[dadosJogo.Instance.currentUser.Nivel][idPalavra].letra_correta)
{

    respondido = true;

    if (bancoPalavras.Instance.palavrasAcerto[auxIdpalavra].acerto != true)
    {
        bancoPalavras.Instance.palavrasAcerto[auxIdpalavra] = new palavraAcertoUser
        {

            idPalavra = bancoPalavras.Instance.palavras[dadosJogo.Instance.currentUser.Nivel][idPalavra].Id,
            idUser = dadosJogo.Instance.currentUser.Id,
            acerto = true,
            nivelPalavra = (dadosJogo.Instance.currentUser.Nivel+1)
        };

        ++dadosJogo.Instance.currentUser.Score;
        ++bancoPalavras.Instance.acertos;
        calcula_porcentagem_casa();
        Debug.Log("Acertos: " + bancoPalavras.Instance.acertos);
    }


    message_hit.SetActive(true);
    sound_won.GetComponent<AudioSource>().Play();
    txtBoardWord.text = bancoPalavras.Instance.palavras[dadosJogo.Instance.currentUser.Nivel][idPalavra].palavra_completa;

    Score.text = "ACERTOS: " + dadosJogo.Instance.currentUser.Score;
}
```

Fonte: Elaborada pelo autor.

O código da figura 4.39 realiza a sequência de passos que é descrito no Algoritmo 1, quando o usuário seleciona a letra correta que completa a palavra no quadro. A primeira estrutura de controle verifica se a letra que foi selecionada é a letra correta, em seguida verifica se o acerto da palavra já foi registrado e registra-o em um vetor, o acerto é registrado relacionando o “Id” da palavra com o “Id” do usuário para posteriormente ser salvo no banco de dados. Em seguida, a pontuação em incrementada, o número de acertos também é incrementado, é executada a função que verifica a construção ou não de uma nova parte da casinha no cenário, mostra na tela a mensagem de acerto ao usuário, executa um áudio que simboliza o acerto e por fim mostra a palavra completa no quadro da tela.

A figura 4.40 apresenta o código implementado em linguagem C# que é executado quando o usuário seleciona a letra errada para completar a palavra.

Figura 4.40: Código executado quando o usuário seleciona a letra errada.

```
respondido = true;
message_error.SetActive(true);

if (bancoPalavras.Instance.palavrasAcerto[auxIdpalavra].acerto != false)
{
    bancoPalavras.Instance.palavrasAcerto[auxIdpalavra] = new palavraAcertoUser
    {

        idPalavra = bancoPalavras.Instance.palavras[dadosJogo.Instance.currentUser.Nivel][idPalavra].Id,
        idUser = dadosJogo.Instance.currentUser.Id,
        acerto = false,
        nivelPalavra = (dadosJogo.Instance.currentUser.Nivel+1)
    };

}

sound_lost.GetComponent<AudioSource>().Play();
txtBoardWord.text = bancoPalavras.Instance.palavras[dadosJogo.Instance.currentUser.Nivel][idPalavra].palavra_completa;

++dadosJogo.Instance.erros[dadosJogo.Instance.currentUser.Nivel];
```

Fonte: Elaborada pelo autor.

O código da figura 4.40 realiza a sequência de passos que é descrito no Algoritmo 1 quando o usuário seleciona a letra errada para completar a palavra no quadro. Primeiramente o código executa o comando que mostra a mensagem de erro, em seguida verifica se o erro da palavra já foi registrado e registra-o em um vetor, o erro também é registrado relacionando o "Id" da palavra com o "Id" do usuário para posteriormente ser salvo no banco de dados. Em seguida, é executado um áudio que simboliza o erro, logo após a palavra correta é mostrada no quadro da tela e o número de erros é incrementado.

A figura 4.41 apresenta o código implementado em linguagem C# que é executado quando o usuário acaba de responder 10 palavras.

Figura 4.41: Código executado quando o usuário acaba de responder 10 palavras.

```
dadosJogo.Instance.ultimo_desempenho = inteligencia.Instance.calcula_desempenho();
inteligencia.Instance.QLearning();
dadosJogo.Instance.salvar_dados();
inteligencia.Instance.seleciona_nivel();
ScoreInitial = dadosJogo.Instance.currentUser.Score;
bancoPalavras.Instance.carrega_palavras_nivel(dadosJogo.Instance.currentUser.Nivel);
```

Fonte: Elaborada pelo autor.

A cada 10 palavras que o usuário responde é executada algumas funções referentes à inteligência artificial do jogo, como mostra o código da figura 4.41. Primeiramente é executada a função que calcula o desempenho do usuário nas últimas 10 palavras respondidas, em seguida é executada a função Q-Learning (que será melhor explicada na seção 4.5), os dados referentes ao usuário são salvos no banco de dados, o nível das próximas palavras é selecionado e suas palavras são buscadas no banco de dados e carregadas em memória primária (em um vetor).

4.5 Algoritmo de Inteligência Artificial

Nesta seção será explicado com mais detalhes como foi o desenvolvimento da inteligência artificial do jogo *Laça Palavras*.

4.5.1 Aprendizado de Máquina

Como já foi explicado no capítulo 2, autores da área de Inteligência Artificial (IA) conceituam o Aprendizado de Máquina (AM) como a área de pesquisa que consiste em construir programas de computadores que possam "aprender" com a experiência conforme a tarefa vai sendo executada, através de percepções, o agente aprende a melhor maneira de resolver ou atuar, além de estruturar o conhecimento existente para levar a um entendimento do aprendizado. AM é bastante utilizado em sistemas classificadores e existem várias técnicas que podem ser utilizadas, dentre elas, Árvores de Decisão, Redes Neurais Artificiais, Algoritmos Evolutivos e Algoritmos de Aprendizagem por Reforço (SANTOS, 2005).

Segundo ANDRADE (2004), os jogos de damas e xadrez foram os primeiros a utilizarem técnicas de AM, embora a maior parte dos jogos desenvolvidos no meio comercial ainda utilizem pré-programação, já existem melhores práticas adotadas. Segundo ANDRADE (2004), a pré-programação consiste no desenvolvimento de *scripts*, durante a fase de construção do jogo, que indicam o comportamento inteligente através de regras lógicas. Mas, como ANDRADE (2004) menciona, essa abordagem é ainda bastante limitada, exige um difícil processo de engenharia do conhecimento, e a evolução e adaptação não é realizada de forma dinâmica. Dessa forma, é bastante relevante considerar o uso de aprendizado de máquina para o desenvolvimento da inteligência artificial de jogos digitais, principalmente jogos digitais educacionais.

Existem basicamente três tipos de Aprendizado de Máquina, são eles: Aprendizado Supervisionado, Aprendizado Não Supervisionado e Aprendizado por Reforço. Segundo ANDRADE (2004), aprendizado supervisionado se dá quando existe uma espécie de "professor" que mostra ao sistema ou agente qual deveria ter sido a ação a ser realizada para cada estado ou situação. No aprendizado não-supervisionado não existe este "professor", a aprendizagem acontece apenas através da percepção de padrões durante a interação do sistema ou agente com o ambiente. Já o aprendizado por reforço acontece também através da interação do sistema ou agente com o ambiente, porém existe um crítico que apenas indica o caminho correto, mas não mostra exatamente a resposta correta (ANDRADE, 2004). ANDRADE (2004) ainda faz uma analogia interessante: ele afirma que a aprendizagem por reforço é inspirada na aprendizagem infantil humana.

> *Uma criança costuma realizar ações aleatórias, e, de acordo com as respostas de seus pais (elogio ou reclamação), aprende quais daquelas ações são boas e quais são ruins. A aprendizagem por reforço é utilizada quando não se consegue obter exemplos de comportamento correto para as situações que o agente enfrenta ou quando o agente atuará em um ambiente desconhecido.* (ANDRADE, 2004, p. 18)

LENZ (2009, p. 11) já conceitua de maneira mais específica o sistema supervisionado e não-supervisionado:

> *O sistema supervisionado diz que o algoritmo de aprendizado (indutor) recebe um conjunto de exemplos de treinamento para os quais os rótulos da classe associada são conhecidos. Cada exemplo (instância) _e descrito por um vetor de valores (atributos) e pelo rótulo da classe associada. Um indutor pode ser visto como um algoritmo de AM capaz de criar uma classificação a partir de um conjunto de exemplos. Seu principal objetivo está em extrair conceitos expressos em alguma linguagem, por exemplo, regras de produção ou árvores de decisão, capazes de serem aplicadas a novos casos. No aprendizado não-supervisionado, o indutor analisa os exemplos fornecidos e tenta determinar se alguns deles podem ser agrupados de alguma maneira, formando agrupamentos ou clusters. Após a determinação dos agrupamentos, em geral, é realizada uma análise para determinar o que cada agrupamento significa no contexto do problema analisado.*

O *Laça Palavras* baseou sua inteligência artificial no algoritmo de aprendizagem por reforço Q-Learning, que será explicado mais adiante. Como o objetivo da inteligência é realizar o ajuste dinâmico de dificuldade, o aprendizado por reforço é o tipo de aprendizagem mais adequado, pelo fato de ser adaptável a cada usuário diferente.

4.5.2 Aprendizagem por Reforço

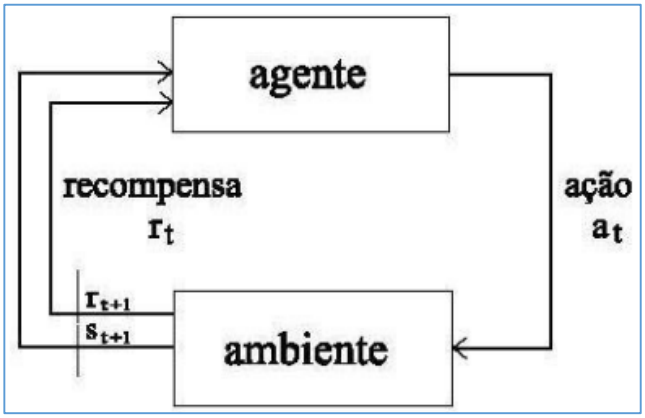
Segundo ANDRADE (2004), "aprendizagem por reforço é um tipo de problema de aprendizagem e não exatamente uma técnica de aprendizagem". Segundo MITCHELL (1997 *apud* MARTINS, 2007) a aprendizagem por reforço refere-se a problemas que o agente ou sistema deve aprender a selecionar as ações disponíveis, que alteram o estado do ambiente e utilizam uma recompensa ou reforço para definir a qualidade da sequência de ações.

Seu funcionamento, consiste em um agente ou sistema que atua em um determinado ambiente através da percepção e realização de ações. Aquilo que o agente ou sistema possui controle absoluto faz parte dele e o que ele não pode modificar diretamente é considerado como parte do ambiente (MARTINS, 2007). Após a realização de alguma ação o ambiente fornece um sinal de retorno, que também pode ser chamado de reforço ou recompensa, tal reforço indica a qualidade da ação realizada, ou seja, se a ação foi boa ou ruim (MARTINS, 2007). Assim, o sistema ou o agente aprende através das suas experiências à medida que interage com o ambiente para atingir um objetivo (MARTINS, 2007). Entende-se por estado do ambiente como a situação das propriedades do ambiente, uma informação qualquer do ambiente como uma sensação imediata, "uma versão processada desta sensação ou uma estrutura complexa" (MARTINS, 2007).

> *O agente deve descobrir quais ações têm maiores recompensas e seu objetivo é maximizar as recompensas em curto e longo prazo. Aprendizado por Reforço é aprender o que fazer dependendo de cada situação possível.* (MARTINS, 2007, p. 553)

A figura 4.42 representa de maneira simples a interação entre agente e ambiente.

Figura 4.42: Representação básica da Aprendizagem por Reforço.



Fonte: MARTINS (2007, p. 553).

Segundo ANDRADE (2004, p. 19), "o sinal de reforço é a base do aprendizado do agente ou sistema e deve indicar o objetivo a ser alcançado". Por isso é muito importante que o reforço seja dado de maneira correta, em coerência com o objetivo final.

> *Por exemplo, em um jogo de damas o reforço pode ser dado ao agente apenas ao final do jogo, sendo positivo quando o agente ganhar ou negativo quando perde ou empata. Com isso, o reforço está mostrando ao agente que seu objetivo é ganhar o jogo, e não perder ou empatar.* (ANDRADE, 2004, p. 19)

Segundo SUTTON (1998 *apud* MARTINS, 2007), o Aprendizado por Reforço é diferente de outras técnicas por utilizar da avaliação das ações tomadas, enquanto que em outros métodos são utilizados instruções ou exemplos que foram treinados anteriormente onde ditam as ações corretas de acordo com cada situação generalizando tais exemplos a situações novas.

Alguns dos principais algoritmos de Aprendizado por Reforço são: Q-Learning; Q(λ); R-Learning; H-Learning; e SARSA. O método escolhido para ser implementado no *Laça Palavras* foi baseado no Q-Learning.

4.5.3 Algoritmo Q-Learning

O algoritmo Q-Learning é uma das técnicas mais populares que pode ser utilizada para a realização da aprendizagem por reforço em um sistema ou agente. Foi proposto por WATKINS (1992), e tem como característica a atualização de valores descontados de recompensas esperadas formalmente representados por Q(*s*,*a*). A cada iteração com o ambiente, definida de acordo com o problema, os valores de recompensa Q são atualizados no

decorrer do aprendizado de acordo com a equação mostrada na figura 4.43 (ANDRADE, 2004):

Figura 4.43: Equação de atualização do algoritmo Q-Learning.

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma . \max {}_a Q(s',a) - Q(s,a)]$$

Fonte: ANDRADE (2004, p. 20).

Q(s,a) representa o valor a ser atualizado em um determinado estado *s* executando uma ação *a*, estado pode ser entendido como a situação em que se encontra os elementos relevantes do ambiente onde o sistema ou agente está inserido em um determinado tempo, γ é o fator de desconto utilizado para garantir que os valores de Q sejam finitos e α é a taxa de aprendizagem, onde que: $0 < \alpha \leq 1$ e $0 \leq \gamma < 1$. Essas taxas são parâmetros que são definidos de acordo com a aplicação (ANDRADE, 2004).

Após executar a ação *a* o sistema ou agente deixa o estado *s* e vai para outro estado *s'* recebendo uma recompensa imediata *r* pela realização de tal ação (ANDRADE, 2004). No cálculo da equação é utilizada a recompensa máxima partindo do estado *s'* realizando a ação *a*, ou seja, que tenha o maior valor de retorno esperado, representado por max $_aQ(s', a)$ (ANDRADE, 2004).

Segundo ANDRADE (2004), é importante notar na equação de atualização do algoritmo Q-Learning a sua eficiência computacional que se utiliza apenas operações básicas. Também não é utilizado nenhum conhecimento da dinâmica do ambiente, apenas das variáveis definidas pelo problema de aprendizagem por reforço como *s*, *s'*, *a*, *r* e parâmetros de aprendizado como *a* e γ .

No algoritmo Q-Learning como recompensas altas estão relacionadas a um bom desempenho, o agente tende a escolher sempre a cada estado *s* a ação *a* que possui o maior valor Q(*s*, *a*), ou seja, o Q-Learning tende a seguir sempre a política ótima estimada a medida que vai aprendendo (FARIA, 1999). Porém, esse comportamento tende a fazer o agente ou sistema perder um pouco a sua capacidade de aprendizagem. Assim, uma boa escolha que pode ser interessante, por exemplo, é escolher em 70% dos casos a ação que retorne o valor máximo e no restante 30% realizar escolhas aleatórias, isso permite que o algoritmo em determinadas iterações explore outras possibilidades. A Figura 4.44 mostra o algoritmo Q-Learning (FARIA, 1999).

Figura 4.44: Algoritmo Q-Learning.

Inicialize $Q(s, a)$ arbitrariamente
Repita (para cada episódio)
  Inicialize s
  Repita para cada passo do episódio
    Escolha $a \in A(s)$
    Execute a ação a
    Observe os valores s' e r
    $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma eQ(s') - Q(s, a)]$
    $s \leftarrow s'$
  até que s seja terminal

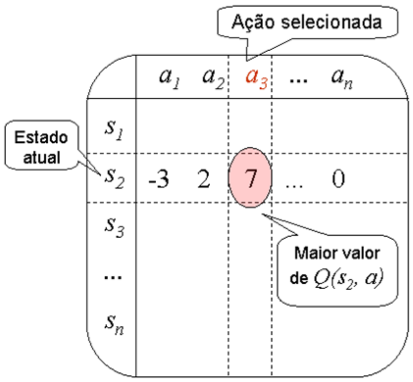
Fonte: FARIA (1999, p. 3).

Uma característica interessante no algoritmo Q-Learning e que o torna bastante popular é a sua simplicidade.

> *A abordagem mais simples e também mais popular representa os valores de recompensa Q por uma matriz bidimensional, que pode ser denominada Q-Table e possui os estados em uma dimensão (linhas) e as ações na outra dimensão (colunas).* (ANDRADE, 2004, p. 20)

A Figura 4.45 é um exemplo básico desta representação.

Figura 4.45: Exemplo de matriz de aprendizagem.



Fonte: ANDRADE (2004, p. 21).

Apesar de ser bastante simples de implementar, essa abordagem de representação através de matriz apresenta uma certa limitação pelo fato de consumir muito espaço computacionalmente, tornando-se inviável para aplicações que possuem muitos de estados e ações (ANDRADE, 2004).

O Q-Learning é uma boa opção para ser utilizado em vários tipos de aplicações que não podem realizar aprendizado supervisionado e que tem como objetivo resolver o problema da adaptação ao usuário necessitando estar em constante aprendizado.

4.5.4 Descrição do Algoritmo

O método de aprendizagem por reforço proposto neste trabalho tem como objetivo o ajuste dinâmico de dificuldade no jogo educacional ortográfico *Laça Palavras* e foi baseado no algoritmo Q-Learning. O ajuste dinâmico de dificuldade é importante no *Laça Palavras* para possibilitar a detecção do nível que o usuário está, reforçar a memorização de palavras que o usuário possui mais dificuldade e que usuários de diferentes níveis consigam jogar e aprender de maneira mais divertida, sem ficarem entediados por ser muito fácil ou desmotivados por ser muito difícil.

De acordo com as características do Q-Learning o mesmo foi escolhido pelo fato de ser bastante popular na literatura, ser fácil de implementar, ser leve computacionalmente (o que é bastante desejável para ser executado em dispositivos móveis) e por ser uma boa opção para resolver o problema do ajuste dinâmico de dificuldade através da aprendizagem por reforço.

Como já foi explicado anteriormente, o *Laça Palavras* possui quatro níveis de dificuldade, onde a cada nível vai sendo incorporado e apresentado classes de palavras da língua portuguesa que se enquadram no caso da concorrência, que consiste em que duas letras estão aptas a representar o mesmo som, no mesmo lugar.

O Q-Learning é um algoritmo de aprendizagem por reforço que trabalha de forma iterativa, dessa forma, foi definido no *Laça Palavras* que cada iteração consiste em apresentar ao usuário dez palavras para serem completadas.

A sequência de passos do método implementado se dá da seguinte forma a cada iteração: primeiramente, o usuário responde dez palavras de um determinado nível; após o término das dez palavras, é calculado seu desempenho. É considerado um bom desempenho se a taxa de acertos das palavras for igual ou superior a 60% (parâmetro comumente usado), caso contrário é considerado um desempenho ruim; após a análise do desempenho é calculado o reforço e atualizadas as recompensas de acordo com a equação de atualização do Q-Learning, as recompensas no *Laça Palavras* foram definidas como probabilidades, cada nível tem sua probabilidade para ser selecionado; após o cálculo do reforço e a atualização das recompensas (probabilidades), é selecionado o nível que possui a maior probabilidade. Essa sequência de passos é executada a cada iteração. Cada passo será explicado com mais detalhes adiante.

O *Laça Palavras* possui um total de 40 palavras diferentes nesta primeira versão, que são divididas em quatro grupos e possuem a característica da concorrência (como foi explicado anteriormente). Cada nível representa predominantemente um grupo de palavras. O primeiro nível possui palavras que contém “Ç” e “SS”, mas que possuem som de [s] entre vogais (exemplo “posso”). O segundo nível possui 90% de palavras que contém “U” e “L”, mas que possuem som de [u] (exemplo “papel”) e 10% de palavras do primeiro nível. O terceiro nível possui 80% palavras que contém “G” e “J”, mas que possui som de [g] entre vogais (exemplo “majestade”), 10% de palavras do primeiro nível e 10% de palavras predominantes do segundo nível. No quarto e último nível contém 70% de palavras que contém “S” e “Z”, mas que possuem som de [z] entre vogais (exemplo “casa”), 10% de palavras do primeiro nível, 10% de palavras predominantes do segundo nível e 10% de palavras predominantes do terceiro nível.

O jogo sempre é iniciado no primeiro nível para qualquer usuário. A partir disso a mudança de nível é definida pelo algoritmo de inteligência. Para cada nível do jogo é armazenado um valor decimal de probabilidade que pode variar entre 0 e 1. Esses valores de probabilidade são alterados de acordo com o reforço e recompensa calculada a cada iteração para que o nível mais adequado seja selecionado de acordo com o desempenho do usuário.

O desempenho do usuário é calculado a cada iteração, ou seja, a cada vez que o usuário responde 10 palavras de um determinado nível. O cálculo é realizado da seguinte forma:

desempenho = (quantidade de acertos / quantidade total de palavras)

O valor do desempenho pode variar de 0 a 1 e é o reforço que o jogo precisa para calcular as recompensas.

A figura 4.46 apresenta o código implementado em linguagem C# da função que calcula o desempenho do usuário.

Figura 4.46: Função que calcula o desempenho do usuário.

```
public float calcula_desempenho()           //cálculo do desempenho
{
    return (float)(bancoPalavras.Instance.acertos) / (bancoPalavras.Instance.qtd_Words);
}
```

Fonte: Elaborada pelo autor.

Após responder às 10 palavras e realizar o cálculo do desempenho, a recompensa é calculada. O jogo precisa saber se as palavras que foram apresentadas na iteração foram fáceis ou difíceis para o usuário e se precisa manter, aumentar ou diminuir o nível, é através do cálculo da recompensa e da atualização das probabilidades que indica o que deverá ser feito na próxima iteração, ou seja, qual nível deverá ser selecionado dinamicamente para a próxima iteração.

A recompensa é calculada de duas formas diferentes: uma quando o desempenho do usuário for igual ou maior que 0,6 e outra quando o desempenho for menor que 0,6. A recompensa é calculada baseada na equação do Q-Learning mostrada abaixo.

$$\alpha[r + \gamma.\max {}_{a}Q(s',a) - Q(s,a)]$$

No *Laça Palavras* essa equação pode ser interpretada da seguinte forma:

- α é a taxa de aprendizado, que foi fixada como valor 1 com o objetivo fazer o algoritmo se adaptar ao usuário de maneira rápida;
- *r* é o reforço, que no *Laça Palavras* é o valor do desempenho do usuário na última iteração;
- γ é a taxa de desconto, que foi fixada como valor 0,9 por ser bastante utilizado na bibliografia;
- $\max_a Q(s', a)$ é a recompensa (probabilidade) máxima que o nível subsequente inferior ou superior (dependendo do desempenho) já teve, para cada nível é armazenada a recompensa (probabilidade) máxima que já teve no decorrer das iterações;
- $Q(s,a)$ é a recompensa (probabilidade) atual do nível referente às últimas 10 palavras apresentadas na última iteração.

Todos esses valores são utilizados na equação para gerar um valor único de recompensa, que será somado ou subtraído com as probabilidades já existentes de cada nível, ou seja, a atualização das probabilidades.

Se o desempenho do usuário for maior ou igual a 0,6 a recompensa será calculada e atualizada da seguinte forma:

recompensa = (desempenho) + (0,9 x (probabilidade máxima do nível subsequente **superior**)) – (probabilidade do nível referente às últimas 10 palavras apresentadas).

A figura 4.47 apresenta o trecho do código implementado em linguagem C# que realiza o cálculo da recompensa quando o usuário tem um desempenho maior ou igual a 0,6.

Figura 4.47: Código do cálculo da recompensa para um bom desempenho.

```
reforco = dadosJogo.Instance.ultimo_desempenho +
         (0.9 * (dadosJogo.Instance.vetor_reforco_max[dadosJogo.Instance.currentUser.Nivel + 1]))
          - dadosJogo.Instance.vetor_reforco_atual[dadosJogo.Instance.currentUser.Nivel];
```

Fonte: Elaborada pelo autor.

Na figura 4.47, “dadosJogo.Instance.ultimo_desempenho” é o desempenho do usuário na última iteração, “dadosJogo.Instance.vetor_reforco_max” é o vetor que armazena a probabilidade máxima de cada nível no decorrer das iterações, sendo que na figura 4.47 é utilizada a probabilidade máxima do nível subsequente superior. A variável “dadosJogo.Instance.currentUser.Nivel” armazena o nível referente às últimas 10 palavras apresentadas, “dadosJogo.Instance.vetor_reforco_atual” é o vetor que armazena a probabilidade corrente de cada nível, assim, na figura 4.47 representa a probabilidade do nível referente às últimas 10 palavras apresentadas.

Esse valor de recompensa é subtraído com a probabilidade do último nível apresentado e com as probabilidades dos níveis inferiores. Ao contrário, nas probabilidades dos níveis superiores essa recompensa é somada.

Para ficar mais claro é possível exemplificar. Supondo que, em determinado momento do jogo, o usuário se encontra no nível 2 e responde 10 palavras, ao terminar de respondê-las o desempenho calculado resulta em 0,7. Como 0,7 é maior que 0,6, é considerado um bom desempenho. Supõe-se que a probabilidade atual do nível 1 seja 0,0 e dos níveis 2, 3 e 4 sejam 0,8, 0,6 e 0,5 respectivamente. Supõe-se também que as probabilidades máximas de cada nível sejam as mesmas das atuais. Assim, a recompensa nesse exemplo, será calculada da seguinte forma:

recompensa = (0,7) + (0,9 x (0,6)) – (0,8);

recompensa = 0,44.

Após o cálculo da recompensa as probabilidades são atualizadas da seguinte forma nesse exemplo:

probabilidade atual do nível 1 = probabilidade atual do nível 1 (0,0) – recompensa (0,44);

probabilidade atual do nível 2 = probabilidade atual do nível 2 (0,8) – recompensa (0,44);

probabilidade atual do nível 3 = probabilidade atual do nível 3 (0,6) + recompensa (0,44);

probabilidade atual do nível 4 = probabilidade atual do nível 4 (0.5) + recompensa (0,44).

A figura 4.48 apresenta o código implementado em linguagem C# que realiza a atualização das probabilidades de cada nível no caso do exemplo com um bom desempenho.

Figura 4.48: Código da atualização das probabilidades no caso do exemplo.

```
dadosJogo.Instance.vetor_reforco_atual[0] -= reforco;
dadosJogo.Instance.vetor_reforco_atual[dadosJogo.Instance.currentUser.Nivel] -= reforco;
dadosJogo.Instance.vetor_reforco_atual[2] += reforco;
dadosJogo.Instance.vetor_reforco_atual[3] += reforco;
```

Fonte: Elaborada pelo autor.

Na figura 4.48 “dadosJogo.Instance.vetor_reforco_atual” é um vetor que armazena as probabilidades de cada nível em casa posição. Na posição “0” armazena a probabilidade do

nível 1, na posição “1” probabilidade do nível 2 e assim sucessivamente. A variável “reforco” armazena o valor da recompensa calculada.

Como o desempenho foi bom, percebe-se que a recompensa foi subtraída no nível que apresentou as palavras e no nível inferior. Já nos níveis posteriores a recompensa foi somada.

Se o desempenho do usuário for menor que 0,6 a recompensa será calculada e atualizada de forma um pouco diferente, como mostra abaixo:

recompensa = (desempenho) + (0,9 x (probabilidade máxima do nível subsequente **inferior**)) – (probabilidade do nível referente às últimas 10 palavras apresentadas).

A figura 4.49 apresenta o trecho do código implementado em linguagem C# que realiza o cálculo da recompensa quando o usuário tem um desempenho menor que 0,6.

Figura 4.49: Código do cálculo da recompensa para um desempenho ruim.

```
reforco = dadosJogo.Instance.ultimo_desempenho +
    (0.9 * (dadosJogo.Instance.vetor_reforco_max[dadosJogo.Instance.currentUser.Nivel - 1]))
    - dadosJogo.Instance.vetor_reforco_atual[dadosJogo.Instance.currentUser.Nivel];
```

Fonte: Elaborada pelo autor.

Percebe-se que nesse caso o cálculo da recompensa utiliza a probabilidade máxima do nível subsequente inferior e não superior, diferentemente do caso anterior. Na atualização das probabilidades é efetuada a soma nos níveis anteriores e realizada a subtração nos níveis posteriores. Utilizando o mesmo exemplo do caso anterior, supõe-se que ao invés do usuário ter seu desempenho resultado em 0,7 seu desempenho resultou em 0,5 e que as probabilidades atuais dos níveis são as mesmas utilizadas no caso anterior. Assim, o cálculo se daria da seguinte forma:

recompensa = (0,5) + (0,9 x (0,0)) – (0,8);

recompensa = -0,3.

Percebe-se que, ao calcular a equação, exemplificando um desempenho ruim, a recompensa foi mais baixa. Como o resultado foi negativo deve-se transformá-lo em positivo para realizar as atualizações conforme abaixo:

probabilidade atual do nível 1 = probabilidade atual do nível 1 (0,0) + recompensa (0,3);

probabilidade atual do nível 2 = probabilidade atual do nível 2 (0,8) - recompensa (0,3);

probabilidade atual do nível 3 = probabilidade atual do nível 3 (0,6) - recompensa (0,3);

probabilidade atual do nível 4 = probabilidade atual do nível 4 (0.5) - recompensa (0,3).

Na figura 4.50 apresenta o código implementado em linguagem C# que realiza a atualização das probabilidades de cada nível no caso do exemplo com um desempenho ruim.

Figura 4.50: Código da atualização das probabilidades no caso do exemplo.

```
dadosJogo.Instance.vetor_reforco_atual[0] += reforco;
dadosJogo.Instance.vetor_reforco_atual[dadosJogo.Instance.currentUser.Nivel] -= reforco;
dadosJogo.Instance.vetor_reforco_atual[2] -= reforco;
dadosJogo.Instance.vetor_reforco_atual[3] -= reforco;
```

Fonte: Elaborada pelo autor.

Nota-se que ao contrário do caso anterior as atualizações são realizadas de forma diferente. No nível inferior a recompensa é somada e nos níveis superiores a recompensa é subtraída, como o desempenho foi ruim as probabilidades dos níveis superiores tendem a diminuir.

Depois de atualizada as probabilidades de cada nível, é verificado se os valores estão entre 0 e 1. Por representarem probabilidades os valores que estiverem fora desse intervalo (entre 0 e 1) devem ser acertados ao limite. Por exemplo, após as atualizações das recompensas, se a probabilidade de algum nível resultar em algum valor maior que 1 ou menor que 0 deve ser transformado em 1 ou em 0.

O último passo do algoritmo de inteligência é selecionar o nível que possui a maior probabilidade para apresentar as próximas 10 palavras a serem respondidas. Em determinado momento pode acontecer de mais de um nível possuir a mesma probabilidade. Nesse caso, se dois ou mais níveis possuírem o mesmo valor e os mesmos forem os valores mais altos, é selecionado o nível de menor dificuldade dentre eles.

A figura 4.51 apresenta o código implementado em linguagem C# que seleciona o nível de maior probabilidade para apresentar as próximas palavras depois das atualizações realizadas pelo algoritmo.

Figura 4.51: Código que seleciona o nível de maior probabilidade.

```
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;
    }
}
```

Fonte: Elaborada pelo autor.

Percebe-se na figura 4.51 que o vetor “dadosJogo.Instance.vetor_reforco_atual”, que armazena as probabilidades de cada nível, é percorrido por um laço verificando qual probabilidade é maior. O nível “ i ” que tem a maior probabilidade é colocado como nível corrente do jogo para apresentar as próximas palavras.

O Algoritmo 2 apresenta o pseudocódigo completo do algoritmo de aprendizado.

Algoritmo 2: Algoritmo de aprendizagem implementado baseado no Q-Learning.

A cada iteração (acabou de responder 10 palavras) faça

desempenho = (quantidade de acertos / quantidade total de palavras);

Se desempenho do usuário >= 0,6

//recompensa baseada na equação

$$\alpha[r + \gamma.\max {}_{a}Q(s',a) - Q(s,a)]$$

//cálculo da recompensa

recompensa = (desempenho) + (0,9 x (probabilidade máxima do nível subsequente **superior**)) – (probabilidade do nível referente às últimas 10 palavras apresentadas);

atualiza as probabilidades referentes a cada nível de acordo com a recompensa calculada;

Se não

Se desempenho do usuário < 0,6

//recompensa baseada na equação

$$\alpha[r + \gamma.\max {}_{a}Q(s',a) - Q(s,a)]$$

//cálculo da recompensa

recompensa = (desempenho) + (0,9 x (probabilidade máxima do nível subsequente **inferior**)) – (probabilidade do nível referente às últimas 10 palavras apresentadas);

atualiza as probabilidades referentes a cada nível de acordo com a recompensa calculada;

Fim se

Seleciona o nível que contém a maior probabilidade;

Fim iteração

Capítulo 5

Testes e comportamento do Laça Palavras

Este capítulo tem como objetivo apresentar de maneira simples o comportamento do jogo *Laça Palavras* através de dois testes. Não é um experimento de avaliação, é apenas uma apresentação do comportamento do ajuste dinâmico de dificuldade do *Laça Palavras* através de dois exemplos simulados de situações que podem ocorrer.

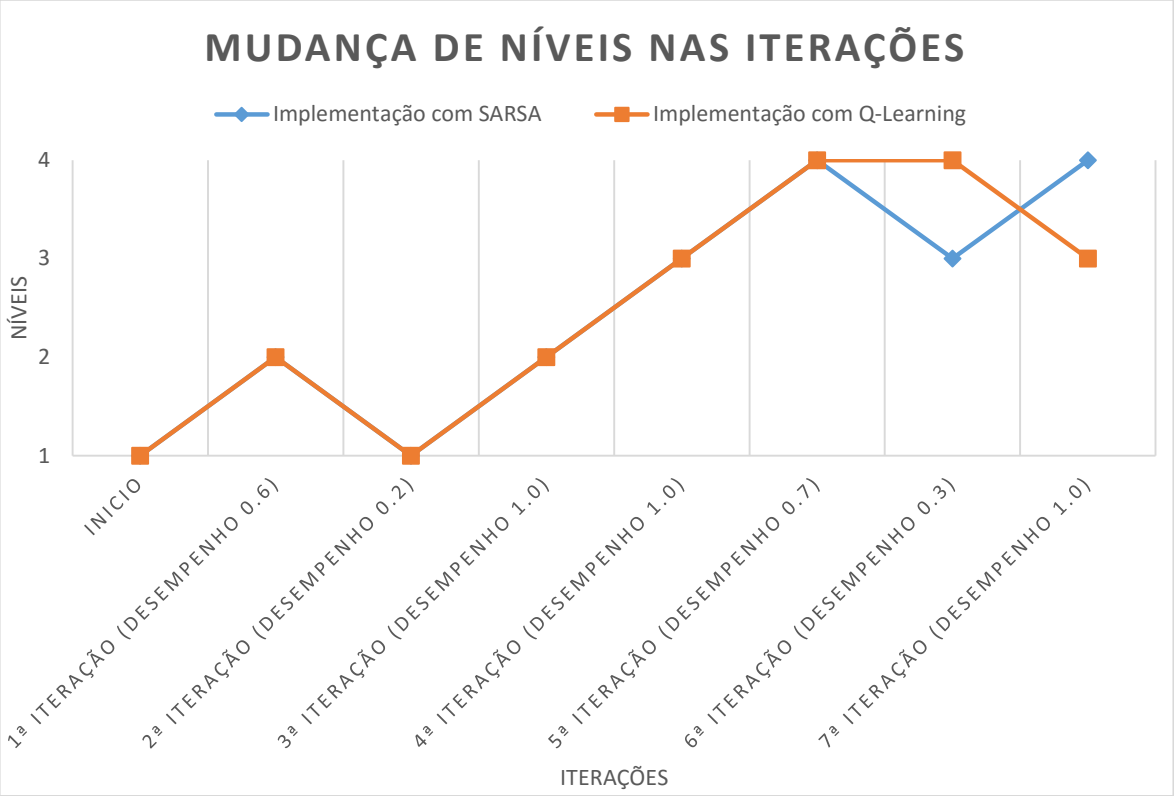
Os testes consistem em simular dois jogadores em que cada um deles tem como objetivo completar todas as palavras e finalizar o jogo. Cada simulação apresenta uma sequência de desempenho nas iterações, que consiste em desempenhos bons e ruins. O propósito é mostrar como o ajuste dinâmico de dificuldade se comporta a cada iteração.

Foi realizada uma simples comparação entre o comportamento do *Laça Palavras* desenvolvido neste trabalho com o *Laça Palavras* desenvolvido no trabalho de SANTOS (2016), com o objetivo de enriquecer mais a apresentação do comportamento do jogo através das simulações. A diferença entre ambos é que o *Laça Palavras* desenvolvido no trabalho de SANTOS (2016) foi implementado com um método de aprendizagem por reforço baseado no algoritmo SARSA e não no algoritmo Q-Learning, como realizado neste trabalho. Ambos os algoritmos são parecidos, porém, o Q-Learning tem uma característica de sempre buscar uma política ótima de comportamento, já o SARSA tem uma característica maior de exploração.

A primeira simulação foi realizada através da seguinte sequência de desempenhos a cada iteração: 60%, 20%, 100%, 100%, 70%, 30% e 100%.

O comportamento de ambas as implementações (Q-Learning e SARSA) em relação a mudança de níveis em cada iteração pode ser observado no gráfico 5.1.

Gráfico 5.1: Mudança dos níveis de dificuldade durante as iterações da primeira simulação.

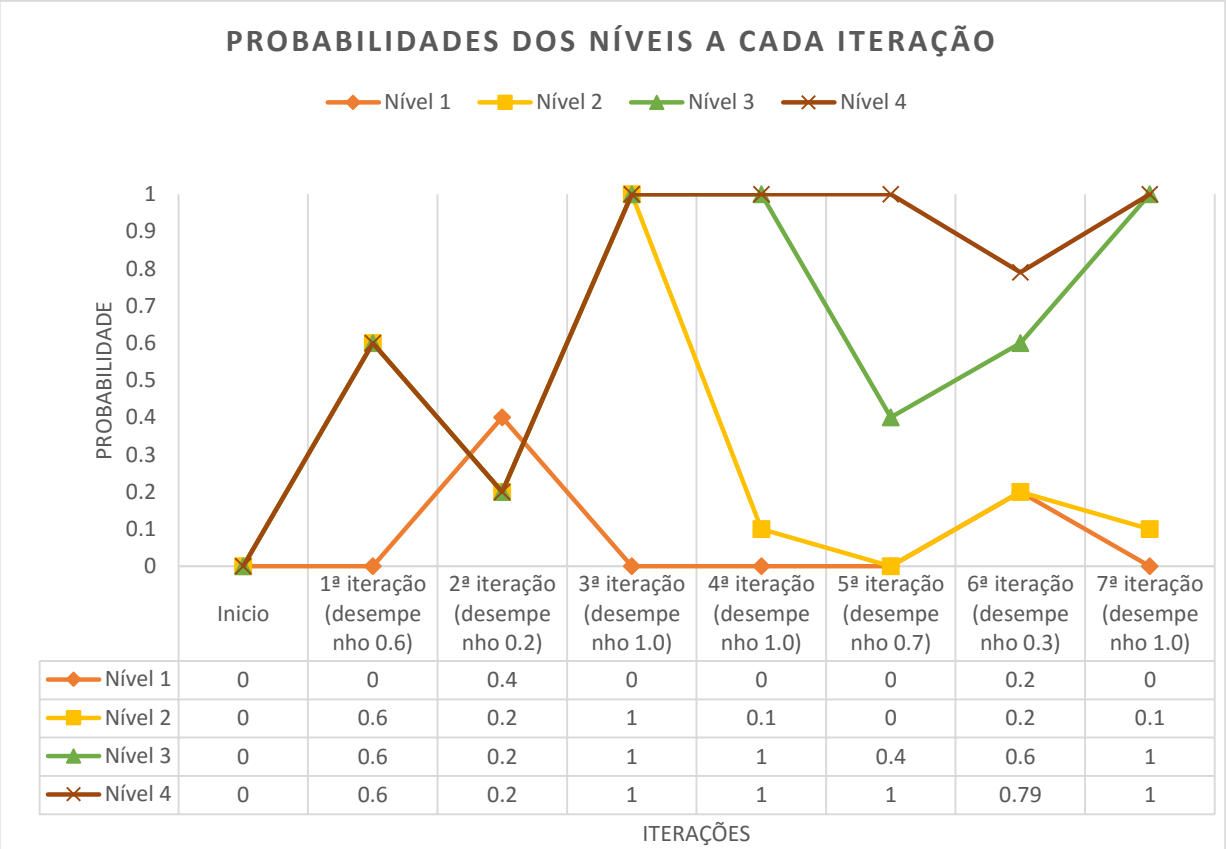


Fonte: Elaborado pelo autor.

Percebe-se no gráfico acima que ambos algoritmos têm um comportamento bastante interessante e coerente. Nota-se que quando o desempenho teve uma taxa satisfatória, o jogo tende a aumentar a dificuldade dinamicamente, da mesma forma quando o desempenho foi insatisfatório o jogo tende a manter ou diminuir o nível. Cada valor referente a cada iteração no gráfico significa o nível que se chegou após a mesma. Por exemplo, após a 2ª iteração desempenhando apenas 20% de acertos (0,2) em ambos os algoritmos o nível é diminuído dinamicamente para o nível 1. Nota-se que o comportamento de ambos algoritmos é bastante parecido, mas, nas últimas iterações existe diferença na mudança de níveis. Na 6ª iteração o método baseado no Q-Learning não diminui o nível, diferentemente do método baseado no SARSA que diminui.

Nessa mesma simulação, foram construídos gráficos que mostram os valores das probabilidades de cada nível a cada iteração à medida que são atualizadas. As atualizações das probabilidades do método baseado no Q-Learning são mostradas no gráfico 5.2.

Gráfico 5.2: Atualizações das probabilidades a cada iteração do método baseado no Q-Learning na primeira simulação.



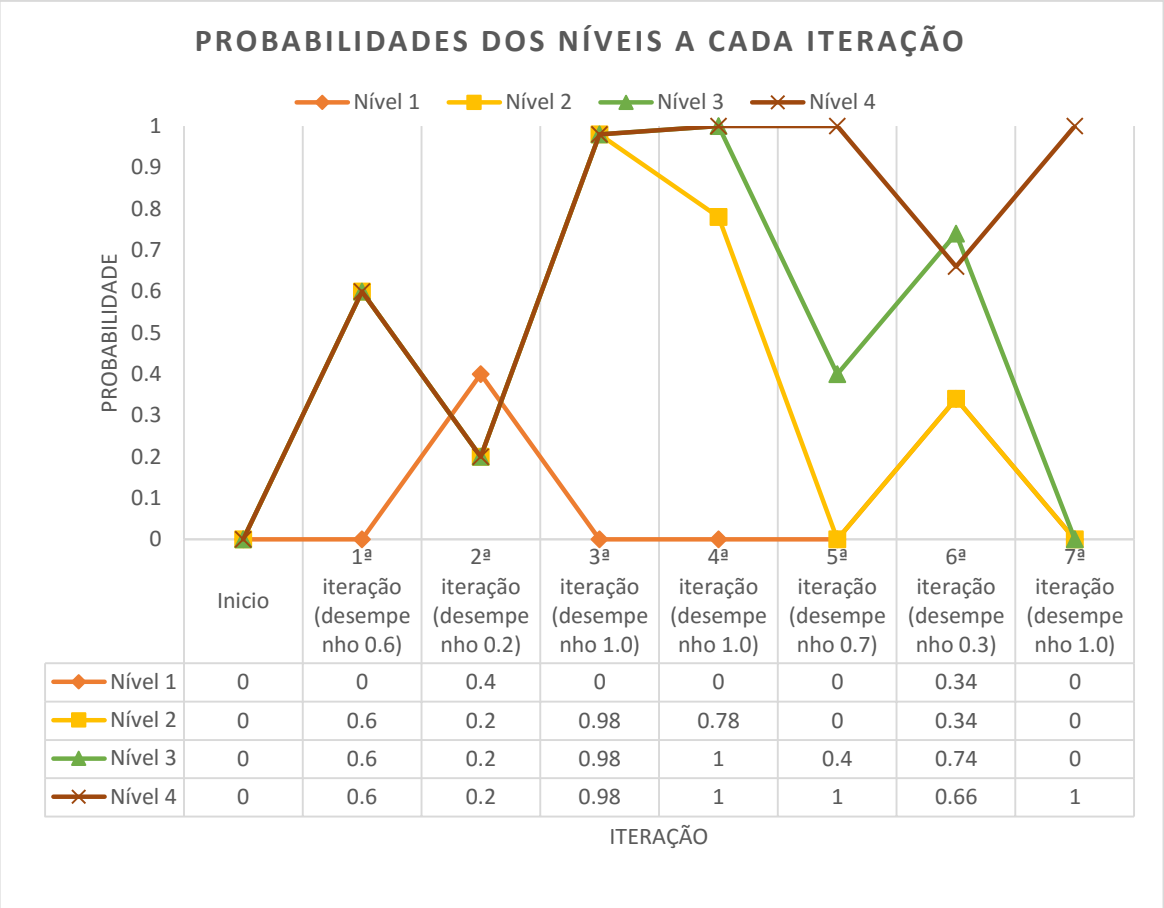
	Inicio	1ª iteração (desempenho 0.6)	2ª iteração (desempenho 0.2)	3ª iteração (desempenho 1.0)	4ª iteração (desempenho 1.0)	5ª iteração (desempenho 0.7)	6ª iteração (desempenho 0.3)	7ª iteração (desempenho 1.0)
Nível 1	0	0	0.4	0	0	0	0.2	0
Nível 2	0	0.6	0.2	1	0.1	0	0.2	0.1
Nível 3	0	0.6	0.2	1	1	0.4	0.6	1
Nível 4	0	0.6	0.2	1	1	1	0.79	1

Fonte: Elaborada pelo autor.

Ao observar no gráfico acima percebe-se que durante as iterações do jogo ao realizar bons desempenhos, níveis mais difíceis tendem a aumentar suas probabilidades e níveis mais fáceis tendem a diminuir suas probabilidades. Da mesma forma, ao realizar desempenhos ruins níveis fáceis tendem a aumentar suas probabilidades e níveis mais difíceis a diminuir suas probabilidades. Note-se que as probabilidades dos níveis mais intermediários (nível 2 e nível 3) variam mais, significando aprendizagem rápida à medida que o desempenho se altera e como ocorreu desempenho satisfatório com mais frequência ao longo das iterações o nível 1 teve na maioria das vezes probabilidades baixas e o nível 4 probabilidades mais altas.

O gráfico 5.3 mostra as atualizações das probabilidades dos níveis do algoritmo baseado no SARSA.

Gráfico 5.3: Atualizações das probabilidades a cada iteração do método baseado no SARSA na primeira simulação.



	Inicio	1ª iteração (desempenho 0.6)	2ª iteração (desempenho 0.2)	3ª iteração (desempenho 1.0)	4ª iteração (desempenho 1.0)	5ª iteração (desempenho 0.7)	6ª iteração (desempenho 0.3)	7ª iteração (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

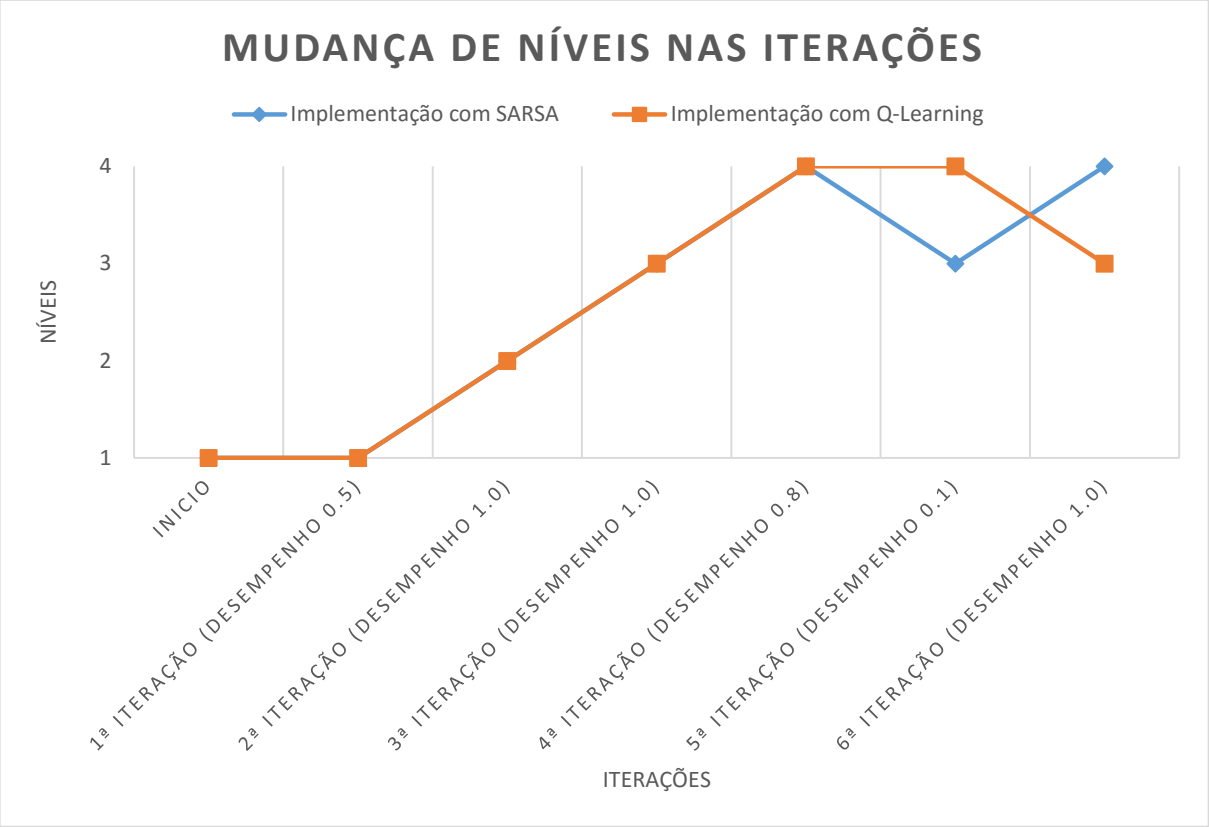
Fonte: Elaborado pelo autor.

Percebe-se que o método implementado baseado no SARSA possui um comportamento parecido nas atualizações das probabilidades dos níveis o que é coerente de acordo com as mudanças de desempenho, porém, as mudanças dos valores no decorrer das iterações são um levemente mais suaves no método baseado no Q-Learning do que no método baseado no SARSA. Ambos os algoritmos parecem cumprir bem seu papel de ajuste dinâmico de dificuldade.

Já a segunda simulação foi realizada através da seguinte sequência de desempenhos a cada iteração: 50%, 100%, 100%, 80%, 10%, 100%.

Também é ilustrado no gráfico 5.4 o comportamento de ambas as implementações (Q-Learning e SARSA) em relação a mudança de níveis em cada iteração nessa segunda simulação.

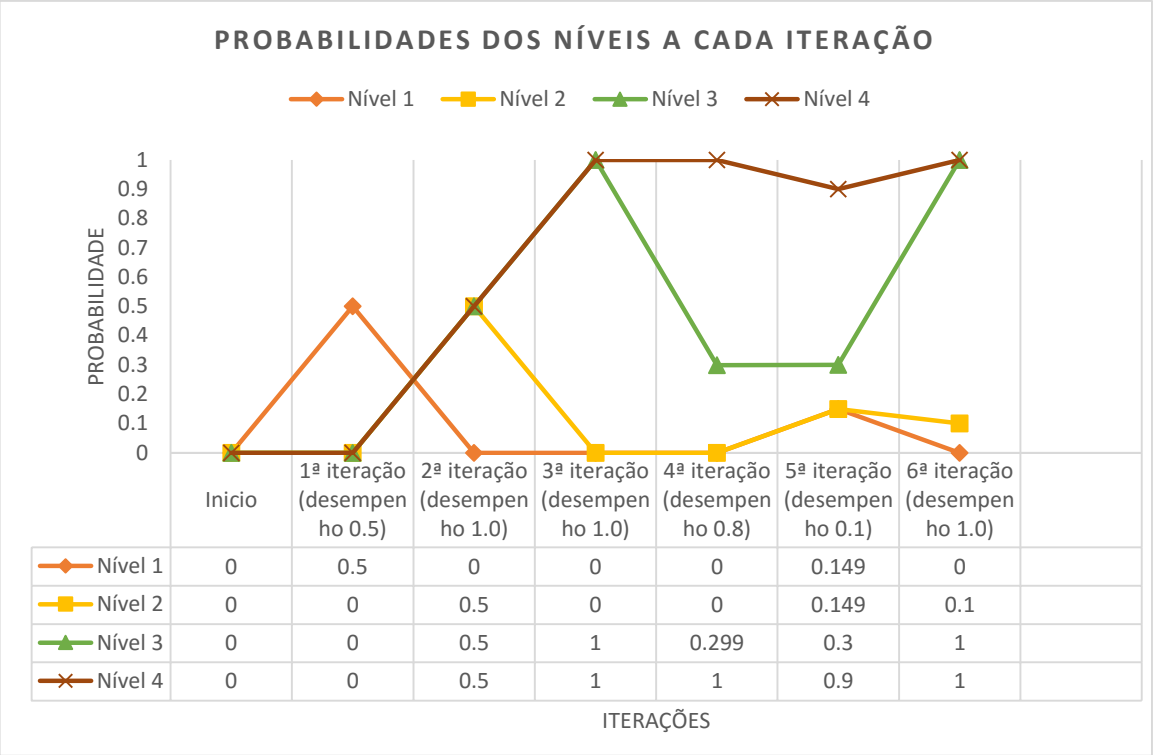
Gráfico 5.4: Mudança dos níveis de dificuldade durante as iterações na segunda simulação.



Diferentemente da primeira simulação, o desempenho começa insatisfatório e o jogo em ambos algoritmos mantém o nível. Nas iterações em que o desempenho é satisfatório, o jogo aumenta o nível de dificuldade, apenas o algoritmo baseado no SARSA diminui do nível 4 para o nível 3 na 5ª iteração quando o desempenho é muito ruim. O algoritmo baseado no Q-Learning na 5ª iteração apenas mantém o jogo no nível 4. Também nessa segunda simulação ambos algoritmos possuem comportamento parecido, mas, o método baseado no Q-Learning parece levar mais em consideração o histórico de recompensas.

Também foi construída nessa segunda simulação gráficos que mostram os valores das probabilidades de cada nível a cada iteração à medida que são atualizadas. O gráfico 5.5 mostra as probabilidades referentes ao método baseado no Q-Learning.

Gráfico 5.5: Atualizações das probabilidades a cada iteração do algoritmo baseado no Q-Learning na segunda simulação.



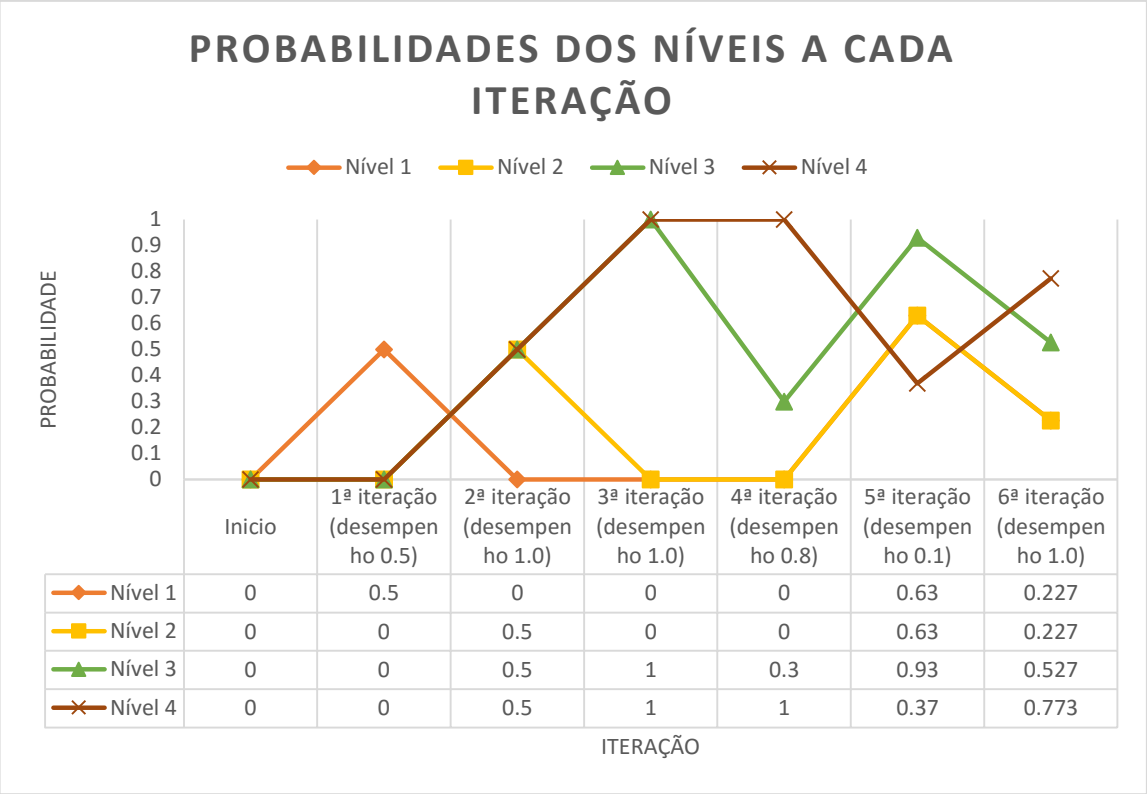
	Inicio	1ª iteração (desempenho 0.5)	2ª iteração (desempenho 1.0)	3ª iteração (desempenho 1.0)	4ª iteração (desempenho 0.8)	5ª iteração (desempenho 0.1)	6ª iteração (desempenho 1.0)
Nível 1	0	0.5	0	0	0	0.149	0
Nível 2	0	0	0.5	0	0	0.149	0.1
Nível 3	0	0	0.5	1	0.299	0.3	1
Nível 4	0	0	0.5	1	1	0.9	1

Fonte: Elaborado pelo autor.

Pode-se notar no gráfico acima que, apesar de a segunda simulação ser diferente da primeira, é importante notar a mesma característica ou comportamento. As probabilidades referentes ao ajuste dinâmico de dificuldade se atualizam a cada iteração de forma coerente de acordo com os desempenhos. Desempenhos ruins tendem a aumentar as probabilidades dos níveis mais baixos e diminuir as dos níveis mais altos. Da mesma forma quando os desempenhos são satisfatórios as probabilidades dos níveis mais altos tendem a aumentar e dos níveis mais baixos a diminuir. Diferentemente da primeira simulação, as probabilidades tiveram uma variação mais suave com exceção apenas do nível 3, isso se deu pelo fato do desempenho variar um pouco menos.

O gráfico 5.6 mostra as atualizações das probabilidades do algoritmo baseado no SARSA também.

Gráfico 5.6: Atualizações das probabilidades a cada iteração do algoritmo baseado no SARSA na segunda simulação.



	Inicio	1ª iteração (desempenho 0.5)	2ª iteração (desempenho 1.0)	3ª iteração (desempenho 1.0)	4ª iteração (desempenho 0.8)	5ª iteração (desempenho 0.1)	6ª iteração (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

Fonte: Elaborado pelo autor.

Na segunda simulação, o comportamento das probabilidades dos níveis no algoritmo baseado no SARSA e similar ao algoritmo baseado no Q-Learning, mas, no final existe uma diferença mostrando uma variação maior do que no Q-Learning. Na 5ª iteração com o desempenho 0.1 a maior probabilidade foi estabelecida para o nível 3, diferentemente do algoritmo baseado no Q-Learning que a maior probabilidade foi para o nível 4. É um comportamento também coerente com os desempenhos e mostra que o ajuste dinâmico de dificuldade é de fato realizado.

Capítulo 6

Conclusão e Trabalhos Futuros

Com a inovação da tecnologia, com uso crescente dos dispositivos móveis e a presença cada vez mais forte dos jogos digitais na vida das pessoas, esse trabalho vem contribuir ainda mais no fortalecimento da utilização das tecnologias da informação na área da educação. Esse trabalho descreve os benefícios que os jogos digitais educacionais, principalmente, os jogos digitais ortográficos promovem no auxílio da aprendizagem e é bastante esclarecedor apresentando a importância da presença de métodos de inteligência artificial nos jogos digitais educacionais, como forma de melhorar a eficácia dos mesmos na aprendizagem, tornando-os mais dinâmicos, menos previsíveis reforçando o aprendizado do usuário nas maiores dificuldades.

Ao final do trabalho como resultado tem-se o jogo digital ortográfico *Laça Palavras* que foi desenvolvido para executar em dispositivos móveis Android e tem como objetivo auxílio nos treinos ortográficos de palavras da língua portuguesa que possuem a característica da concorrência, que consiste em que duas letras estão aptas a representar o mesmo som, no mesmo contexto. Ainda como foco principal foi realizada a implementação de um método de aprendizagem por reforço baseado no algoritmo Q-Learning com o objetivo de mostrar de maneira prática a implementação de um método de inteligência artificial em um jogo digital educacional e principalmente realizar o ajuste dinâmico de dificuldade, tornando o jogo mais eficaz e divertido.

Como trabalho futuro, o jogo *Laça Palavras* pode ser avaliado de maneira mais detalhada com crianças recém-alfabetizadas para testar seu auxílio no treino de palavras da língua portuguesa. Também como trabalho futuro, pode-se ter uma nova versão do jogo com as interfaces e animações melhoradas e com novos cenários que podem focar em auxiliar no aprendizado de outros casos da ortografia da língua portuguesa.

Referências

ALVES, Lynn. Relações entre os jogos digitais e aprendizagem: delineando percurso. In Educação, Formação & Tecnologias. 2008. vol.1(2); pp. 3-10, novembro de 2008, disponível no URL: http://eft.educom.pt.

AMATE, Flávio Cezar. Desenvolvimento de Jogos Computadorizados para Auxiliar a Aquisição da Base Alfabética de Crianças. Universidade de São Paulo. São Carlos. 2007.

ANDRADE G.D. Aprendizagem por Reforço e Adaptação ao Usuário em Jogos Eletrônicos. Trabalho de Graduação. Centro de Informática. Universidade Federal de Pernambuco. Recife, 2004.

ANDRADE, G. D. Balanceamento Dinâmico de Jogos: Uma Abordagem Baseada em Aprendizagem por Reforço. Dissertação (Mestrado em Ciência da Computação) Centro de Informática. Universidade Federal de Pernambuco. Recife, 2006.

ANDROID. A história do Android. Disponível em: <www.android.com/intl/pt-BR_br/history/>. Acesso em 22 de fevereiro de 2016.

ARANHA, Glaucio. Jogos eletrônicos como um conceito chave para o desenvolvimento de aplicações imersivas e interativas para o aprendizado. Ciências & Cognição (UFRJ), On-line, v. 07, p. 105-110, 2006.

ARAUJO, Bruno Baère Pederassi Lomba de. Um estudo sobre adaptatividade dinâmica de dificuldade em jogos. PUC-Rio - Certificação Digital Nº 1012625/CA. Departamento de Informática. PUC-Rio. Rio de Janeiro. Setembro de 2012.

AURELIO, O minidicionário da língua portuguesa. 4ª revista e ampliada do minidicionário Aurélio. 7ª impressão – Rio de Janeiro, 2002.

BARBOSA, Soraia Teixeira; VEIGA, Janaína; CARVALHO, Carlos Vitor. Estudo do Uso de Técnicas de Inteligência Artificial em Jogos 2D. Revista Eletrônica TECCEN, Vassouras, v. 5, n. 1 p. 5-20, jan./abr., 2012.

BATTAIOLA, A. L. Jogos por computador: Histórico, relevância tecnológica e mercadológica, tendências e técnicas de implementação. Anais do XIX Jornada de Atualização em Informática, p. 83–122, 2000.

BOURG, D. M.; SEEMANN, G. AI for Game Developers. – O'Reilly, 2004.

CANALTECH. Conhece todas as versões de Android? Disponível em: < http://canaltech.com.br/dica/mobile/Conhece-todas-as-versoes-de-Android/>. Acesso em 22 de fevereiro de 2016.

CLUA, E. W. G. and BITTENCOURT, J. R. Uma Nova Concepção para a Criação de Jogos Educativos. Anais do XV Simpósio Brasileiro de Informática na Educação – SBIE 2004. 9 a 12 de nov. Manaus AM.

COPPIN, Ben. Inteligência artificial. Rio de Janeiro: LTC, 2010.

COREL. Disponível em: <www.coreldraw.com/br/product/software-de-design-grafico/topNav=br>. Acesso em 9 de janeiro de 2016.

COSTA, Paula Dornhofer Paro; PRAMPERO, Paulo Sergio; SALAZAR, Zady Castañeda. Inteligência Artificial aplicada a Jogos Digitais. Universidade Estadual de Campinas – UNICAMP. Campinas dezembro 2009. Disponível no site: <http://www.dca.fee.unicamp.br/~martino/disciplinas/ia369/trabalhos/t4g1>.

DA SILVA, B. M., & VANDERLINDE, M. Inteligência Artificial, Aprendizado de Máquina. Disponível: <http://www.ceavi.udesc.br/arquivos/id_submenu/387/brigiane_machado_da_silva___marcos_vanderlinde.pdf> Acesso em: 21 junho 2015.

DIAS, Danielle Gomes. O ensino e a aprendizagem da ortografia. Perspectivas Online. www,perspectivasonline.com.br. Volume 3, número 9, 2009.

E. ZERMELO. Uber eine Anwendung der Mengdenlehre auf die theories des Schachspiels. Atas do Décimo Quinto Congresso Internacional de Matemáticos, vol. 2, pp. 501–504, 1913.

FERNANDES, Anita Maria da Rocha. Inteligência artificial: noções gerais. Florianópolis: Visual Books, 2003.

FILHO, Evandro Luis da Rosa Fensterseifer; WEBER, Vinicius. Trabalho Sistemas Operacionais. Disponível em: <http://www-usr.inf.ufsm.br/~efilho/include/site.html>. Acesso em 9 de janeiro de 2016.

GAMBIM, Grazieli. Inteligência artificial: Utilização de jogos eletrônicos no campo educacional. In: vii colóquio internacional, 2013, São Luiz Gonzaga. VII colóquio internacional, 2013.

GROS, Begoña. The impact of digital games in education. First Monday, v. 8, n. 7, jul. 2003.

GRÜBEL, J. M. & BEZ, M. R. Jogos Educativos. Revista Novas Tecnologias na Educação, 4(1), 1-7, 2006.

IDC. International Data Corporation. Smartphone OS Market Share, 2015 Q2. 2015. Disponível no site: <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>. Acesso em 22 de fevereiro de 2016.

KENSKI, V. M. Novas Tecnologias, o redimensionamento do espaço e do tempo e os impactos no trabalho docente. 1998. Disponível em: http://www.usba.br/~prossiga/vani.htm. Acesso em: 02 dezembros 2012.

LAUDON, K.C.; LAUDON, J.P.. Sistemas de Informação Gerenciais. São Paulo: Pearson Prentice Hall, 2011. p. 114.

LEMLE, Miriam. Guia Teórico do Alfabetizador. Série Princípios. Editora Ática. 1988.

LENZ, Alexandre Rafael. Utilizando Técnicas de Aprendizado de Máquina para Apoiar o Teste de Regressão. Setor de Ciências Exatas. Universidade Federal do Paraná. Curitiba. Agosto de 2009.

LUCCHESE, F.; RIBEIRO, B. Conceituação de jogos digitais. 2009. São Paulo, 2009. Disponível em: <http://www.dca.fee.unicamp.br/˜martino/disciplinas/ia369/trabalhos/t1g3.pdf>. Acesso em: 31 de Maio de 2015.

MARÇAL, E, ANDRADE R., RIOS, R. Aprendizagem utilizando Dispositivos Móveis com Sistemas de Realidade Virtual. Revista Novas Tecnologias na Educação, 2005 - cinted.ufrgs.br. Disponível no site http://www.inf.ufes.br/~cvnascimento/artigos/a51 _realidadevirtual_revisado.pdf. Acesso 17/02/2016.

MARTINS, J. G.; MOCO, S. S.; MARTINS, A. R.; BARCIA, R. M. Realidade Virtual Através de Jogos na Educação**.** Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia da Produção. 2001.

MARTINS W, AFONSECA UR, NALINI LE, GOMES VM. Tutoriais inteligentes baseados em aprendizado por reforço: concepção, implementação e avaliação empírica. Anais do SBIE. 2007 Nov 1.

MAYER, Maximiliano. A história do Android. 2015. Disponível em: < https://www.oficinadanet.com.br/post/13939-a-historia-do-android >.

MERCATO, Mattia. A doce história do Android. 2015. Disponível em: < http://www.androidpit.com.br/historia-do-android>. Acesso em 22 de fevereiro de 2016.

MICROSOFT. Introdução à linguagem C# e ao .NET Framework. Disponível em: <https://msdn.microsoft.com/pt-br/library/z1zx9t92.aspx>. Acesso em 9 de janeiro de 2016.

MITCHELL, T. M. Machine Learning. McGraw-Hill. EUA. 1997.

MONARD MC, BARANAUSKAS JA. Conceitos sobre aprendizado de máquina. Sistemas Inteligentes-Fundamentos e Aplicações. 2003;1:1.

MONTEIRO. João Gabriel G. X. Propostas para o balanceamento dinâmico de jogos com ajuste de dificuldade e payoff lentos. Trabalho de Graduação. Centro de Informática. Universidade Federal de Pernambuco. Recife. 2009.

MONTEIRO, João Bosco. Google Android: Crie aplicações para celulares e tablets. Casa do Código. 2012.

MORAIS, Artur Gomes (Org.). O aprendizado da ortografia. Belo Horizonte: Autêntica, 2002.

MORATORI, Patrick Barbosa. Porque utilizar jogos educativos no processo de ensino aprendizagem? UFRJ. Rio de Janeiro, RJ. Dezembro 2003.

PASSOS EB, DA SILVA Jr JR, RIBEIRO FE, Mourão PT. Tutorial: Desenvolvimento de jogos com unity 3d. InVIII Brazilian Symposium on Games and Digital Entertainment 2009.

PEREIRA, Guilherme Vota. A Inteligência artificial aplicada na educação. Disponível em: <http://www.ceavi.udesc.br/arquivos/id_submenu/387/guilherme_vota_pereira.pdf>. Acesso em: 02 dez. 2012.

PERY, L. C.; NUNES, W. V.; CARDOSO, S. P.. Jogos Educativos Digitais: Ludicidade e Interatividade no Ensino nas Séries Iniciais. In: Congresso Iberoamericano de Informática Educativa IE 2010, 2010, Santiago de Chile. Memórias del Congreso Iberoamericano de Informática Educativa IE 2010. Santiago de Chile: Universidad de Chile; Facultad de Ciencias Físicas y Matemáticas.Departamento de Ciências de la Comp, 2010. v. 1. p. 107-113.

PORGOZELSKI, Kelli Damer; LIMA, Michelle Fernandes. Alfabetização: conceito e uma breve reflexão sobre a história da escrita. 2° Congresso Internacional de Educação de Ponta Grossa. Paraná. 2010.

POZZEBON, Eliane; Frigo; Bittencourt. Inteligência Artificial na Educação
Universitária: Quais as contribuições? Universidade Federal de Santa Catarina**.** Florianópolis, 2003.

PRETZ, Eduardo; RITZEL, Marcelo Iserhardt. Agentes de software para jogos educacionais. Centro Universitário Feevale. Disponível no site: http:// www.niee.ufrgs.br/eventos/CIIEE/2007/pdf/CP-312.pdf. Acesso 20/04/2015.

RIEDER, E. Zanelatto, J. Brancher. Observação e análise da aplicação de jogos educacionais bidimensionais em um ambiente aberto. Journal of Computer Science, vol 4, n. 2, páginas.63-71, 2005.

RUSSEL, Stuart J.; NORVIG, Peter. Inteligência Artificial. Editora Campus, Rio de Janeiro.

SACCOL, Amarolinda Zanela; REINHARD, Nicolau. Tecnologias de informação móveis, sem fio e ubíquas: definições, mapeamento do estado-da-arte e oportunidades de pesquisa. RAC, v. 11, n. 4, p. 175-198, 2007.

SANTOS, Vinicius Cordeiro. Aplicação do algoritmo SARSA no balanceamento dinâmico de dificuldade de um jogo digital ortográfico. Trabalho de conclusão de curso. Departamento de Computação. FACET. Universidade Federal dos Vales do Jequitinhonha e Mucuri. 2016.

SANTOS, Cícero Nogueira dos. Aprendizado de máquina na identificação de sintagmas nominais: o caso do português brasileiro. Rio de Janeiro, 2005. Disponível em: <https://www.google.com.br/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&cad=rja&uact=8&ved=0CC4QFjAC&url=http%3A%2F%2Fwww.linguateca.pt%2FRepositorio%2FDissertacaoCicero2005.pdf&ei=PNiIVcfAEdbLsASU-bLICg&usg=AFQjCNGJh6yE0YxXsjJdTSPeo2F9td5Xxw&bvm=bv.96339352,d.cWc> . Acesso em: 21, junho, 2015.

SARTINI, B.A, GARBUGIO, G, BORTOLOSSI, H.J, SANTOS, P.A, BARRETO, L.S.
Uma introdução a teoria dos jogos. II Bienal da SBM. Universidade Federal da Bahia, 25 a 29 outubro de 2004.

SAVI R; Ulbricht VR. Jogos digitais educacionais: benefícios e desafios. RENOTE. 2008.

SILVA, Maycon Prado Rocha. Jogos Digitais: definições, classificações e avaliação. Universidade Estadual de Campinas – UNICAMP. Campinas, setembro de 2009.

SILVA, C., Guedes, A. C., Barbosa, D. Proposta de um Processo de Desenvolvimento de Jogos Educativos. In: XXI Simpósio Brasileiro de Informática na Educação. João Pessoa, PB.

SILVA, F. Agentes Inteligentes em Jogos de Computador. Monografia. Instituto de Matemática e Estatística, Universidade de São Paulo, Brasil. 2005.

SILVA, Mirna Paula; SILVA, Victor do Nascimento; CHAIMOWICZ, Luiz. Dynamic Difficulty Adjustment Through an Adaptive AI. Departamento de Ciência da Computação. Instituto de Ciências Exatas. Universidade Federal de Minas Gerais. Brasil. 2015. Disponível em: <http://homepages.dcc.ufmg.br/~vnsilva/papers/147312_2.pdf>. Acesso em 13 de janeiro de 2016.

SOARES, M. Alfabetização e letramento. São Paulo: Contexto, 2003.

SOARES, Michele Dos Santos; FEIJO, Bruno. Design of 2d educational games of adventure using lua. Pontifícia Universidade Católica do Rio de Janeiro - PUC-Rio. Maxwell. 2013.

SQLITE. About. Disponível em: <www.sqlite.org/about.html>. Acesso em 9 de janeiro de 2016.

STRICKLAND, Jonatham. Como funciona o Android (Google Phone). Disponível em: < http://tecnologia.hsw.uol.com.br/google-phone.htm>. Acesso em 22 de fevereiro de 2016.

SUTTON, R. S.; BARTO, A. G. Reinforcement learning: an introduction. MIT Press. Cambridge, Massachusetts, EUA. 1998.

TANENBAUM, A. S., WOODHULL. Sistemas Operacionais: projeto e implementação. 2a. ed.. Porto Alegre: Bookman, 2000.

TAROUCO, L., 2004. Jogos Educacionais. Revista Novas Tecnologias na Educação, CINTED/UFRGS, vol.2 N.1, Março, 2004.

TONIN, Graziela Simone; GOLDMAN, Alfredo. Tendências em Computação Móvel. Departamento de Ciências da Computação do Instituto de Matemática e Estatística da Universidade de São Paulo, 2012.

TOZOUR, P. The Evolution of Game AI. In S. Rabin (ed.): AI Game Programming Wisdom. Charles River Media, Inc. 2002.

WATKINS, Christopher J.C.H., DAYAN, Peter. Q-Learning. Machine Learning, 8(3):279-292, 1992.

WEISZFLOG, W. Michaelis Moderno Dicionário Da Língua Portuguesa. [S.l.]: Melhoramentos, 2007.

APÊNDICE A – Conjunto de palavras do jogo *Laça Palavras*

Lista de palavras do nível 1 do jogo *Laça Palavras*

Palavras com “Ç” e “SS” com som de [s]:

- Engraçado;
- Dança;
- Assunto;
- Tosse;
- Assustar;
- Pássaro;
- Professora;
- Girassol;
- Osso;
- Assado.

Lista de palavras do nível 2 do jogo *Laça Palavras*

Palavras com “U” e “L” com som de [u]:

- Papel;
- Desagradável;
- Legal;
- Caracol;
- Anel;
- Álcool;
- Gol;
- Sol;
- Canal.

Palavras com “Ç” e “SS” com som de [s]:

- Passos.

Lista de palavras do nível 3 do jogo *Laça Palavras*

Palavras com “G” e “J” com som de [g]:

- Girafa;
- Hoje;

- Majestade;
- Mágica;
- Zoológico;
- Página;
- Gente;
- Jiló.

Palavras com “Ç” e “SS” com som de [s]:

- Poço.

Palavras com “U” e “L” com som de [u]:

- Sol.

Lista de palavras do nível 4 do jogo *Laça Palavras*

Palavras com “S” e “Z” com som de [z]:

- Treze;
- Blusa;
- Doze;
- Duzentos;
- Asa;
- Rosa;
- Lousa.

Palavras com “Ç” e “SS” com som de [s]:

- Assoprar.

Palavras com “U” e “L” com som de [u]:

- Álcool.

Palavras com “G” e “J” com som de [g]:

- Página.