FACULDADE DE TECNOLOGIA, CIÊNCIAS E EDUCAÇÃO

Graduação

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Parser inteligente para utilização em jogos do estilo *text adventure*

Nayara Ghiotto da Silva
Prof. Dr. Maurício Acconcia Dias (Orientadora)

RESUMO

Em jogos eletrônicos atuais é comum observarmos os gráficos e a jogabilidade como principais atributos, no entanto o gênero *text adventure* propõe o foco em seus enredos e *puzzles,* possuindo interfaces simples que em geral, baseiam-se apenas em comandos textuais. A interação com o jogador neste gênero é realizada por meio de um analisador textual (interpretador de comandos), que realiza a interpretação do comando digitado e retorna a ele as próximas coordenadas. No entanto, a comunicação proporcionada pelos analisadores requer que os usuários tenham conhecimento das palavras aceitas no jogo, visto que os mesmos possuem baixa variedade de comandos válidos, o que influencia tanto na jogabilidade, quanto na interação do jogador com a história. O presente artigo propõe o desenvolvimento de um *parser* inteligente que interprete comandos inseridos pelo usuário, realizando a busca otimizada em sua base de dados, de modo a disponibilizar a opção de inserção de novas palavras (caso a mesma não exista na base) sem um limite máximo estipulado. O *parser* dispõe-se a suprimir limitações geralmente contidas nos analisadores existentes, fato esse que tenciona o jogador a conhecer as palavras-base e/ou inserir comandos pré-definidos do jogo (ex: ir para o norte). Contudo, ao final da pesquisa o mesmo será aplicado a um jogo real, proporcionando maior interatividade / imersão entre jogador e jogo e disponibilizado ao público em geral.

Palavras-chave: *Text adventure*. *Parser.* Jogo.

ABSTRACT

In the current eletronic games is common we see graphics and jogability as principal atributs, however the genre text adventure proposes the focus on it’s

plot and puzzles, having simple interface that in general, is based in texts commands. The interaction with player in this genre is performed by an textual analyzer (command interpreter), that performs the interpretation of the typed command and returns to him the next coordinates. However, communication provided by the analyzers, requires that the users have knowledge of the accept words in the game, since they have low variety of valid commands which influences both the gameplay as player interaction with history. The present paper proposes development of an intelligent parser that interprets commands entered by the user doing the optimized search in it's data base, in order to provide the option to insert new words (if the word do not exist) without a stipulated maximum limit. The parser is available to remove limitation usually contained in existing analyzers, it makes the player to know base words and insert pre-defined game commands (ex. go north). However, at the end of the search, parser will be applied to a real game, providing greater interactivity/immersion between player and game and available to the general public

Keywords: *Text Adventure. Parser.* Game**.**

Introdução

Atualmente no cenário tecnológico de games, têm se evidenciado um maior foco em jogos 3D devido à diversão em decorrência da realidade paralela estabelecida com o jogador, o que ocasionou menor demanda na criação e utilização de jogos do estilo *text adventure* (nome dado aos jogos em modo texto), onde todo o enredo se passa de forma textual (VIEIRA, 2014).

Jogos deste gênero não possuem gráficos como seu principal atributo, mas sim a resolução de seus quebra-cabeças (*puzzles*) e enigmas ao decorrer das fases, bem como a exploração natural do mapa contido em seu enredo. Embora sejam imersos em seu contexto, alguns jogos pertencentes a esse formato não são puramente em modo texto, possuindo "fases" em que o jogador consiga visualizar o gráfico propriamente dito do game em 3D sendo, portanto, uma categoria mista.

Os jogos em modo texto acontecem com o direcionamento obtido pelos comandos digitados pelo jogador. Técnicas de Processamento de Linguagem Natural (PLN) e Mineração de Dados auxiliam nessa etapa aceitando, interpretando e direcionando o comando inserido. Segundo Moraes (2007) são técnicas de análise e extração de dados a partir de textos, palavras ou frases, que envolvem a aplicação de algoritmos computacionais que processam textos

e identificam informações úteis e implícitas. Contudo, o jogo modo texto também possui uma dimensão física que inclui o jogador no contexto do mapa, muitas vezes caracterizados como espaços ilógicos, onde a cada escolha tomada, uma “área” é bloqueada, criando um labirinto e induzindo o jogador a criar seu próprio mapa. Este modo de jogo acarreta no estímulo da imaginação e do raciocínio.

Considerando o fator primordial no funcionamento da ferramenta para um bom desempenho na prática, se faz necessário que o desenvolvimento esteja juntamente conectado ao mapa (ambiente), de modo a realizar a interação do interpretador de texto com o jogo em si.

O desenvolvimento da ferramenta proposta possibilitará não somente a interpretação de palavras já pré-definidas, mas também novos comandos que serão processados e armazenados na mesma por meio da busca otimizada em seu banco de dados.

O desenvolvimento da ferramenta e o alcance do resultado esperado têm como base o estudo aprofundado em jogos *text adventure*, técnicas de PLN e na análise detalhada das deficiências e pontos de melhorias, juntamente com suas respectivas propostas de solução. Como base inicial, foram pré-definidas palavras chaves a serem utilizadas pelos jogadores, equivalente à técnica de mineração de dados, para que a ferramenta possa assimilar, relacionar, interpretar e distinguir palavras. Segundo Aranha (2006) técnicas de mineração de dados geralmente são muito úteis atuando em bases de dados organizados, podendo identificar os conhecimentos relevantes da base textual. Sendo assim, tais técnicas serão empregadas para análise dos resultados finais.

Atualmente os jogos vêm ganhando espaço no mercado tecnológico devido ao grande avanço obtido e sua grande demanda. Embora todos sejam caracterizados como entretenimento, um game pode ser desenvolvido para diversas finalidades, para diversos públicos e possuem diversos gêneros. A presente pesquisa é voltada para o gênero textual, conhecidos como *text adventure.*

Com o surgimento dos games em 3D, houve temor quanto ao possível fim dos jogos do gênero de aventura textual, no entanto podemos notar que

ainda possuem lançamentos cujo foco se dá aos *puzzles*, a exploração do cenário e ao estímulo do raciocínio lógico, características eminentes do gênero.

Os *parsers* (analisadores textuais) contidos nos *text adventures* possuem níveis distintos de sofisticação, diferente das primeiras ferramentas criadas que aceitavam apenas verbos como comando, atualmente o nível de entendimento remetem-se desde curtas palavras, até frases completas. Em função das funcionalidades específicas do analisador textual, foi optado pelo desenvolvimento em linguagem de programação *Csharp*, aplicada à ferramenta de criação de jogos *Unity*.

O *Game Engine* (motor de Jogo) Unity é uma ferramenta para desenvolvimento de jogos em duas ou três dimensões (BERGAMASCHI*, 2014*), destacada pela simplicidade em seu estilo de programação e organização de projetos, além da facilidade de aprendizado devido à interface intuitiva contida e sua ampla série de tutoriais disponibilizados pela empresa desenvolvedora *Unity Technologies.* O *engine* possui sua versão *Pro* (paga) *e Free* (gratuita),ambas permitindo a reutilização de elementos já prontos como personagens,texturas, animações, gráficos e sons, por exemplo, disponíveis na *Asset Store* do *site* oficial, diminuindo assim o custo e o tempo de desenvolvimento *(*COWAN; KAPRALOS, 2014).

A *Unity* permite que o *game* desenvolvido seja aplicado a diversas plataformas, como *Windows / Windows Phone, Android*, *IOS*, *Linux, Xbox* e *BlackBerry,* sendo programável nas linguagens C# *(Csharp)*, *JavaScript* e *Boo*.

Para o presente projeto foi optada pela utilização do C#, uma linguagem de alto nível orientada a objetos, criada pela *Microsoft* para serem executadas nas plataformas *.NET Framework*. Tal linguagem permite maior controle dos objetos e estruturas do projeto, além de nos fornecer recursos de enumerações, valor anulável e acesso direto à memória. Deste modo, a aplicação do C# na *Unity* visa a melhoria da jogabilidade dos *text adventures*, visando facilitar não somente o aprendizado do jogador dessa modalidade, mas também do analisador contido no jogo.

O objetivo principal do projeto de pesquisa apresentado é o desenvolvimento de uma ferramenta de análise textual (*parser*) inteligente, para ser utilizada em jogos do gênero *text adventure*. Por inteligente entende-se que exista alguma vantagem do *parser* proposto neste trabalho com relação

à simples tradução dos comandos digitados pelo usuário como, por exemplo, a interpretação de novos comandos ou também uma busca eficiente em um banco de comandos. Para alcançar o objetivo principal deste projeto, alguns objetivos específicos podem ser definidos:

- **Obtenção de características comuns em jogos *text adventure***: a pesquisa sobre os jogos deste gênero, já existentes, tem como objetivo apontar as características principais destes jogos para auxiliar no desenvolvimento do projeto.

- **Escolha da técnica de PLN a ser utilizada**: um estudo aprofundado em técnicas PLN e mineração de dados, concentrado nas técnicas para o auxílio e compreensão de linguagens humanas, irá auxiliar na escolha das melhores técnicas a serem empregadas no desenvolvimento deste projeto

- **Desenvolvimento do *parser* em linguagem de programação C**: o desenvolvimento do objetivo proposto pode ser realizado por diferentes linguagens de programação, para o presente artigo será utilizada a linguagem C, por ser uma alternativa interessante devido à sua facilidade de implementação e portabilidade para diferentes *engines* e/ou ferramentas de desenvolvimento. Este objetivo será obtido através do estudo da linguagem.
- **Implementação da ferramenta em um jogo final**: ao final da pesquisa a ferramenta desenvolvida será integrada a um jogo real que está em desenvolvimento no grupo de desenvolvimento de jogos da Fatece (Faculdade de Tecnologia, Ciências e Educação) e será disponibilizado para o público.

Estima-se que, ao final deste projeto de pesquisa, o software de análise textual resultante apresente bom desempenho quando integrado a um jogo do gênero *text adventure*. Ao desenvolver esta ferramenta utilizando inteligência artificial, espera-se também o aumento gradativo da capacidade de aprendizado da ferramenta, baseado nas técnicas escolhidas.

Este artigo divide-se em sete seções, onde a primeira aborda a introdução ao tema e os objetivos propostos, conforme apresentado acima. A segunda seção apresenta o estudo e a análise baseado em jogos do gênero *text adventure,* enfatizando as necessidades básicas e específicas do *parser,* como por exemplo, a utilização do processamento de linguagem natural aplicado a ferramenta. Logo, a terceira seção refere-se ao estudo das técnicas de processamento de linguagem natural, juntamente da apresentação da abordagem utilizada na presente pesquisa. A seção quatro realiza a apresentação de trabalhos relacionados à proposta do presente artigo, objetivando a demonstração da ampla aplicação de analisadores textuais e a relevância do mesmo no contexto atual, já a quinta seção demonstra a metodologia e as ferramentas utilizadas para o desenvolvimento do analisador textual, bem como as etapas da implementação para a criação da ferramenta final. Por fim, as seções seis e sete constituem a definição das técnicas de testes e validação da ferramenta, bem como os resultados obtidos e sua consideração final.

1 Histórico dos jogos *text adventures*

Os jogos de aventura (*adventures*) são caracterizados pela ênfase em seu enredo, deste modo os *text adventures* são jogos textuais com histórias interativas que induzem o jogador como protagonista e estimulam o raciocínio lógico para a resolução dos *puzzles* apresentados ao decorrer das fases. Os quebra-cabeças (*puzzles*) não possuem um padrão, ocorrendo variações entre resolver charadas, explorar labirintos e mover peças, que usualmente são necessários para o avanço no jogo, visto que não há como progredir sem a resolução dos mesmos (VALENTE, 2005).

Devido ao grande enfoque textual, a narrativa aplicada é responsável pelo direcionamento do jogador no cenário, no estímulo à exploração dos ambientes e pela demonstração quanto ao contexto histórico do jogo. Vinculando os fatores essenciais do gênero, como exploração e resolução dos enigmas. Portanto, pode-se afirmar que a construção da história em um jogo se dá não somente a partir de decisões tomadas em seu enredo, mas também com base na lógica predefinida pelo programa (OLIVEIRA, 2009).

Segundo JERZ (2001) os *text adventures* surgiram na década de 1970, com o lançamento do *Colossal Cave Adventure* em 1976, criado pelo programador William Crowther. A capacidade de entretenimento apresentado deu origem a novos sucessos do gênero como *Planetfall*,*Zork I, II e III,Spider and Web,The Hitchhiker's Guide to the Galaxy, Cypher - Cyberpunk Text Adventure,* entre outros que serão apresentados abaixo:

- ***Colossal Cave Adventure:***[1] A produção em modo texto tem como objetivo a exploração de uma caverna em busca de seus tesouros e elementos sobrenaturais. Por meio de comandos é possível tomar decisões, escolher caminhos e escapar de mortes inesperadas.

- ***Planetfall:***[2] Lançado em 1983, a aventura se passa em uma nave estelar que cai em um planeta deserto e então, busca-se desvendar os mistérios do local. O jogo possui inúmeros objetos utilidade não é descrita, além de *puzzles* complexos que contemplam a jogabilidade empregada.

- ***Zork:***[3] Considerado um dos primeiros jogos de ficção interativa, foi desenvolvido baseando no famoso *Colossal Cave Adventure* e lançado pela primeira vez em 1979. O jogo dividido em três partes ficou conhecido pela otimização de seu analisador de texto, que permitia não somente a entrada de comandos simples, mas também conjunções em frases mais sofisticadas.

- ***Spiderand Web:***[4] A ficção interativa de Andrew Plotkin representa a história de um espião que tem por objetivo desvendar séries de desafios embutidos na obra. O jogo recebeu prêmios de melhor jogo, melhores *puzzles,* melhor NPC (Non-player Character) individual e melhor uso de meios no *Xyzzy Awards* em 1998.

[1]*Colossal Cave Adventure*:
[2]*Planetfall:*http://www.infocom-if.org/games/games.html
[3]*Zork:*http://textadventures.co.uk/games/view/5zyoqrsugeopel3ffhz_vq/zork
[4]*Spider and Web:*
http://textadventures.co.uk/games/view/qski30iszearnimjkrzvag/spider-and-web

- ***The Hitchhiker's Guide to the Galaxy:***[5] A aventura foi projetada e lançada em 1984 baseando-se na série de ficção científica “O guia do mochileiro das galáxias” da época. Embora tenha obtido grande sucesso, houve duras críticas quanto à impossibilidade de sobrevivência ao final do jogo, caso não seja retirado e armazenados objetos pré-definidos e não especificados ao decorrer das fases.

- ***Cypher - Cyberpunk Text Adventure:***[6] O jogo criado em 2012 é uma adaptação do gênero, cujo desenvolvimento abrange efeitos gráficos e sonoros o que aumenta a imersão do jogador. Foram realizadas correções após o lançamento, de modo a melhorar as restrições do analisador de texto e corrigir erros de gramática contidos na produção.

É notório o avanço dos recursos contidos nos analisadores textuais ao longo dos anos, no entanto no auge do gênero eram considerados limitados, aceitando apenas palavras pré-definidas ou conjunções simples. Com a ascensão dos consoles, os *text adventures* perderam espaço no mercado de jogos, desencadeadas pelo desenvolvimento de novas tecnologias, onde foram aplicadas animações gráficas, que proporcionavam maior interação jogador/máquina e maior facilidade em sua jogabilidade, entretanto atualmente têm se evidenciado o retorno do gênero, com lançamentos como *Lifeline, Sun Dogs, A Dark Room, London Fallen e 80 Days*, por exemplo.

- ***Lifeline:***[7] Disponível para Android e IOS, o jogo desenvolvido pela *Three Minute Games* em 2016, tem como objetivo auxiliar na sobrevivência de um astronauta (Taylor) preso em uma lua desconhecida. O jogador torna-se então orientador de Taylor para seu retorno a um lugar seguro, onde cada decisão tomada influência diretamente seu destino. A aventura é disponível nas bibliotecas de aplicativos *Play Store* e *Apple Store.*

[5]The Hitchhiker’s Guide to the Galaxy: http://www.infocom-if.org/games/games.html
[6]Cypher – Cyberpunk Text Adventure: http://www.cabrerabrothers.com/cypher.html
[7]Lifeline:http://lifelinegame.wikia.com/wiki/Lifeline_(Series)

- ***Sun Dogs:***[8] A aventura de texto lançada em 2015, têm como intuito a exploração do sistema solar, num cenário onde a humanidade já habita dentre os planetas, luas e estações espaciais. Proporcionando um enredo fascinante contendo asteróides congelados, mega cidades em Marte e até *gangsters* espaciais, visa-se a exploração e resolução de missões ao decorrer da história. O jogo disponibiliza a função de *modding* (modificação) para os usuários, de modo a permitir alterações e adições em seu enredo.

- ***A Dark Room:***[9] O jogo do gênero Ficção Científica é passado inteiramente em modo texto, e tem como objetivo a sobrevivência em um cenário pós-apocalipse a partir da exploração em um quarto escuro. A medida em que o jogo avança, a capacidade de recolher objetos, realizar construções, armadilhas e conversação com estranhos são disponibilizadas ao jogador, de modo a auxiliar na resolução dos *puzzles* inseridos ao decorrer da história. Seu primeiro lançamento ocorreu em Junho de 2013, com sua versão criada para navegadores. Logo após, no final do mesmo foi disponibilizado para o *IOS*, e somente em 2016 teve sua versão adaptada para *Android.*

- ***London Fallen:***[10] Lançado em 2009, o jogo te permite a escolha de sua identidade e habilidades para sua sobrevivência nas escuras e perigosas ruas desse mundo interativo, onde o jogador assume o papel de um novo morador em um sub-mundo e irão explorar a cultura, bem como as atividades permitidas ou não na cidade. O diferencial de *London Fallen* são os pesos atribuídos às escolhas realizadas, elas podem ajudar no decorrer do jogo ou exilar personagens. O game está disponível online pelos navegadores e através de *download* para versões *Android* e *IOS.*

[8]*Sun Dogs:http://www.sundogsgame.com/*
[9]*A Dark Room: http://adarkroom.doublespeakgames.com/*
[10]*London Fallen:http://fallenlondon.storynexus.com/*

- ***80 days:***[11] Criado baseado na novela *Jules Verne, Around the World in 80 Days*, o jogo é categorizado como ficção interativa, e tem como objetivo dar a volta ao mundo em 80 dias. A aventura proporciona ao jogador diversos itens valiosos e arriscados ao longo das fases, onde é necessário tomar decisões, de modo a alterar o destino do personagem. Disponível para *Windows, Android e IOS.*

A evolução consiste não somente nas melhorias aplicadas aos enredos, *puzzles* e analisadores, mas também nas plataformas finais, onde os jogos serão disponibilizados. A aplicação em navegadores, consoles e dispositivos móveis proporciona um alcance maior de usuários, em decorrência de sua comodidade e facilidade de acesso.

Os jogos do gênero *text adventure* precisam de uma ferramenta de interpretação de linguagem para funcionar como interface entre o jogador e o jogo. Estas ferramentas utilizam técnicas de uma área da computação chamada Processamento de Linguagem Natural (PLN) que será apresentada a seguir.

2 Processamento de linguagem natural

As técnicas PLN referem-se a um campo da computação que combina tecnologia, inteligência artificial e lingüística computacional de modo a proporcionar a comunicação entre máquina e usuário por meio de uma linguagem natural, Filho (2009). Contudo, visa-se a interpretação de um texto cujo conteúdo extraído sirva de base para aplicações concretas, Ovchinnikova (2012).

Desenvolvida em 1950, com a publicação do artigo "*Computing Machinery and Intelligence*" de Alan Turing, a PLN tem se tornado uma questão importante devido a sua ampla utilização ao longo do tempo, sendo aplicada atualmente em atividades como Máquina de Tradução, *Chatbots*, Resumo Automático, Agentes de conversação, Sistemas de busca na web, Corretores

[11] 80 days: https://www.inklestudios.com/80days/

Ortográficos, entre muitos outros, que possibilitam maior interatividade com os usuários e auxiliam no aumento das pesquisas da área.

Embora seja uma área em expansão, as aplicações de PLN têm seu desempenho prejudicado pela incompletude dos recursos linguísticos utilizados, a ampla variação morfológica e sintática das sentenças, pela complexidade das tarefas específicas do processamento e principalmente pelo grande volume de ambiguidade presente na linguagem natural humana, Silva (2003). Para tanto, realiza-se a utilização de aspectos de linguagens como sintaxe, semântica, pragmática, morfologia e fonologia, de modo a indicar aspectos da linguagem natural à máquina.

2.1 Análise sintática

O analisador sintático (*parser*) realiza a organização de palavras contidas em uma frase, estruturando e combinando regras gramaticais de modo a gerar árvores de derivação da estrutura sintática da sentença analisada, Gonzalez e Lima (2003). Contudo, a análise sintática trata também a presença de ambiguidades em frases, a fim de obter todas as possíveis estruturas que a representam e viabilizar o processamento semântico.

2.2 Análise semântica

Realiza a associação e combinação de significados à sintática das frases inseridas, não somente em palavras individuais, mas também no conjunto resultante delas. O analisador semântico visa a análise do sentido das palavras reagrupadas pelo analisador sintático, uma vez que o analisador morfológico permitiu identificar estas palavras individualmente, Neto, Tonin e Prietch, (2010).

2.3 Pragmática

Adequa as sentenças inseridas em diversas situações, atribuindo significado ao contexto em que ela é aplicada, podendo ser associadas por meio de relações paradigmáticas e sintagmáticas, onde as paradigmáticas

associam de acordo com seu significado e as sintagmáticas por serem frequentemente encontradas em uma mesma frase.

2.4 Morfologia

A análise morfológica realiza a identificação de palavras isoladas em uma sentença, dividindo-as em unidades significativas / classes gramaticais, de acordo com seus delimitadores (espaços e pontuações), Oliveira (2002). Deste modo, a morfologia classifica as palavras das sentenças com base em sua linguagem natural e sua categoria gramatical.

Segundo Oliveira (2002), a importância do analisador morfológico se dá para a compreensão da frase, de modo a atribuir significado a cada uma das palavras componentes.

2.5 Fonologia

A fonologia realiza o estudo dos sistemas de sons (fonemas) de uma determinada língua, a fim de investigar o conhecimento fonológico dos falantes. Tem-se por objetivo o reconhecimento da fala de acordo com as ondas sonoras emitas pelo falante, de modo a proporcionar a interação usuário/máquina.

> O reconhecimento de fala envolve a interpretação de ondas sonoras e a associação destas com elementos de fala, podendo reconhecer somente palavras isoladas dentro de um léxico pré-determinado ou reconhecer fala contínua de uma determinada língua (FILHO, 2009).

Deste modo, os analisadores buscam a representação do significado de uma palavra em uma sentença obtida por meio de sua forma lógica, que codifica todos seus possíveis sentidos e relacionamentos semânticos, determinando assim a viabilidade das palavras inseridas e seus respectivos significados, Gonzalez e Lima (2003).

Atualmente o processamento de linguagem natural tem se baseado em técnicas de *Machine Learning* (Aprendizado de máquina), cuja finalidade visa a análise dos dados obtidos e o aprendizado automático do *parser* aplicado ao jogo, por meio da implementação de algoritmos de aprendizagem interativos. Tais implementações são empregadas não somente para obter um melhor

desempenho do analisador textual, mas também para o tratamento de limitações (sinônimos, ambiguidades) impostos a ele.

Neste projeto, para a construção de um interpretador de comandos para jogos do gênero *text adventure,* será utilizada a técnica de morfologia onde as palavras que indicam comandos serão isoladas das frases e traduzidos para comandos que possam ser executados no jogo. A adição de procedimentos inteligentes ao processo irá facilitar e proporcionar a comunicação do jogador com o jogo por meio de sua interface.

A próxima seção apresenta trabalhos relacionados ao desenvolvimento de *parsers* e suas diferentes aplicações.

3 Trabalhos Relacionados

Atualmente existem diversas aplicações de *parsers* em programas e/ou ferramentas, cuja finalidade varia entre entretenimento, aplicações educacionais e até mesmo na área da saúde. Desta forma, abaixo são apresentados pesquisas onde analisadores textuais são empregados de formas distintas.

Maziero, Pardo e Nunes (2007) abordam em sua pesquisa a aplicação de um método de segmento baseado em informações morfológicas decorrentes do analisador *parser* Palavras[12], por meio da utilização do método dissertativo RST (*Rhetorical Structure Theory*) de modo a incorporá-lo ao sistema DiZer(*Discourse analyzer for BrazilianPortuguese*) objetivando um melhor desempenho do analisador retórico automático.

O trabalho desenvolvido por Miranda (2012) apresenta a proposta de utilização do FIGN, um conjunto de ferramentas de *software* criado para o desenvolvimento de jogos FI (Ficção Interativa) com a funcionalidade de um gestor de narrativas integrado. A ferramenta objetiva o desenvolvimento de jogos do gênero de forma facilitada, sem a necessidade de conhecimento aprofundado em programação e possibilita que o desenvolvedor direcione o gestor de narrativas (Função de sugestões, apresentação do enredo da história, negar acesso às zonas do mapa e atribuir itens, por exemplo), aos

[12] Palavras: http://visl.sdu.dk/~eckhard/pdf/PLP20-amilo.ps.pdf

pontos de enredo, de modo a auxiliá-lo no controle de processo do jogador ao decorrer do jogo.

Em pesquisa realizada por Souza (2015) é proposto o desenvolvimento de um ambiente computacional para leitura e escrita de livros-jogos aplicados a dispositivos móveis. Em seu trabalho relata-se o ressurgimento do gênero textual em formato de “livros” decorrentes das novas tecnologias como *smartphones* e *tablets*.

Ainda Souza (2015), apresenta-se a utilização de *text adventures* voltado a preenchimento de formulários, onde os mesmos podem ser disponíveis em *browers,* possibilitando a análise de desempenho da obra. Contudo, são descritas ferramentas de preenchimento de formulários como *Gamebook Maker*, a fim de analisar o método mais eficaz para a criação de um livro-jogo de qualidade.

No trabalho de Oliveira, Pozzebon e Frigo (2017), aborda-se a utilização do *parser* voltado a educação por meio da análise do jogo *Scribblenauts*, onde são disponibilizadas 22.802 palavras ao jogador, de modo a serem utilizadas para interação com o jogo. A aplicação e escolha das palavras usadas ao decorrer das fases possibilitam a criação de objetos/personagens para a resolução dos *puzzles* existentes, estimulando não somente a criatividade, mas também o desenvolvimento estratégico e o raciocínio lógico do jogador.

Na abordagem de Almeida (2017), tem-se como objetivo a implementação de um *parser* multilíngue, visando o auxílio e a solução de problemas manifestados por planejadores automáticos, cuja ênfase se faz aos utilizadores de linguagens como PDDL (*Planning Domain DefinitionLanguage*), STRIPS (*Stanford Research Institute Problem Solver*) e ADL (*Action Description Language*), de modo a identificar as distinções, semelhanças, melhorias e pontos de destaque dos modelos atuais.

Com base nos presentes trabalhos apresentados, torna-se visível a ampla aplicação de *parser s*em programas, ferramentas e iniciativas que visam a melhoria do recurso, de modo a aplicá-los em áreas diversas. Este projeto então propõe a implementação de uma ferramenta para a interpretação de comandos que pode ser utilizada em jogos do tipo *text adventure* e utilizou o método e as ferramentas apresentadas a seguir.

4 Métodos e Ferramentas

O estudo aprofundado dos jogos *text adventures* proporciona não somente a obtenção de características básicas do gênero, mas também evidencia os pontos primordiais de seus analisadores textuais, como por exemplo, a aplicação do processamento de linguagem natural ao *parser* para a análise e interpretação de comandos digitados pelo usuário. Baseado no estudo, a escolha da técnica apropriada para o desenvolvimento do analisador proposto foi feita utilizando-se a análise morfológica, que consiste na interpretação de palavras isoladas em uma sentença e as convertem em comandos traduzíveis ao jogo.

A construção do analisador se deu através da utilização da análise dos requisitos inicialmente definidos, objetivando o desenvolvimento de uma ferramenta inteligente que interprete comandos inseridos e também armazene os comandos desconhecidos em um banco de dados, realizando uma busca otimizada das palavras em sua base de dados quando necessário. Para tanto, foi necessária a criação de um arquivo para armazenamento de palavras, visando não somente o arquivamento dos novos comandos, como também a facilitação de sua busca posteriormente. Com tudo, juntamente da pesquisa citada anteriormente foram implementados métodos de ordenação e pesquisa de dados, de modo a analisar o melhor método para o objetivo aqui proposto.

No início considerou-se a utilização da linguagem de programação *Csharp* (C#) devido à disponibilidade de controle de objetos e estruturas do projeto do jogo que está sendo realizado pelo grupo de jogos Sofa87-96 [13]da Fatece, onde será aplicado o *parser* desenvolvido neste trabalho. A linguagem permite a integração ao *Game Engine Unity* que é a ferramenta utilizada pelos desenvolvedores do jogo. Porém, um software desenvolvido em linguagem C# possui uma baixa portabilidade para outras *engines* e ferramentas de desenvolvimento, o que não ocorre com a linguagem C. Neste caso a mudança foi justificada por este fato, sendo este trabalho desenvolvido em linguagem C, utilizando a IDE *Code::Block*s versão 16.01, e o compilador MingW / GCC versão 4.9.2, vale ressaltar que a IDE permite o desenvolvimento em diversas

[13] Sofa87-96: http://jogos.fatece.edu.br/index.html

linguagens de programação, como C/C++, *Python*, CSS, *Java/JavaScript*, *Fortran*, XML, PHP, HTML, e encontra-se disponível de forma gratuita para *Windows*, *Linux* e *MacOS*.

Ao fim da implementação, a fase de teste se faz necessária para garantir que a ferramenta funcione corretamente, abrangendo todas as suas especificações. Deste modo, o *parser* foi disponibilizado aos membros do grupo de desenvolvimento de jogos da Fatece, no qual foi submetido aos testes de inserção de novas palavras, busca e reconhecimento de palavras e a verificação do tempo de resposta para a busca realizada no banco de dados. Baseando-se nas etapas citadas, a seção abaixo irá retratar os resultados obtidos.

A metodologia aplicada no presente artigo consiste nas etapas exemplificadas na figura abaixo:

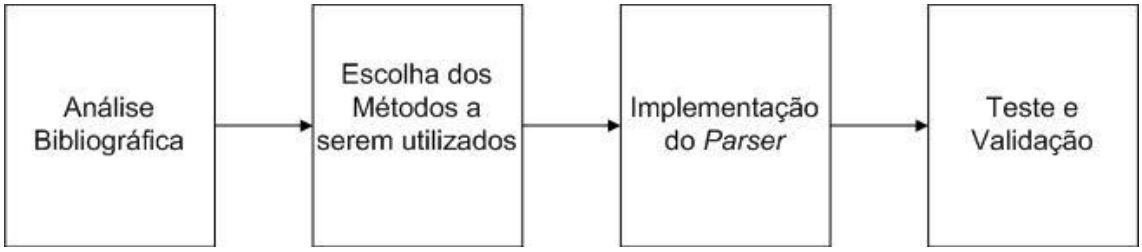

Figura 1. Metodologia de desenvolvimento do trabalho
Fonte: Elaborado pelo autor

Abaixo serão apresentados os resultados obtidos pelo *parser* desenvolvido.

5 Resultados

5.1 Requisitos

O planejamento e a documentação de um projeto têm como finalidade a obtenção de resultados condizentes com os esperados, logo, a definição dos requisitos funcionais e não-funcionais aqui presentes têm como objetivo a especificação das funcionalidades contidas no *parser* desenvolvido.

5.1.1 Requisitos funcionais

Tabela 1. Definição dos requisitos funcionais do *parser*

RF01- O *parser* deve realizar a busca em sua base dados para análise da palavra inserida, juntamente a seu significado.
RF02 - O *parser* deve permitir que o usuário insira uma nova palavra em seu banco.
RF03 - Ao inserir uma nova palavra em sua base, o *parser* deve solicitar o significado da palavra, considerando uma já existente.
RF04 - O *parser* não deve conter um limite máximo de palavras em sua base de dados.
RF05 - O *parser* deve retornar a confirmação de inserção da palavra na base de dados.
RF06 - O *parser* deve retornar uma mensagem ao usuário em caso de erro ao acessar o banco de palavras.
RF07 - O *parser* deve retornar uma mensagem ao usuário em caso de erro na gravação da palavra no banco de dados.
RF08 - O *parser* deve retornar uma mensagem ao usuário em caso de erro ao fechar o banco de dados.
RF09 - O *parser* não deve solicitar a inserção de uma palavra já contida em sua base.
RF10 - Ao ser inserida uma palavra que não seja pré-definida, porém já existente no banco de dados, o *parser* deve reconhecê-la automaticamente.

Fonte: Elaborado pelo autor

5.1.2 Requisitos não-funcionais

Tabela 2 - Definição dos requisitos não-funcionais do *parser*

RF01- O analisador textual deve obter um tempo de retorno para a busca da palavra menor que 2 minutos.
RF02 - O *parser* implementado deve rodar em sistemas operacionais como Windows.
RF03 - O *parser* deve conter uma cópia de sua base de dados para backup.
RF04 - O *parser* deve conter portabilidade para outros sistemas operacionais.

RF05 - O *parser* deve conter portabilidade para IDEs diversas, como por exemplo, o *Game Engine Unity*.
RF06 - A utilização do *parser* deve ser de fácil utilização ao usuário.

Fonte: Elaborado pelo autor

5.2 Implementação

A implementação do *parser* foi feita em linguagem C utilizando a IDE Code::Blocks com o compilador MinGW. A ideia foi criar a ferramenta de modo que seja facilmente modificável para ser incluída no desenvolvimento de jogos. O fluxograma da Figura 2 descreve o funcionamento do programa.

Como estrutura básica o programa utiliza um banco de dados simples feito pela manipulação de um arquivo chamado “banco.dat”. Este arquivo contém registros (structs) que são compostos por duas strings, uma que armazena o comando desconhecido e outra que armazena o comando conhecido equivalente.

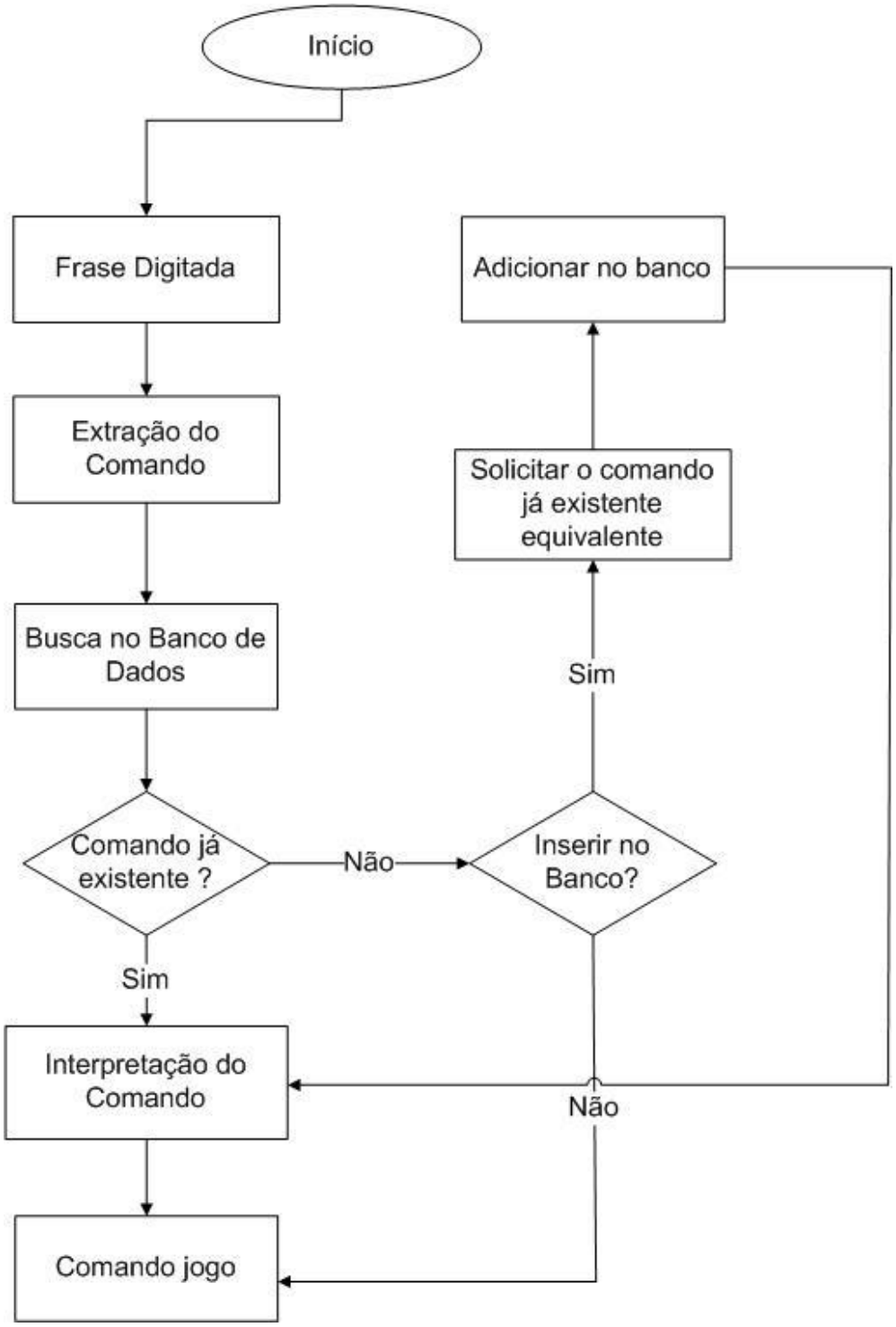

Figura 2. Fluxograma do analisador textual desenvolvido
Fonte: Elaborado pelo autor

Assim que o programa é iniciado todas as palavras armazenadas no banco de dados são carregadas para um vetor e então o vetor é trabalhado no software. Optou-se por este modelo para que a manipulação dos dados ocorresse obrigatoriamente em memória primária diminuindo o tempo de

acesso dos dados e também excluindo a necessidade de utilização de algoritmos para memória secundária.

Foi também implementada a possibilidade de se limpar o banco de dados para que os comandos armazenados fossem “esquecidos”. A remoção de comandos é feita na estrutura e depois os dados são gravados no banco de dados.

A busca pela palavra inicia-se com a separação da frase inserida pelo usuário em *“tokens”* da mesma forma que ocorre na primeira fase de *“parsers”* de compiladores. Cada token é então analisado por uma função que faz a busca das palavras inicialmente em uma lista de comandos conhecidos de forma sequencial. Neste caso a busca sequencial é a melhor alternativa por se tratar de uma lista pequena que não tem necessidade de ser ordenada ou mesmo buscada de forma inteligente.

Caso a palavra não seja encontrada nesta primeira busca, inicia-se a busca em um vetor que contém os dados do banco de dados de comandos. As comparações e operações com *strings* foram todas desenvolvidas com funções da biblioteca *strings.h* por apresentarem um bom desempenho. Caso a palavra seja encontrada, a função é novamente chamada para o comando equivalente que já está armazenado na struct. Caso não seja encontrada, a opção de inclusão do comando é iniciada.

O processo de se incluir o comando novo é simples. Inicialmente para cada palavra o software pergunta se o usuário deseja incluir como um novo comando. Em caso positivo, a palavra é armazenada no vetor em uma *struct* e, em seguida, é perguntado ao usuário qual comando ele deseja associar àquela palavra. Assim que o usuário fornece o comando a nova equivalência é armazenada no banco de dados e pode ser utilizada dali em diante como comando.

5.3 Testes

Os testes realizados foram com objetivo de verificar se o *parser* era realmente capaz de armazenar as palavras novas e fazer a busca pelos comandos sem problemas. Foi feita uma estrutura de tamanho 1000 onde palavras foram inseridas e depois buscadas. A inserção para número grandes

de palavras foi feita aleatoriamente para permitir que os testes fossem realizados no tempo para entrega deste trabalho.

Duas são as principais questões deste *parser,* a primeira é qual a palavra que será testada como comando. Como sabemos, nem todas as palavras da língua portuguesa serão possíveis comandos. Para isso seria possível eliminar alguns tipos de palavras como preposições, adjetivos, pronomes, dentre outros. Inicialmente o foco ficou nos verbos em seu infinitivo, que representam ações e sempre terminam com R, como por exemplo, fazeR, falaR, subiR, e também as direções para frente, cima, baixo trás.

Nestes casos a identificação é feita por uma simples busca do caracter R no final da palavra, ou então pela simples comparação com uma lista de direções já existente (pois são simples e em menor número).

Resolvido este problema, o problema seguinte era como otimizar a busca pela palavra no banco. A busca é um problema complexo que aumenta de acordo com o número de entradas. Normalmente uma busca sequencial seria suficiente para solucionar o problema, porém foi interessante implementar uma busca binária onde foram analisados também os algoritmos de ordenação.

Os testes realizados com um número de até 1000 palavras não demonstraram um desafio computacional para o hardware de um computador pessoal. A diferença nas buscas era pequena, pois no caso da busca binária era necessário realizar uma ordenação que também acabava por acrescentar complexidade ao algoritmo e impactava no tempo de execução. O ganho pela implementação da busca binária com ordenação em comparação a busca sequencial (apresentado pela Figura 3) só pode ser notado a partir de aproximadamente 500 elementos, e no caso dos 1000 elementos testados a diferença de tempo foi de aproximadamente 0,10 segundos.

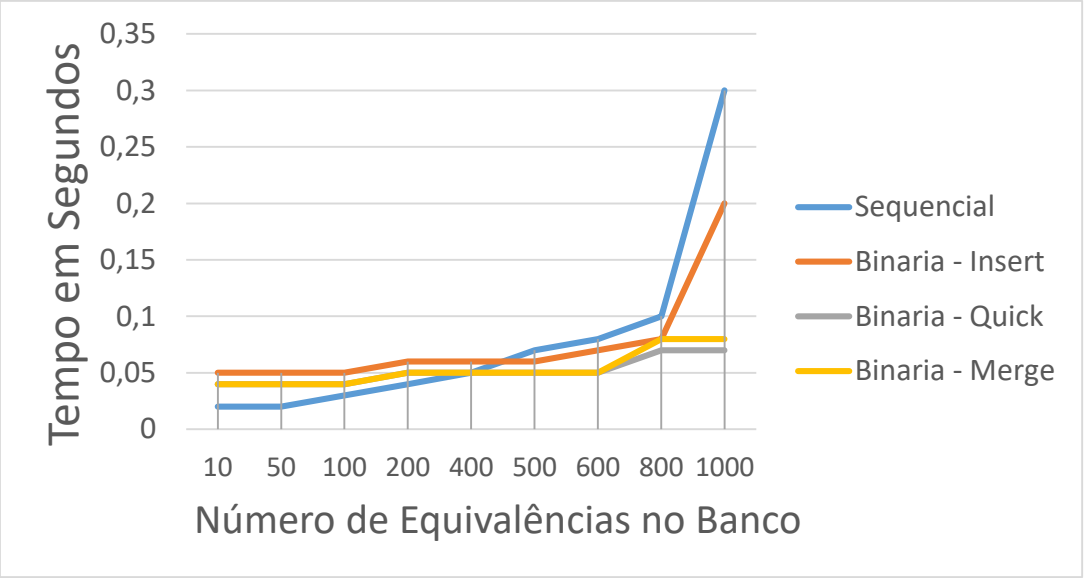

Figura 3. Gráfico do tempo de execução das diversas implementações
Fonte: Elaborado pelo autor

Algumas telas do *parser* são apresentadas pelas Figuras 4 e 5. Sendo assim, a implementação do *parser* está completa e será apresentada a seguir a análise dos resultados.

6 Análise e resultados

Os resultados demonstram que o *parser* desenvolvido atingiu o objetivo proposto que era de ser uma ferramenta que pode ser utilizada em jogos do gênero *text adventure* e que também apresentasse alguma vantagem em relação ao que há disponível em jogos atualmente.

A grande questão observada após o desenvolvimento do trabalho é que esta solução permite inclusive uma tradução de comandos. Caso uma analise inicial seja feita com os comandos existentes e uma lista de equivalência for composta, o mesmo jogo pode receber comandos em inglês e interpretar como comandos em português. A língua do jogo será mantida, porém um jogador pode utilizar comandos em sua língua nativa. Isso seria interessante para jogos que não possuem tradução para determinados tipos de idioma e possuem textos em inglês por exemplo.

A questão da implementação de diferentes tipos de buscas mostrou que a complexidade do algoritmo realmente influi na questão da implementação, porém a partir de um certo número de comandos equivalentes. O esforço não é

válido para um número menor que 500 casos, sendo a implementação e busca sequencial a mais simples e, portanto, a indicada para o caso.

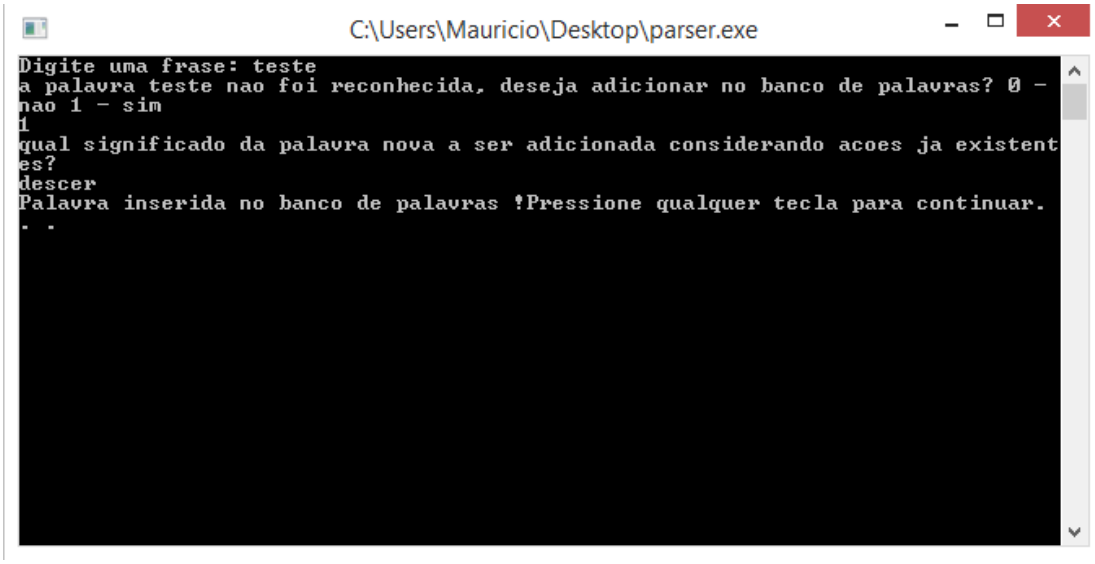

Figura 4. Exemplo de inserção de palavras
Fonte: Elaborado pelo autor

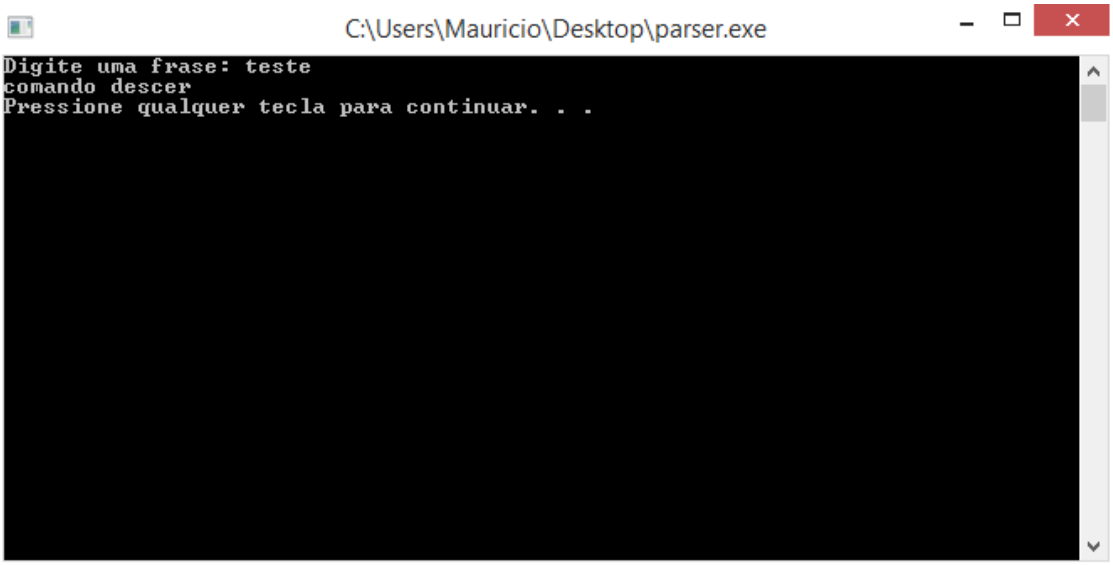

Figura 5. Palavra inserida sendo reconhecida como comando.
Fonte: Elaborado pelo autor

A estrutura desenvolvida também pode ser modificada para diferentes tipos de análise de palavras podendo identificar outras palavras e executar outras operações que não sejam baseadas em verbos e direções associadas a comandos. Esta estrutura pode ser reutilizada em qualquer software de análise textual para diferentes tipos de aplicações.

Considerações Finais

O objetivo principal deste trabalho foi o desenvolvimento de um *parser* inteligente para ser utilizado em jogos do estilo *text adventure*. Para isso foram estudadas as soluções existentes e foi proposta a modificação presente neste trabalho.

O objetivo foi atingido e o *parser* se comportou como esperado. A inteligência neste caso está no fato do software ser capaz de aprender e utilizar palavras novas a partir de um significado inicial atribuído a elas. Outra questão importante observada é a possibilidade de tradução de comandos para outras línguas sem modificações no código-fonte do jogo.

A implementação de diversos tipos de ordenação e mais de um algoritmo de busca permitiu a comparação e a escolha de uma forma otimizada, ainda que em casos específicos a busca sequencial seja a melhor alternativa.

Outro objetivo importante deste trabalho foi atingido que é o desenvolvimento de ferramentas para serem utilizadas pelo grupo de desenvolvimento de jogos da faculdade que possui atualmente um projeto de desenvolvimento de um jogo *text adventure*. Esta integração entre trabalhos de conclusão de curso e grupos de desenvolvimento é vital para a manutenção da qualidade de ambos.

Como trabalhos futuros têm-se o desenvolvimento de novas técnicas de identificação de palavras, bem como a integração em jogos desenvolvidos no grupo de jogos. Ainda seria interessante aplicar a estrutura em outros softwares que utilizam a identificação de palavras para seu funcionamento. Talvez a implantação de um banco de dados em memória secundária para determinados casos permita ao programa uma gama maior de possíveis aplicações.

Referências

ALMEIDA, “Desenvolvimento e Aplicação de um *Parser* Multilingual para Planejadores Automáticos”. Brasília, Julho de 2017.

ARANHA, 2006, “A Tecnologia de Mineração de Textos. Revista Eletrônica de Sistemas de Informação”.Disponível em:

<http://www.periodicosibepes.org.br/index.php/reinfo/article/view/171/66> Acesso em 2017 jun. 14.

BERGAMASCHI, 2014, “Estudo sobre a utilização de VUFORIA e Unity 3D com RA para dispositivos móveis”.
Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/wrva/2014/031.pdf>

COWAN E KAPRALOS, 2014, “A Survey of Frameworks and Game Engines for Serious Game Development”.

FILHO, 2009, “O Uso do Processamento de Linguagem Natural na Construção de Chatterbots”. Disponível em:
<https://dcc.catalao.ufg.br/up/498/o/Eustaquio2009.pdf>

GARIBA, M. JR., SCHNEIDER, M. C. K., ROSA, A. E., CASAGRANDE, J. B., SANTOS, C. S."Reconhecimento de Fala e Processamento da Linguagem Natural".

GONZALEZ, M. LIMA, V. L. S. “Recuperação de Informação e Processamento da Linguagem Natural”. XXIII Congresso da Sociedade Brasileira de Computação, Campinas, 2003. Anais do III Jornada de Mini-Cursos de Inteligência Artificial, Volume III, p.347-395.

JERZ, DENNIS G. Jerz's Introduction (Storytelling and Computer Games; UWEC Panel, May 2001). Disponível em:
<http://jerz.setonhill.edu/if/adams/intro.html> Acesso em 22 de Junho de 2017.

LOPES, S. C. MARIA, “Mineração de dados textuais utilizando técnicas de clustering para o idioma português”, Rio de Janeiro, Outubro de 2004.
Disponível em:
<http://www.inf.unioeste.br/~jorge/ARTIGOS%20INTERESSANTES/LING%DC%CDSTICA%20e%20MECANISMOS%20DA%20LINGUAGEM/Tese_coc_ufrj.pdf>

MAZIERO, PARDO e NUNES, “Identificação automática de segmentos discursivos: o uso do *parser* PALAVRAS”, São Carlos – SP, Agosto de 2007.

MIRANDA N. N. FREDERICO, “Gestor de narrativas para jogos de ficção interativa em texto”, Universidade de Lisboa - Campo Grande, 2012.

MORAES e AMBRÓSIO, “Mineração de textos”, Dezembro de 2007.
Disponível em: http://www.inf.ufg.br/sites/default/files/uploads/relatorios-tecnicos/RT-INF_005-07.pdf

NETO, TONIN e PRIETCH. “Processamento de Linguagem Natural e suas Aplicações Computacionais”. Rondonópolis – MT, 2010.

OLIVEIRA, “Criando e Recriando histórias”, Campo Limpo - SP, 2009.
Disponível

em:<http://www.niee.ufrgs.br/eventos/SBIE/2009/conteudo/artigos/completos/60931_1.pdf>

OLIVEIRA, FABIO ABREU DIASDE. “Processamento de linguagem natural: princípios básicos e a implementação de um analisador sintático de sentenças da língua portuguesa”. In: Revista de Ciência da Informação. Rio de Janeiro. n. 5. Maio 2002.

OLIVEIRA, POZZEBON e FRIGO, “Scribblenauts Unmasked: Avaliação do Jogo Digital e seus Aspectos Educativos”, Santa Catarina, Setembro de 2016.

OVCHINNIKOVA, E. Integration of World Knowledge for Natural Language Understanding. [S.l.]: Springer, 2012. 15 p.

PESSETTE e GERHARDT, “Uma Ferramenta para Criação de RPGs Textuais”, Florianópolis, Dezembro de 2004.

SEE e SOUZA, “Projeto e Implementação de um Game estilo *Adventure”*, Florianópolis,Junho de 2003.
SILVA, J. C. B. “Uma Arquitetura Multiagente para um Sistema de Processamento de Línguas Naturais Robusto e Evolutivo”. PhD thesis, Universidade de Lisboa, 2003.

SOUZA, M. V. T. “Desenvolvimento de Ambiente Computacional para Escrita e Leitura de Livros - Jogos em Dispositivos Móveis”, Recife, Junho de 2015.

VALENTE, “Guff: Um framework para desenvolvimento de jogos”, Niterói, Agosto de 2005.

VIEIRA e PEREIRA, “Jogos Digitais: Evolução, Instrumento em educação e mercado de trabalho”, Paraná, 2014.