UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

FERRAMENTA PARA AUXÍLIO NA TRADUÇÃO DE JOGOS DE COMPUTADOR

TRABALHO DE GRADUAÇÃO

Carlos Renan Silveira

Santa Maria, RS, Brasil

2008

FERRAMENTA PARA AUXÍLIO NA TRADUÇÃO DE JOGOS DE COMPUTADOR

por

Carlos Renan Silveira

Trabalho de Graduação apresentado ao Curso de Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de
Bacharel em Ciência da Computação

Orientador: Prof. Dr. Cesar Tadeu Pozzer

Trabalho de Graduação N. 241

Santa Maria, RS, Brasil

2008

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Graduação

FERRAMENTA PARA AUXÍLIO NA TRADUÇÃO DE JOGOS DE COMPUTADOR

elaborado por
Carlos Renan Silveira

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

COMISSÃO EXAMINADORA:

Prof. Dr. Cesar Tadeu Pozzer
(Presidente/Orientador)

Profª Dr. Iara Augustin (UFSM)

Profª Msc. Oni Reasilvia de Almeida Oliveira Sichonany (UFSM)

Santa Maria, 01 de Fevereiro de 2008.

DEDICATÓRIA

Dedico esse trabalho à minha família, minha namorada e aos meus amigos da comunidade de tradução, que passam grandes dificuldades traduzindo jogos.

AGRADECIMENTOS

Agradeço, primeiramente, à minha família, que sem o seu amparo e incentivo eu não estaria aqui.

Agradeço também à minha namorada, Adeline, que com carinho e dedicação, me ajudou nas horas difíceis, sempre me mantendo na linha e não deixando eu cair na tentação de ficar jogando o dia inteiro.

Não posso esquecer do meu orientador, Pozzer, que se não fosse pela sua matéria de Programação de Jogos 3D, a melhor matéria que tive em todo o curso, não aguçaria minha vontade de entrar no ramo de programação de jogos e afins.

Também menciono aqui meus colegas e amigos de aula e em especial o Scholz e o Bevilacqua, que me ajudaram na estruturação e formatação deste trabalho final.

A todos os meus amigos da Decadium Studios, agradeço pelo tempo disponibilizado para responder minhas perguntas, muitas vezes sem sentido algum para eles, mas de grande importância para mim.

E por fim, mas não menos importante, agradeço aos meus amigos da Mega Tecnologia, que foram muito compreensíveis a respeito do momento e das dificuldades que eu estava passando. Valeu Rafael!

"Quando não se pode fazer o que se deve, deve se fazer o que se pode."
— B. S. BASTOS

RESUMO

Trabalho de Graduação
Curso de Ciência da Computação
Universidade Federal de Santa Maria

FERRAMENTA PARA AUXÍLIO NA TRADUÇÃO DE JOGOS DE COMPUTADOR

Autor: Carlos Renan Silveira
Orientador: Prof. Dr. Cesar Tadeu Pozzer
Local e data da defesa: Santa Maria, 01 de Fevereiro de 2008.

Este trabalho apresenta o processo de pesquisa e desenvolvimento de uma ferramenta para auxílio na tradução dos textos de jogos de computador, a qual apresenta ao usuário uma interface integrada, simples e eficiente para a extração, tradução e posterior incorporação dos textos de arquivos de recurso dos jogos. A pesquisa tem como foco o estudo das características de algumas ferramentas disponíveis utilizadas por fãs na tradução de jogos, que possuem as funcionalidades necessárias à tarefa de tradução para então desenvolver uma nova ferramenta. Para o desenvolvimento da ferramenta, foram utilizadas a linguagem de programação C# e a plataforma .NET, em conjunto com o IDE *Visual C# 2008 Express Edition*. Além disso, esta ferramenta poderá ser de grande ajuda aos tradutores da comunidade de tradução de jogos GameVicio.

Palavras-chave: Tradução; jogos de computador; extração de texto; interpretador; editores de texto.

ABSTRACT

Trabalho de Graduação
Curso de Ciência da Computação
Universidade Federal de Santa Maria

TOOL FOR ASSISTANCE ON COMPUTER GAMES TRANSLATIONS

Author: Carlos Renan Silveira
Advisor: Prof. Dr. Cesar Tadeu Pozzer

This paper shows the researching process and development of a tool to assist in the translation of the computer games texts, ofering a integrated and simple interface, but eficient, for extraction, translation, and after incorporation of the texts in the computer games's resource files. The research has as main focus the study of the peculiarities of some avaliable tools wich have the necessary functionalities to the task of translating the game's texts. In the developement of this tool it was used the C# programing language and the .NET platform, combined with the IDE *Visual C# 2008 Express Edition*. Besides that, the developed tool is very helpful to the users of the games translation community GameVicio.

Keywords: translation, computer games, text extraction, interpreters, text editor.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ANSI	American National Standards Institute
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
COM	Component Object Model
DLL	Dynamic Link Library
IDE	Integrated Development Environment
PS2	Playstation 2
PSP	Playstation Portátil
XBOX	Console desenvolvido pela Microsoft
XML	Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

Cresce a cada dia a demanda por jogos eletrônicos em todos os países do mundo. Nos primeiros dois meses de 2007, as vendas domésticas de jogos de computador alcançaram US$203 milhões, sendo um aumento de 48% em relação ao mesmo período do ano passado (MARRIOT, 2007). Segundo (BERNAL-MERINO, 2007), a indústria de jogos eletrônicos está ganhando tanto quanto a indústria de filmes, mas ela também precisa de orçamentos elevados para cobrir os custos de desenvolvimento de um jogo. Por causa disso, fica clara a necessidade de expandir mercados.

Existem vários mercados emergentes, como o brasileiro, que aumentou as vendas de computadores em 23% em relação ao ano passado (BRAUN, 2007). Empresas que desejam lucrar mais com a venda de jogos de computador precisam também entrar nesses mercados emergentes, mesmo que sejam pequenos comparados aos grandes mercados, como o dos Estados Unidos, Europa e Ásia. Um exemplo de interesse em outros mercados é o caso da empresa *Eletronic Arts*, que em 2005 abriu um centro de desenvolvimento e tradução em Singapura, para atender o mercado asiático em crescimento (YUNKER, 2005).

Segundo (CHANDLER, 2004), "localização é o processo de traduzir o jogo em outras línguas". Historicamente a localização de jogos se resumia em apenas a tradução de algumas frases inseridas no código. Porém, hoje ela já faz parte do processo de desenvolvimento de jogos de várias empresas. Se, no processo de desenvolvimento, não forem tomadas medidas para que a internacionalização do jogo seja possível, a empresa terá sérios problemas quando desejar traduzir o jogo posteriormente para outras línguas. Existem basicamente dois tipos de localização: a localização completa e a localização parcial.

A localização parcial traduz somente os textos dos jogos. Não leva em conta as falas

e imagens. Esse tipo de localização é barata, visto que não é necessário contratar atores, agendar sessões de gravação de voz em estúdios e processar arquivos de som. Além disso, a localização parcial de um jogo pode ser desafiante quando é usado sincronização labial nos personagens do jogo, pois as animações faciais talvez necessitam de remodelagem para ter um maior nível de qualidade.

Ela é perigosa pois é necessário alterar o código do jogo, na qual demanda mais programação e tempo em testes. Os textos traduzidos também devem ser analisados cuidadosamente para que qualquer alteração no texto da língua original seja repassado às versões traduzidas. O benefício é que os jogadores possuirão uma experiência de jogo mais personalizada. As localizações parciais são geralmente feitas para jogos de qualidade lançados em mercados secundários como Holanda e Itália (CHANDLER, 2004).

Já a localização completa é aquela em que textos, sons, imagens, manuais e a caixa do jogo são traduzidas completamente para o mercado alvo. O grande problema desse tipo de localização é o custo para se ter todo o jogo traduzido. O benefício é que o jogador receberá o jogo totalmente adequado à sua língua e cultura.

Ambos os tipos são feitos pelas *softhouses*, empresas desenvolvedoras do jogo, e pelas *publishers*, empresas especializadas em divulgar o jogo. Mas existem também as localizações feitas pelos fãs. Tais localizações são ilegais, pois infringem leis de *copyright*, porém são as soluções encontradas pelos jogadores, já que a desenvolvedora do jogo não dá mais suporte, no caso de jogos antigos, ou não criou o jogo contendo outras línguas. Mesmo sendo contra as leis, tais localizações nem sempre são contestadas pelos detentores dos direitos sobre o jogo, geralmente porque os jogos eletrônicos em questão não são considerados comercialmente viáveis na língua de destino (Wikipedia, 2007a). Isso acontece para os países em desenvolvimento onde a população não conhece a língua nativa do jogo, geralmente o inglês, tal como o Brasil.

No Brasil, com exceção das pessoas que utilizam o espanhol nas fronteiras e de outras línguas em espaços de comunidades de imigrantes (polonês, italiano, alemão, etc.) e de grupos nativos, somente uma parcela pequena da população utiliza alguma língua estrangeira como exercício da comunicação oral, e mesmo em grandes centros a porcentagem de pessoas que utilizam línguas estrangeiras é relativamente pequena (Ministério da Educação e do Desporto. Secretaria de Educação Fundamental, 1998).

Para que a distribuidora tenha interesse no mercado de entretenimento do Brasil, a

ponto de traduzir um jogo para o português brasileiro, é necessário, dentre outros fatores, melhorar a imagem do mercado brasileiro no que se refere a jogos de computador, visto que aqui a pirataria de produtos, principalmente eletrônicos, como aplicativos e jogos, é ainda muito alta. Apesar dos esforços existentes por parte do governo na questão da pirataria, a porcentagem de jogos piratas ainda é altíssima e isso faz com que a indústria de jogos não enxergue com bons olhos o mercado consumidor brasileiro (AZEVEDO, 2006).

Como são poucas as empresas que lançam jogos em português já de fábrica, surgem comunidades no Brasil que desenvolvem as localizações não oficiais. Tais comunidades geralmente utilizam os próprios *softwares* existentes disponibilizados pela desenvolvedora do jogo. Tais ferramentas são utilizadas para fazer modificações no jogo. Uma tradução desenvolvida por comunidades é aplicada no jogo como uma extensão para os jogos que possuem essa característica, ou então os arquivos de recursos originais são substituídos pelos arquivos modificados, contendo os novos textos e imagens traduzidas.

Muitas vezes não é possível conseguir as ferramentas das próprias desenvolvedoras e, por isso, os usuários das comunidades criam ferramentas capazes de extrair os textos dos arquivos de recursos dos jogos. Essas ferramentas, na maioria das vezes, só servem para os arquivos do jogo a que foram destinadas.

Considerando o cenário acima delimitado, este trabalho visa o desenvolvimento de uma ferramenta para auxiliar nas traduções não-oficiais de jogos de computador, utilizando a linguagem de programaçao C# (ECMA International, 2006) e a plataforma .NET. Com ele pretende-se contribuir para a comunidade de tradução com uma ferramenta que reúna os principais recursos utilizados na tarefa de tradução de jogos de computador.

O presente trabalho está organizado em quatro capítulos. O capítulo 2 discute como é feita uma tradução não-oficial, apresentando as características desse tipo de tradução, com exemplos de algumas ferramentas utilizadas pelas comunidades de tradução de jogos. O capítulo 3 apresenta a ferramenta desenvolvida e seus módulos principais, a linguagem criada para ser utilizada na extração e incorporação dos textos e a interface de manipulação de textos, bem como um exemplo de uso da ferramenta. O capítulo 4 apresenta as conclusões da pesquisa e desenvolvimento da ferramenta.

2 TRADUÇÕES DE JOGOS DE COMPUTADOR

Este capítulo descreve o comportamento das comunidades de tradução, sua divisão de cargos e como são feitas as traduções não-oficiais. Também mostra algumas ferramentas utilizadas pelas comunidades na tarefa de traduzir um jogo. No final, é mostrado um comparativo das funcionalidades existentes entre as ferramentas apresentadas e a desenvolvida.

2.1 As traduções de fãs

A razão de localizações serem feitas por fãs é que, principalmente no Brasil, grande parte dos jogos estão em sua língua original, em sua maioria o inglês, ou então a distribuidora do jogo libera para o Brasil a versão em espanhol, pois essa é a língua com maior atuação na América do Sul. Hoje em dia, ainda não é comum se encontrar jogos totalmente localizado em português, com sons, textos e imagens, apesar de ter aumentado o lançamento de jogos destinados ao público brasileiro.

A tradução feita por terceiros não possui garantias de qualidade da escrita, nem do funcionamento, apesar das traduções serem testadas pelos próprios criadores e pela comunidade de tradutores. Muitos fãs se reunem e formam grandes comunidades de tradução, para a mútua ajuda no trabalho de traduzir jogos. Essas comunidades possuem organização de trabalho, com divisão de responsabilidades e criação de projetos de tradução. Por ser um trabalho voluntário, as traduções não seguem um cronograma fixo, pois a disponibilidade de tempo varia de tradutor para tradutor.

2.1.1 A comunidade GameVicio

A GameVicio é uma das maiores comunidades de tradução de jogos do Brasil. Possui mais de quinhentos mil inscritos e contém mais de duzentas e cinqüenta traduções pron-

tas, que podem ser obtidas gratuitamente do *site* (FRAUCHES, 2007). Segundo (Alexa, 2007), o *site* figura na posição 178º da classificação brasileira de *sites*.

A comunidade foi criada em 2003 por Marcos Said, hoje diretor do portal. O *site* exibe informações relacionadas a jogos eletrônicos, mas a principal atividade são as traduções feitas pela comunidade. Possui uma hierarquia onde figuram administradores, moderadores, administradores de projetos e tradutores, cada um com suas responsabilidades.

Possui também um gerenciador de projetos, onde os arquivos disponibilizados pelos administradores de projetos são baixados pelos tradutores e, após serem traduzidos, são enviados novamente para a revisão, através do gerenciador. No gerenciador também aparecem dados estatísticos do número de arquivos totais, quantos já foram traduzidos, quantos estão reservados por tradutores e quantos já foram revisados e estão prontos para serem incorporados ao jogo.

2.2 O uso de ferramentas nas traduções não-oficiais

Na tradução feita por terceiros, várias ferramentas são utilizadas pelos tradutores. As principais são editores hexadecimais, como o *XVI32*, que servem para abrir os arquivos de recursos dos jogos e identificar sua estrutura analisando os bytes que formam o arquivo, e programas para extrair os textos dos arquivos de recursos depois de identificado sua estrutura. Esses programas extratores são desenvolvidos algumas vezes pelos próprios tradutores que conhecem algum tipo de linguagem de programação, ou então eles utilizam ferramentas destinadas a esse tipo de tarefa, como por exemplo o *Game Extractor* e o *MultiEx Commander*. Outros programas utilizados pelos tradutores são editores textuais, comparadores de textos, como por exemplo o *Beyound Compare*, tradutores eletrônicos, dicionários eletrônicos e qualquer outro *software* que seja útil para trabalhar com textos e traduções.

A seguir são detalhados algumas das ferramentas utilizadas em todo o processo de tradução de um jogo de computador, descrevendo algumas características das mesmas e expondo seus pontos fortes e fracos.

2.2.1 A ferramenta *XVI32*

O XVI32 (MAAS, 2003) é um editor hexadecimal gratuito que roda na plataforma Windows, utilizado para decodificar arquivos. Ele possui um inspetor de dados, uma

janela onde é informado o significado da seqüência de dados demarcada. Trabalha facilmente com grandes arquivos, podendo abrir arquivos com até 2GB de tamanho. Ele guarda as configurações em arquivos que são salvos na mesma pasta do executável, e por isso não altera o registro do sistema operacional.

Apresenta os dados tanto em formato texto (ASCII e ANSI) como em hexadecimal, com dois cursores sincronizados, um na área de texto e outro na área dos dados em hexadecimal, como é visto na Figura 2.1. É possível redimensionar completamente a janela, para adaptar a visão das colunas, bem como alterar as fontes e o tamanho das mesmas.

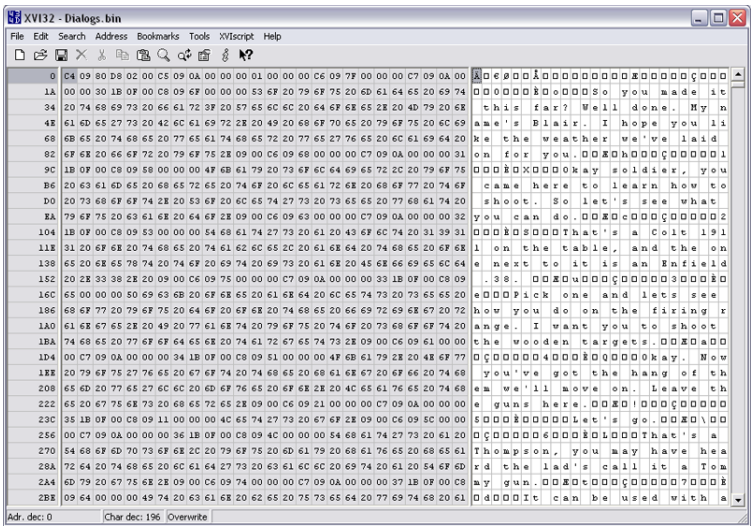

Figura 2.1: O editor hexadecimal *XVI32* com um arquivo binário aberto

É possivel fazer buscas tanto por seqüência de carateres (textos) como por valores em hexadecimal, como também a substituição de dados em ambas as formas de representação. Sua instalação é simples, bastando descompactar o conteúdo do arquivo que foi obtido na internet para uma pasta qualquer, e por ser gratuíto, se torna uma ferramenta extremamente aconselhada para auxiliar na tarefa de tradução.

2.2.2 A ferramenta *Game Extractor*

Game Extractor (WATSON, 2002) é uma ferramenta de arquivos desenvolvida primeiramente para abrir e manipular arquivos de jogos. É multiplataforma, com binários para as principais distribuições, como Windows, Linux e Mac O/S. Desenvolvida em linguagem JAVA[1], possui um sistema de *plugins*, possibilitando ao usuário desenvolver seu

[1]JAVA é uma linguagem de programaçao desenvolvida pela empresa Sun Microsystems. É uma linguagem interpretada, orientada à objetos. Sua compilação gera "bytecodes"que são intepretados por uma máquina virtual. Os programas escritos na linguagem JAVA podem ser executados em qualquer sistema operacional que tenha a máquina virtual JAVA (DEITEL; DEITEL, 2001)

próprio *plugin* para ler um arquivo que ainda não é reconhecido pela ferramenta. Além das funcionalidades básicas de visualizador de arquivos, também possui renomeador de arquivos, busca e customização da interface. Tem suporte para arquivos de jogos de várias plataformas, entre elas Windows, XBOX, PS2 e PSP. A Figura 2.2 mostra a interface do programa, com alguns arquivos listados.

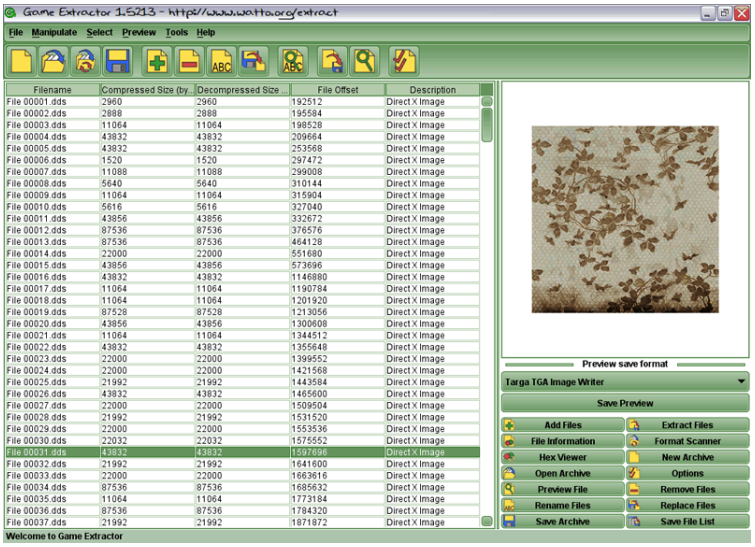

Figura 2.2: Arquivos listados na ferramenta *Game Extractor*

Existem duas versões desta ferramenta. A versão básica contém todas as funcionalidades necessárias para ler arquivos de jogos e é gratuíta. Já a versão completa contém todas as funcionalidades para editar arquivos de jogos, como também a possibilidade de pré-visualizar alguns tipos de arquivos. Possui também um *scanner* para indentificar arquivos ainda não suportados pela ferramenta. Apesar da versão completa ser paga, o valor é baixo, sendo o custo de US$5,00.

2.2.3 A ferramenta *MultiEx Commander*

O software *MultiEx Commander* foi criado pela *The XeNTaX Foundation* (The XeNTaX Foundation, 2007). É uma ferramenta capaz de analisar, extrair e importar dados dos arquivos de recursos dos jogos. O interessante desta ferramenta é a capacidade de extensão de arquivos de jogos na qual ela pode analisar, utilizando um sistema de *plugins*. Cada *plugin* contém código que é interpretado pela ferramenta, possibilitando qualquer usuário criar novos *plugins* para arquivos de jogos cuja a ferramenta não tenha ainda a capacidade de analisar. A Figura 2.3 mostra um exemplo do funcionamento da ferramenta, listando os arquivos existentes.

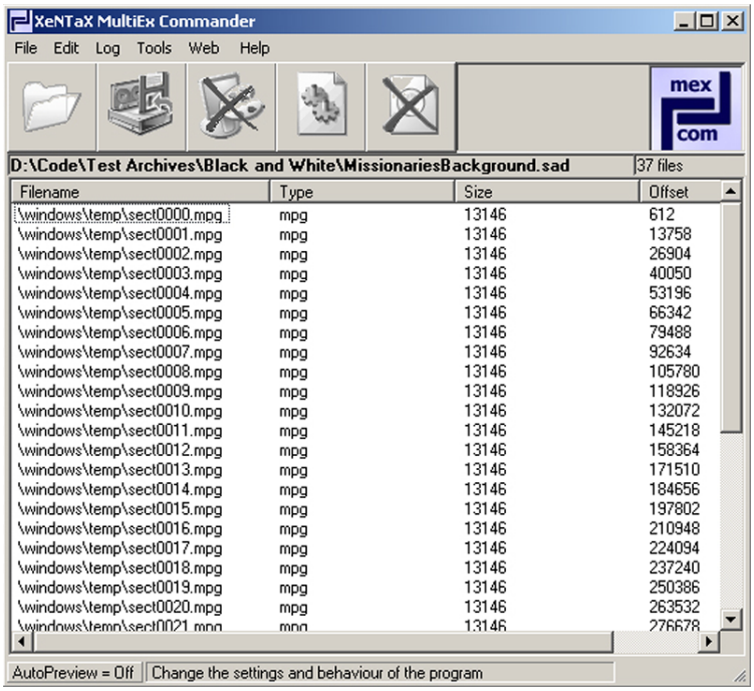

Figura 2.3: Uma lista de arquivos de imagem listados no *MultiEx Commander*

O ponto negativo da ferramenta *MultiEx Commander* é que ela é paga. Existe uma versão de demonstração que pode ser utilizada apenas para analisar arquivos de recursos que já são reconhecidos pela ferramenta. A versão completa custa 10,00 €, e libera todas as suas funcionalidades, como edição de arquivos e criação de novos arquivos de recursos e a pré-visualização de outros tipos.

2.2.4 A ferramenta *Beyound Compare*

Beyond Compare (Scooter Software, 2007) é um aplicativo utilizado para comparar arquivos e pastas, mostrando as diferenças entre dois arquivos ou duas pastas. É um poderoso utilitário, capaz de criar um *snapshot* de um diretório, e compará-lo com um *snapshot* antigo. Um *snapshot* é um instantâneo da comparação que está sendo feita. É armazenado em um arquivo a relação das pastas ou arquivos comparados, e esta relação pode então ser usada para futuras comparações, sem a necessidade de abrir as pastas ou arquivos originais novamente.

Também é capaz de mesclar mudanças nos arquivos e comparar saídas de programas. Possui destacamento de código por diferentes cores, para fácil comparação das diferenças.

Esta é uma ferramenta cara, variando de US$30,00 para uma licença privada até US$2500,00 para uma licença corporativa. A versão de demonstração funciona por 30

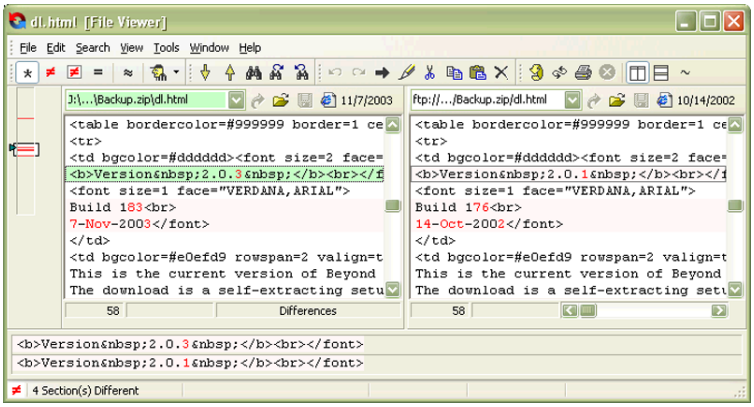

Figura 2.4: Dois arquivos sendo comparados no aplicativo Beyond Compare

dias e possui todas as funcionalidades ativas.

2.3 Comparativo das funcionalidades

A primeira funcionalidade da ferramenta *Translator Assistant* é o sistema de comparação de textos. Assim como a ferramenta *Beyound Compare*, a ferramenta desenvolvida possui a capacidade de comparar arquivos textuais. A diferença existente é que a comparação é feita apenas da linha destacada nos arquivos, e não de todas as linhas automaticamente, como é feito no *Beyound Compare*. Também não é possível comparar pastas, como na outra ferramenta, porém é possível comparar até três arquivos simultaneamente, em Unicode ou ASCII. Mesmo não sendo possível a comparação de pastas, essa funcionalidade não é necessária para a tarefa de tradução.

A segunda funcionalidade da ferramenta *Translator Assistant* é a capacidade de extrair dados (textos e outras informações) de arquivos de recurso dos jogos. Essa funcionalidade, presente nas ferramentas MultiEx Commander e Game Extractor, permite que a ferramenta desenvolvida extraia os textos dos arquivos de recursos para posterior tradução. Da mesma forma nas ferramentas citadas acima, é utilizado um arquivo de texto estruturado, descrito na Seção 3.2.2, que contém informações e comandos escritos numa linguagem definida, que são interpretados pela ferramenta para extrair as informações dos arquivos de recurso dos jogos. A diferença que existe é que a ferramenta desenvolvida não é paga e não é possível extrair imagens e sons, apenas textos.

Na Tabela 2.1 está a comparação resumida das funcionalidades entre as ferramentas apresentadas e a ferramenta desenvolvida. Observa-se nessa tabela que a ferramenta desenvolvida apresenta 4 funcionalidades essenciais para a tarefa de tradução: compara-

Tabela 2.1: Comparação das funcionalidades entre as ferramentas existentes

	Beyound Compare	Game Extractor	MultiEx Commander	XVI32	Translator Assistant
Comparação de pastas	X	-	-	-	-
Comparação de arquivos textuais	X	-	-	-	X
Extração de textos	-	X	X	-	X
Extração de imagens	-	X	X	-	-
Extração de sons	-	X	X	-	-
Edição hexadecimal	-	-	-	X	-
Edição de textos	-	-	-	-	X
Incorporação parcial das modificações	-	X	X	-	X
Preço	US$25,00	US$5,00	10,00 €	Gratuíto	Gratuíto

ção de arquivos textuais, extração de textos, edição de textos e incorporação parcial das modificações.

Não foi feita a análise de um editor de texto, visto que qualquer editor pode ser utilizado, sem restrições, e fica por conta do gosto do tradutor. É possível utilizar a ferramenta desenvolvida para editar os textos ou outra qualquer da preferência do tradutor.

3 A FERRAMENTA TRANSLATOR ASSISTANT

Nesta seção é descrito como a ferramenta proposta, chamada Translator Assistant, foi desenvolvida, suas principais funcionalidades e como tais funcionalidades irão ajudar na tarefa de traduzir um jogo de computador. Também são descritas as características da linguagem C# e da plataforma .NET, que foram utilizadas na construção da ferramenta.

A ferramenta é dividida em dois grandes módulos: o módulo de extração de textos, que inclui os sistemas de interpretação e de gerenciamento de *plugins*, e o módulo de edição de textos. Essa estrutura pode ser vista melhor na Figura 3.1.

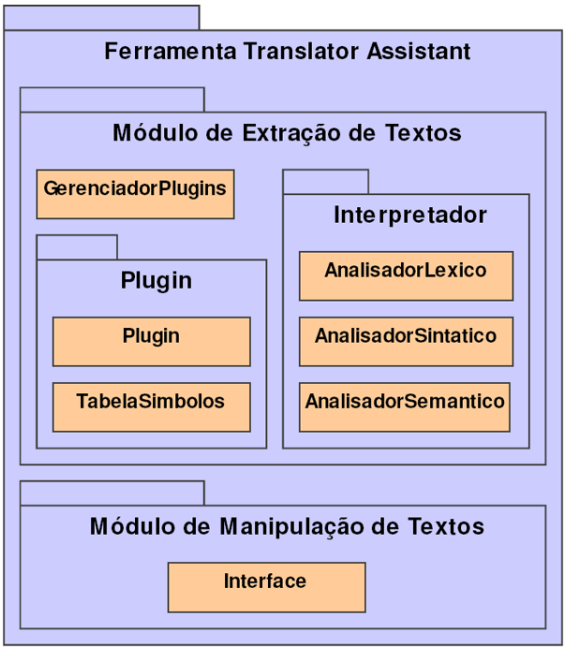

Figura 3.1: A estrutura da ferramenta Translation Assistant

Na Figura 3.2 é possível observar as classes que representam o funcionamento de um *plugin*. A classe *Plugin* é responsável por manipular as informações presentes em um *plugin*, com funções de extração dos códigos de leitura e de escrita, e da chamada da função que executa esses códigos. A classe *TabelaSimbolos* é responsável por geren-

ciar as variáveis encontradas nos códigos de leitura e escrita do *plugin*, e utiliza a classe *ItemTabelaSimbolo* para armazernar o nome, tipo e valor de cada variável declarada. A classe *GerenciadorPlugins* é responsável por registrar os *plugins* no menu principal da ferramenta.

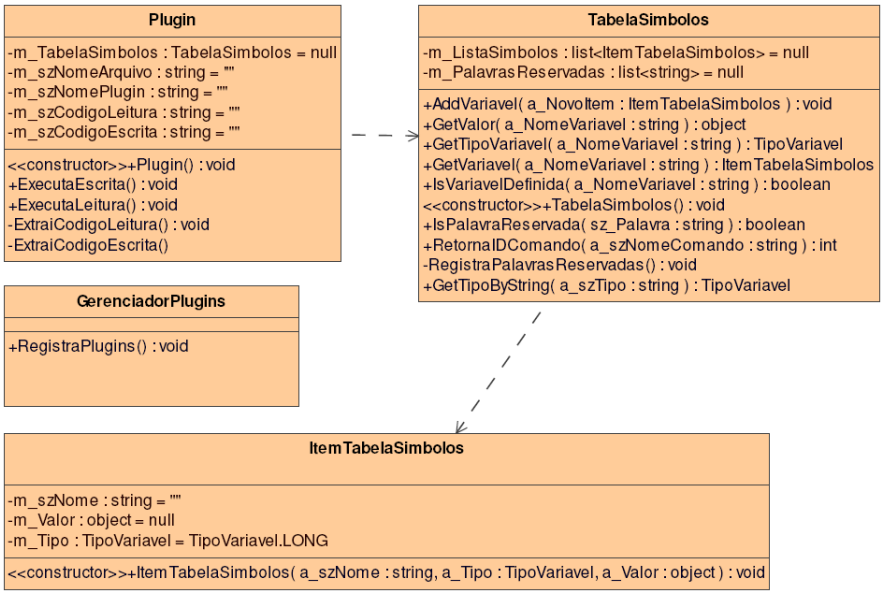

Figura 3.2: Classes que representam o funcionamento de um *plugin*

A Figura 3.3 mostra as classes que representam o funcionamento do interpretador da ferramenta. A classe *Interpretador* é responsável por executar os códigos de leitura e escrita do *plugin*, e utiliza a classe *AnalisadorLexico* para interpretar uma linha de código, separando em unidades de controle.

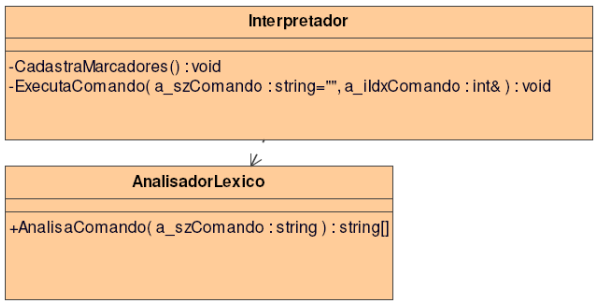

Figura 3.3: Classes que representam o funcionamento do interpretador da ferramenta

3.1 Ambiente de desenvolvimento

Nas seções seguintes são descritas as características da plataforma e da linguagem utilizada para desenvolver a ferramenta. Foi escolhida a linguagem C# para o desenvol-

vimento pois é uma linguagem robusta e de fácil utilização, já consolidada no ambiente *Windows* e com possibilidade de portar a ferramenta para outros sistemas operacionais, como *Linux* e *MacOS*.

3.1.1 A plataforma .NET

A plataforma .NET é um modelo completamente novo para criar programas nos sistemas operacionais da família Windows bem como em outros sistemas operacionais, tal como MacOS X e várias distribuições Linux/Unix (TROELSEN, 2007). Algumas características principais da plataforma .NET são:

- Interoperatibilidade compreensiva entre código legado: binários COM podem interagir com novos binários .NET e serviços de chamada de plataforma possibilitam que bibliotecas escritas em C[1] sejam chamadas no código .NET.

- Integração completa e total entre linguagens: a plataforma .NET suporta herança entre linguagens de programação diferentes, bem como tratamento de exceções e depuração de código.

- Uma *engine* comum compartilhada entre todas as linguages com suporte .NET: um dos aspectos desta *engine* é um conjunto de tipos bem definidos que todas as linguagens entendem.

- Uma biblioteca de classes básica compreensível: esta biblioteca provê abrigo da complexidade das chamadas da API de baixo nível e oferece um modelo de objeto consistente que é usado por todas as linguagens que suportam a plataforma.

- Um modelo de publicação realmente simples: não existe a necessidade de cadastrar um binário no registro do sistema. Além disso, .NET permite que múltiplas versões de uma DLL coexistam em harmonia na mesma máquina.

A plataforma .NET foi desenvolvida sobre três pilares principais: o CLR (*Common Language Runtime*), o CTS (*Common Type System*) e o CLS (*Common Language Specification*). Do ponto de vista do programador, a plataforma pode ser entendida como um ambiente de execução e uma biblioteca de classes básicas.

[1] C é uma linguagem de programação procedural, imperativa, de propósito geral, criada nos laboratórios ATT Bell por Dennis Ritchie em 1972 (SCHILDT, 1985).

A camada de execução é referenciada como a *common language runtime*, ou CLR. A primeira tarefa da CLR é localizar, carregar e gerenciar os tipos .NET, e tambem é responsável pelos detalhes de baixo nível como gerenciamento de memória, criação dos domínios da aplicação, checagens de segurança, limites de contexto de objetos e *threads*.

Outro pilar da plataforma .NET é o CTS. A especificação do CTS descreve completamente todas as possibilidades de tipos de dados e contruções de programação suportados pela camada de execução, especifica como essas entidades podem interagir uma com outra e detalha como elas são representadas no formato de metadado da plataforma .NET.

A CLS é uma especificação que define um subconjunto de tipos comuns e construções de programação que todas as linguagens que dão suporte à plataforma .NET devem entender. Portanto, construindo-se tipos .NET que obedecem as especificações da CLS, é garantido que qualquer outra linguagem que tenha suporte à plataforma possa utilizá-los. A Figura 3.4 mostra a relação entre as três entidades que constituem a plataforma .NET.

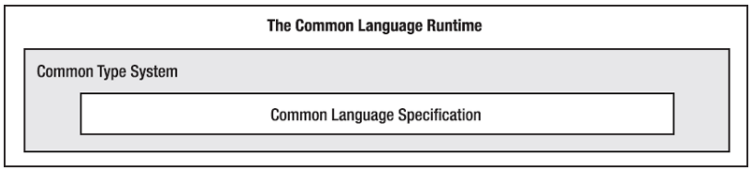

Figura 3.4: A relação entre as três entidades que formam a base da plataforma .NET

3.1.2 A linguagem de programação C#

Sendo a plataforma .NET completamente diferente das tecnologias antigas, a Microsoft criou a linguagem C# especialmente para trabalhar em conjunto com a nova plataforma. A seguir são apresentadas algumas características da linguagem.

- Sem necessidade de ponteiros: programas escritos em C# não tem necessidade de manipulação direta com ponteiros, mas pode ser feito se necessário.

- Gerenciamento automático de memória através de coletor de lixo: por ter coletor de lixo, a linguagem C# não possui a palavra reservada *delete* para a liberação da memória ocupada.

- Sobrecarga de operadores: assim como em C++[2], é possível trabalhar com sobrecarga de operadores em C#.

[2]C++ é uma linguagem de programação orientada à objetos derivada do C, com um alto nível de abstração. Foi criada por Bjarne Stroustrup em 1983 nos laboratórios da ATT Bell (STROUSTRUP, 1991).

- Programação genérica: é possível trabalhar com tipos e membros genéricos, construindo códigos seguros e eficientes.

- Criação de tipos através de vários arquivos: é possível criar classes contendo parte do seu código em vários arquivos, através da palavra reservada *partial*.

Além dessas características, o principal ponto que deve ser entendido é que a linguagem C# só produz código executável através da plataforma .NET. Tal código é chamado "código gerenciado". As unidades binárias que possuem código gerenciado são chamados de *assembly*.

Os *assemblies* são constituídos de códigos em CIL (*Common Intermediate Language*), metadados e um manifesto. Por conter esses três elementos, essas unidades binárias são auto-descritivas. O código CIL representa o código escrito em uma linguagem intermediária. O código CIL é compilado na execução, utilizando um compilador JIT (*just-in-time*) para a plataforma e CPU específicos. Desse modo, os *assemblies* podem ser executados em uma variedade de arquiteturas, dispositivos e sistemas operacionais.

Os metadados descrevem completamente os formatos dos tipos internos como também os formatos dos tipos contidos em *assemblies* referenciados. O *runtime* do .NET utiliza esse metadado para resolver a localização de tipos e seus membros dentro do binário e facilitar a chamada de métodos remotos.

O manifesto é responsável por documentar cada módulo dentro do *assembly*, estabelece a versão da unidade binária e também documenta qualquer *assembly* externo referenciado pela unidade corrente. Um exemplo de código-fonte em C# pode ser visto com mais detalhes no Apêndice B.

3.2 A ferramenta desenvolvida

Nas subseções seguintes são descritos os módulos que compõem a ferramenta, a estrutura de um arquivo de extensão que é reconhecido pela ferramenta e a linguagem criada utilizada para extrair os textos dos arquivos de recurso dos jogos. Também é descrito o funcionamento do gerenciador de *plugins* e do interpretador implementado na ferramenta.

3.2.1 O módulo de extração de textos

A principal funcionalidade da ferramenta é o módulo para auxiliar na extração dos textos dos arquivos de recursos dos jogos de computador. Esse módulo contém um sis-

tema de gerenciamento de *plugins* e um interpretador dos comandos existentes em um *plugin*. Os comandos são separados em seções distintas de leitura e escrita dentro do *plugin*, pois nem sempre a mesma lógica é aplicada para extrair os textos dos arquivos de recursos e para salvar os textos traduzidos nos mesmos arquivos.

3.2.2 A estrutura dos arquivos de extensão

Os arquivos de extensão, também chamados de *plugins*, são utilizados para extrair os textos dos arquivos de recursos dos jogos. Eles contêm códigos escritos na linguagem criada, descrita no item 3.2.4, que são interpretados pela ferramenta. Os comandos são executados sobre o arquivo de recurso alvo, no qual são extraídos os textos e mostrados na primeira área de trabalho da ferramenta, como mostra a Figura 3.13.

A forma como é escrito um *plugin* segue a idéia de um arquivo XML (BOS, 1999) e pode ser visto na Figura 3.6. Ele possui tags que definem as informações que serão interpretadas pela ferramenta. Cada *plugin* é dividido em três seções.

A primeira seção serve para identificiar o *plugin*, informando o nome que irá aparecer no menu entre as tags `<name>` e `</name>`, o nome do jogo, que é informado entre as tags `<game>` e `</game>` e a extensão do arquivo que é identificado, colocado entre as tags `<extension>` e `</extension>`. Essas informações servem para guiar a ferramenta no momento da criação do item de menu do *plugin*. Se tais informações não estiverem presentes no arquivo devidamente identificadas pelas tags, a ferramenta acusará um erro na hora de montar o menu.

A segunda seção é onde contém o código que será executado na extração dos textos do arquivo alvo. O código vem entre as tags `<code_read>` e `</code_read>`. Os comandos devem seguir as regras da linguagem definidas no item 3.2.4. Se algum comando estiver errado, a ferramenta acusará o erro e informará a linha do comando.

Na terceira seção irá o código a ser executado quando for selecionada a opção de salvar o texto traduzido. Os comandos devem vir entre as tags `<code_write>` e `</code_write>`, e seguem as mesmas regras da seção do código de leitura, também gerando um erro caso a ferramenta encontre comandos errados, falta de parâmetros ou parâmetros incorretos.

As três seções descritas acima podem ser vistas na Figura 3.5, que mostra a estrutura básica de um *plugin*.

```
<plugin>
        <menu>
                <name></name>
                <game></game>
                <extension></extension>
        </menu>
        <code_read>

        </code_read>
        <code_write>

        </code_write>
</plugin>
```

Figura 3.5: Exemplo da estrutura de um *plugin*

O intuito de utilizar *plugins* na ferramenta é para dar a capacidade de expansão de tipos de arquivos de recursos dos jogos que a ferramenta poderá analisar, extrair e salvar. Se não existisse essa capacidade, para cada novo jogo lançado que utiliza um arquivo de recurso diferente, seria necessário a criação de uma nova ferramenta, ou a recompilação da ferramenta existente, para dar suporte a esse novo arquivo.

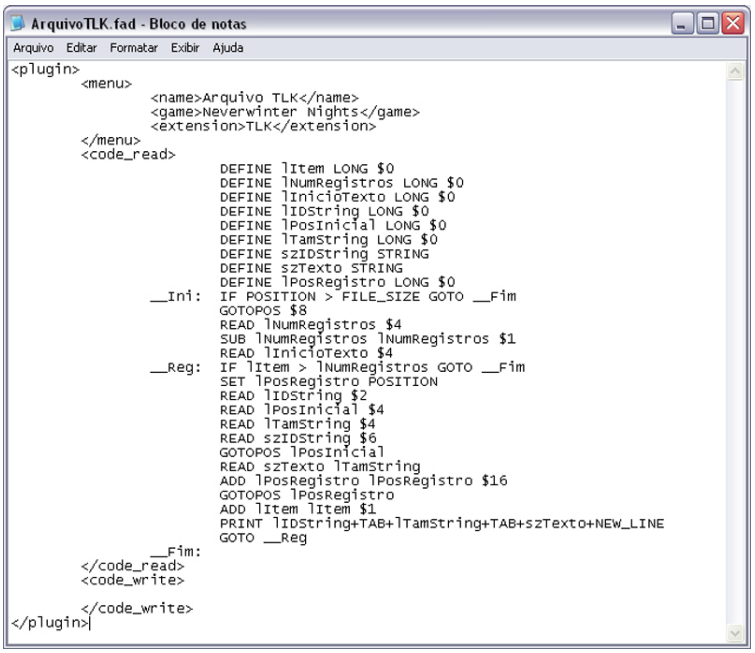
ArquivoTLK.fad - Bloco de notas
Arquivo Editar Formatar Exibir Ajuda
<plugin>
<menu>
<name>Arquivo TLK</name>
<game>Neverwinter Nights</game>
<extension>TLK</extension>
</menu>
<code_read>
DEFINE lItem LONG $0
DEFINE lNumRegistros LONG $0
DEFINE lInicioTexto LONG $0
DEFINE lIDString LONG $0
DEFINE lPosInicial LONG $0
DEFINE lTamString LONG $0
DEFINE szIDString STRING
DEFINE szTexto STRING
DEFINE lPosRegistro LONG $0
__Ini: IF POSITION > FILE_SIZE GOTO __Fim
GOTOPOS $8
READ lNumRegistros $4
SUB lNumRegistros lNumRegistros $1
READ lInicioTexto $4
__Reg: IF lItem > lNumRegistros GOTO __Fim
SET lPosRegistro POSITION
READ lIDString $2
READ lPosInicial $4
READ lTamString $4
READ szIDString $6
GOTOPOS lPosInicial
READ szTexto lTamString
ADD lPosRegistro lPosRegistro $16
GOTOPOS lPosRegistro
ADD lItem lItem $1
PRINT lIDString+TAB+lTamString+TAB+szTexto+NEW_LINE
GOTO __Reg
__Fim:
</code_read>
<code_write>
</code_write>
</plugin>
```

Figura 3.6: Exemplo de um *plugin* aberto no Bloco de Notas

### 3.2.3 O gerenciador de *plugins*

Quando a ferramenta é inicializada, o gerenciador verifica se existem *plugins* na pasta configurada e para cada arquivo existente são construídos os submenus dentro da ferra-
```

menta. Esse funcionamento pode ser visto no diagrama de seqüência da Figura 3.7.

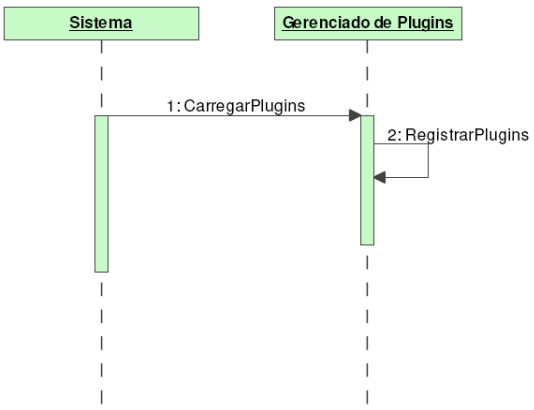

Figura 3.7: Diagrama de sequência do carregamento dos *plugins*

Cada menu contém os dados informados no *plugin* carregado, como o nome do *plugin*, nome do jogo a que é destinado e a extensão de arquivo reconhecida. Cada menu de *plugin* contém dois botões em forma de submenu, um para extrair os textos, no qual deverá ser informado o arquivo de recurso contendo os textos, e o último para salvar os textos traduzidos, devendo informar também o arquivo de recurso que irá receber os textos. Um exemplo pode ser visto na Figura 3.8, onde dois *plugins* foram carregados e adicionados no menu principal da ferramenta.

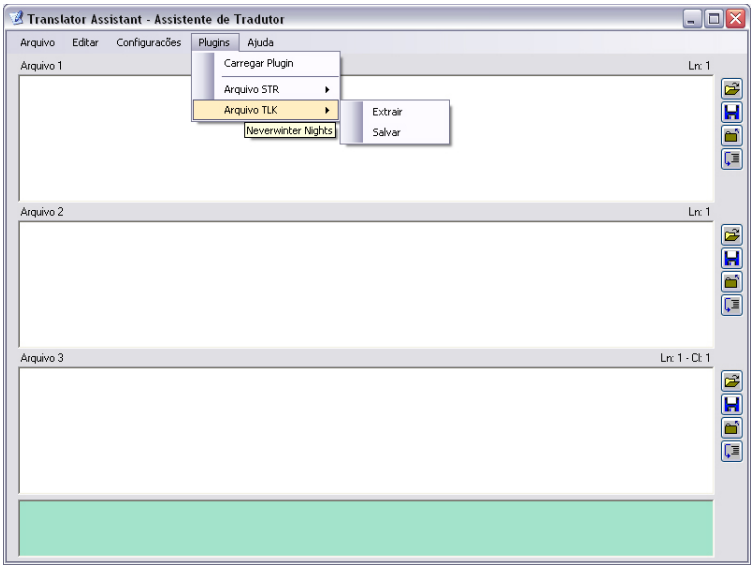

Figura 3.8: Tela inicial da ferramenta Translation Assistant com dois *plugins* carregados

3.2.4 A linguagem criada

Para que a ferramenta possa analisar diferentes arquivos de recurso e extrair os textos sem que seja necessária a recompilação, foi criada uma linguagem de programação simples, muito parecida com uma linguagem de montagem (Wikipedia, 2007b), que é escrita nos *plugins* e interpretada pela ferramenta. A linguagem contendo comandos básicos de operações matemáticas, desvios incondicionais e condicionais, atribuições e marcadores para criação de laços de execução. Cada palavra do comando, composto por um mnemônico e parâmetros, deve ser separada por um espaço. Os marcadores devem estar no início da linha, contendo dois-pontos no final do nome, separado por um espaço do comando. A lista de comandos está definida na Tabela 3.1.

Também foram criadas constantes para que sejam utilizadas no código. Tais constantes estão indicadas na Tabela 3.2. Essas constantes são substituídas por seus valores relacionados quando forem chamadas no código.

3.2.5 O interpretador de códigos da linguagem

O módulo de interpretação dos comandos de um *plugin* foi desenvolvido conforme a estrutura de um interpretador de linguagens, contendo um analisador léxico, um sintático e um semântico, sendo a análise sintática e semântica feitas no momento da execução de um comando.

A interpretação dos comandos começa com a criação de uma tabela de símbolos, que conterá as variáveis definidas pelo usuário utilizadas na interpretação. Após isso são extraídos todos os comandos da seção presente no *plugin*. A seção usada é definida dependendo da ação chamada (extrair ou salvar) no menu do *plugin*. Após a extração do código do *plugin*, os comandos são separados em uma lista contendo um comando por nó.

Após a montagem da lista, é feito uma varredura nos nós para identificar e catalogar todos os marcadores que existem. Cada marcador é salvo na tabela de símbolos identificando a linha que foi encontrado, para que quando for chamado através de um comando `GOTO`, a interpretação do código pule e continue na linha correta.

A próxima etapa é a interpretação dos comandos digitados. Um laço externo percorre cada nó da lista de comandos, separa todas as palavras contidas na string de comando, verifica qual o comando a ser executado e chama a função de interpretação correspondente.

Tabela 3.1: Comandos com seus parâmetros e respectivas ações

Comando	Parâmetros	Ação
GOTOPOS	Oper1	Avança o cabeçote de leitura/gravação até a posição informada.
GOTO	Marcador	Desvio incondicional para o marcador informado.
READ	NomeVar Oper1	Lê o número de bytes indicado por Oper1 e grava a informação na variável definida por NomeVar.
WRITE	NomeVar Oper1	Escreve no arquivo de recurso o valor definido pela variável NomeVar, tantos bytes informados em Oper1.
PRINT	Texto	Imprime o texto informado na área de trabalho. O operador + pode ser usado para concatenar *strings*.
IF	Oper1 Cond Oper2 GOTO Marcador	Desvio condicional. Testa Oper1 com Oper2 e desvia para o marcador informado caso verdadeiro.
DEFINE	NomeVar TipoVar ValInicial	Define uma variável com o nome NomeVar sendo do tipo TipoVar e, opcionalmente, contendo o valor inicial definido por ValInicial.
SET	NomeVar Oper1	Atribui o valor passado em Oper1 para a variável definida em NomeVar.
ADD	NomeVar Oper1 Oper2	Soma os valores Oper1 e Oper2 e atribui o resultado em NomeVar.
SUB	NomeVar Oper1 Oper2	Subtrai o valor Oper1 de Oper2 e atribui o resultado em NomeVar.
MUL	NomeVar Oper1 Oper2	Multiplica o valor Oper1 com Oper2 e atribui o resultado em NomeVar.
DIV	NomeVar Oper1 Oper2	Divide o valor Oper1 com Oper2 e atribui o resultado em NomeVar. Caso Oper2 seja 0, é exibido um erro.
SPLIT	NomeVar Oper1	Separa a *string* definida pela variável NomeVar utilizando como caracter de quebra o valor definido em Oper1.
SCAN	NomeVar	Pega o texto de uma linha da área de entrada e armazena na variável definida em NomeVar.
GETVAL	NomeVar Oper1	Obtém o valor da posição Oper1 do vetor criado com o comando SPLIT e armazena na variável NomeVar.
STRLEN	NomeVar1 NomeVar2	Guarda o tamanho da *string* indicada pela variável NomeVar1 no valor da segunda variável NomeVar2.

Tabela 3.2: Constantes e seus respectivos valores

Constante	Valor referente
NEW_LINE	Insere um caracter de nova linha. Utilizado em *strings*.
TAB	Insere um caracter de tabulação. Utilizado em *strings*.
NUM_LINES	Informa o número total de linhas da área de trabalho que contém os textos a serem incorporados. O valor é um inteiro.
ACT_LINE	Informa a linha atual que foi captada pelo comando `SCAN`. O valor é um inteiro.
FILE_SIZE	Informa o tamanho do arquivo de recurso aberto para extração ou incorporação dos textos. O valor é um inteiro longo.
POSITION	Informa a posição atual do cursor de leitura e escrita no arquivo de recurso aberto. O valor é um inteiro longo.

Dentro da função de execução do comando é verificado se os parâmetros foram passados corretamente. Caso algum parâmetro necessário não tenha sido passado, é mostrado um erro e a execução é interrompida. O funcionamento resumido pode ser visto na Figura 3.9.

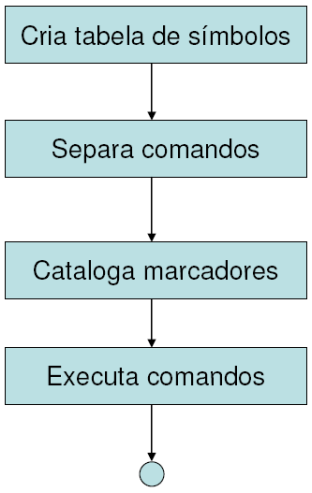

Figura 3.9: Fluxograma de funcionamento do interpretador

3.2.6 O módulo de edição de textos

Nesta seção é mostrado como funciona o sistema de manipulação de textos. Existem três áreas de trabalho na ferramenta. Cada área de trabalho serve para mostrar o texto de

um arquivo. A idéia de possuir três áreas de trabalho é para que duas possam ser usadas contendo textos em outra língua, onde serão comparados os textos para que, na terceira área, seja escrito ou editado o texto que será posteriormente adicionado no arquivo de recurso do jogo. É possível ver uma comparação entre textos e a interface da ferramenta na Figura 3.10.

Existe uma área de trabalho fixa na parte inferior do programa. Esta área de trabalho, que tem uma cor diferente das outras três, serve para marcar as linhas atualmente ativas, ou seja, as linhas que contém o cursor, nas áreas de trabalho superiores. Se as três áreas de trabalho superiores estiverem exibindo textos, cada linha ativa em tal área é mostrado nessa área fixa, sendo assim especialmente útil para comparação das linhas. Nesta área também é possível modificar o texto apresentado. Nesse caso, a modificação irá alterar apenas a linha ativa da área superior correspondente.

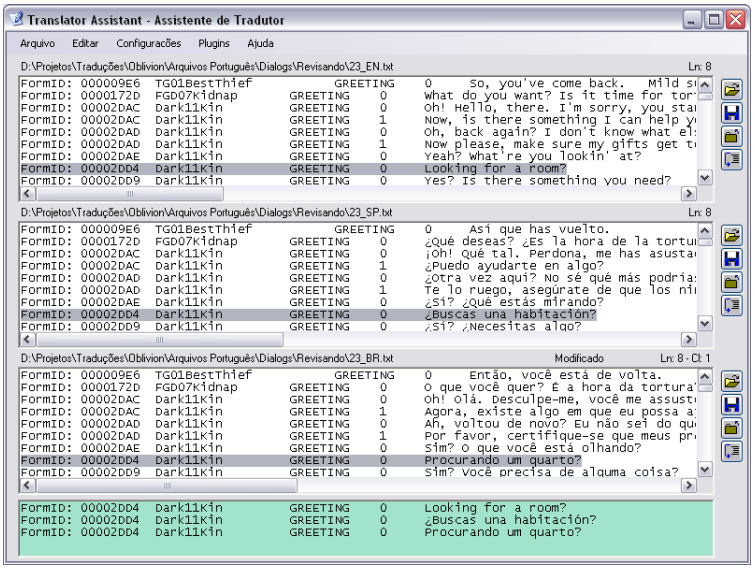

Figura 3.10: Comparação de três arquivos em línguas diferentes

3.3 Exemplo de uso da ferramenta

Nesta seção é descrito como e em qual momento do processo de tradução a ferramenta desenvolvida é utilizada. Foi utilizado o arquivo de recuros contendo os diálogos do jogo *Neverwinter Nights*, desenvolvido pela empresa *Bioware*.

3.3.1 O processo de extração de textos

O primeiro passo necessário para traduzir um jogo de computador é identificar quais são os arquivos de recurso que contêm os textos que serão traduzidos. Essa identificação é

feita analisando visualmente cada arquivo presente nas pastas do jogo. Existem arquivos com nomes significativos, como "Dialogs.tlk"ou "Texts.bin", porém não é sempre que se encontram arquivos com tal nomenclatura. Para tal tarefa, o uso de um editor hexadecimal é mais adequado, visto que é possível visualizar os bytes que formam o arquivo.

Após identificado os arquivos que possuem os textos, a próxima etapa é descobrir como está estruturado cada arquivo de recurso identificado. Isso é necessário para que seja escrito o código correto para a extração dos textos, pois as informações devem ser interpretadas corretamente para que não seja perdido textos e dados importantes. Cada empresa possui a sua maneira de estruturar os arquivos de recursos dos jogos. Algumas utilizam formatos conhecidos, como o formato XML. Outros já estruturam seu arquivos utilizando uma forma particular, podendo algumas vezes estarem criptografados, dificultando assim a tradução dos mesmos. Outros jogos podem também utilizar diferentes formas de estruturas para cada arquivo de recurso.

A Figura 3.11 mostra o arquivo de diálogos do jogo *Neverwinter Nights* chamado "Dialogs.tlk". Após a análise visual do arquivo através da ferramenta *XVI32*, descobriu-se que o cabeçalho do arquivo é formado por vinte bytes contendo o nome e versão do arquivo, o número de registros e a posição onde começa a seção dos textos.

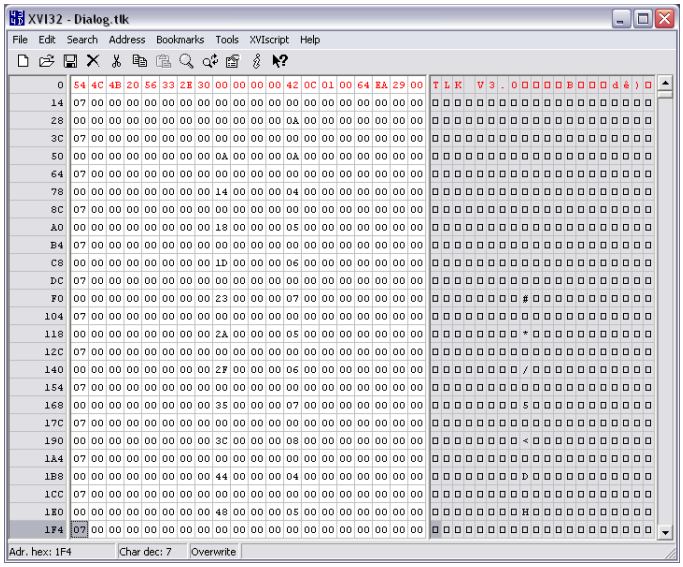

Figura 3.11: Arquivo de diálogos do jogo *Neverwinter Nights* aberto na ferramenta *XVI32*

Para extrair a informação do cabeçalho, foi utilizado o código da Figura 3.12 escrito na linguagem criada. No início são definidas as variáveis que irão armazenar as informações extraídas, utilizando o comando `DEFINE`. Logo após as declarações, o cursor de leitura

do arquivo de recurso é posicionado no início do arquivo através do comando GOTOPOS, e após são lidos as informações e gravadas nas variáveis através dos comandos READ. Após a extração das informações, os valores das variáveis são impressos na primeira área de trabalho da ferramenta.

```
DEFINE szNomeArquivo STRING
DEFINE lLinguagem LONG $0
DEFINE lNumRegistros LONG $0
DEFINE lEndInicioTextos LONG $0
GOTOPOS $0
READ szNomeArquivo $8
READ lLinguagem $4
READ lNumRegistros $4
READ lEndInicioTextos $4
PRINT szNomeArquivo+TAB+lLinguagem+TAB
PRINT lNumRegistros+TAB+lEndInicioTextos+NEW_LINE
```

Figura 3.12: Código para extração do cabeçalho do arquivo de recurso

Logo após o cabeçalho do arquivo iniciam os registros dos textos. Cada registro é composto por 40 bytes, sendo os primeiros 4 bytes a flag de indicação do registro. Os 16 bytes seguintes formam o nome do arquivo de som referente ao texto. Os próximos 4 bytes significam o volume do som e os 4 bytes seguintes o volume do tom. Os próximos 4 bytes informam a posição inicial do texto, referente à posição inicial da seção dos textos, e os últimos 4 bytes informam o tamanho do texto do registro.

Para a extração dos registros, foi utilizado o código apresentado na Figura 3.14. O código começa declarando variáveis através do comando DEFINE, depois executa um teste com o comando IF para saber se já pegou todos os registros, pulando para o final caso verdadeiro. Para cada registro, são lidas as informações e armazenadas nas variáveis através dos comandos READ.

Logo após, é feito o cálculo para descobrir onde inicia o texto referente ao registro, efetuando o deslocamento do cabeçote de leitura até a posição calculada e, após o armazenamento do texto, o cabeçote volta para a posição onde estava e o índice dos registros acrescido. No final, é executado um desvio incondicional para o teste inicial através do comando GOTO. Como resultado desta execução, têm-se os textos extraídos na tela da ferramenta (Figura 3.13).

Apesar dos códigos estarem separados, para um melhor entendimento, eles são escritos juntos na seção do código de leitura do *plugin*.

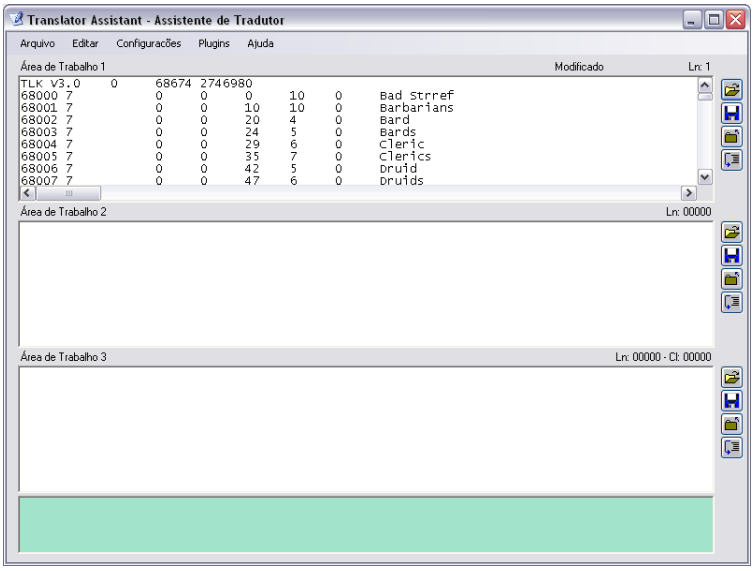

Figura 3.13: Resultado da execução do código apresentado na área de trabalho superior da ferramenta

3.3.2 A tradução dos textos extraídos

Após a extração dos textos dos arquivos de recursos, o próximo passo é traduzir os textos para a língua alvo, como por exemplo o português do Brasil. Nessa etapa é comum utilizar um editor de textos para a tradução. Os mais usados são o *Word* da Microsoft ou o *Bloco de Notas* que acompanha o sistema operacional Windows. Por ser mais simples e prática, a tradução na ferramenta Translator Assistant é mais recomendada, visto que os textos extraídos já estão prontos para serem traduzidos com facilidade na primeira área de trabalho da ferramenta. Algumas ferramentas auxiliares também são utilizadas nessa etapa, como dicionários eletrônicos e tradutores *online*.

Também é possível traduzir os textos no próprio arquivo de recurso utilizando o editor hexadecimal, mas esse tipo de tradução é recomendada apenas como última alternativa, pois apresenta várias limitações. A principal delas é que os textos não podem ter tamanho maior que o texto original. Se uma frase possui 28 bytes de tamanho em inglês, por exemplo, o texto traduzido deverá ter os mesmos 28 bytes de tamanho. Algumas frases pequenas na língua original deverão ser simplificadas e, em muitos casos, palavras abreviadas deverão ser usadas para que o tamanho continue o mesmo. Se o arquivo de recurso possuir a informação do tamanho do texto e o byte responsável por armazenar essa informação for descoberto, então esse problema acaba não existindo, mas mesmo assim, a tradução se torna cansativa, pois para cada texto traduzido, a informação do tamanho referente ao mesmo deve ser alterada.

```
        DEFINE lFlagRegistro LONG $0
        DEFINE szNomeArqSom STRING
        DEFINE lVariacaoVolume LONG $0
        DEFINE lVariacaoTom LONG $0
        DEFINE lPosInicialTexto LONG $0
        DEFINE lTamTexto LONG $0
        DEFINE lTamSom LONG $0
        DEFINE szTexto STRING
        DEFINE lIdx LONG $0
        DEFINE lUltimaPos LONG $0
        DEFINE lPosTemp LONG $0

__Ini:  IF lIdx > lNumRegistros GOTO __End
        READ lFlagRegistro $4
        READ szNomeArqSom $16
        READ lVariacaoVolume $4
        READ lVariacaoTom $4
        READ lPosInicialTexto $4
        READ lTamTexto $4
        READ lTamSom $4
        SET lUltimaPos POSITION
        SET lPosTemp $0
        ADD lPosTemp lEndInicioTextos lPosInicialTexto
        GOTOPOS lPosTemp
        READ szTexto lTamTexto
        PRINT lIdx+TAB+lFlagRegistro+TAB+szNomeArqSom+TAB
        PRINT lVariacaoVolume+TAB+lVariacaoTom+TAB+lPosInicialTexto+TAB
        PRINT lTamTexto+TAB+lTamSom+TAB+szTexto+NEW_LINE
        ADD lIdx lIdx $1
        GOTOPOS lUltimaPos
        GOTO __Ini
__End:
```

Figura 3.14: Código para extração dos registros do arquivo de recurso

Outro problema encontrado na tradução utilizando editores hexadecimais é a dificuldade em escrever textos. Como esses editores são utilizados com a finalidade de manipular bytes diretamente, escrever textos se torna um trabalho cansativo, propenso a muitos erros de ortografia e escrita de textos em lugares errados. Também não é possível utilizar ferramentas auxiliares, como dicionários eletrônicos, já que as informações são apresentadas nesses editores como seqüências de bytes.

3.3.3 A incorporação dos textos traduzidos

A última etapa da tradução é a reincorporação dos textos traduzidos e revisados nos arquivos de recurso, na mesma posição de onde foram extraídos, seguindo a estrutura do arquivo. Também são alterados os outros bytes que descrevem as informações a respeito dos textos alterados, como, por exemplo, o tamanho e a posição inicial de cada frase, o

tamanho do arquivo e o início da seção dos textos.

O sistema de incorporação dos textos traduzidos está funcionando parcialmente, visto que é necessário a implementação de novos comandos na linguagem para manipulação de *strings*. Mas, para textos pequenos em que os diálogos estão um em cada linha, como na Figura 3.10, é possível executar a incorporação. Na Figura 3.15 é possível observar como seria o código escrito na linguagem para a incorporação dos textos extraídos do arquivo de recurso do jogo *Neverwinter Nights*.

```
        DEFINE lFlagRegistro LONG $0
        DEFINE szNomeArqSom STRING
        DEFINE lVariacaoVolume LONG $0
        DEFINE lVariacaoTom LONG $0
        DEFINE lPosInicialTexto LONG $0
        DEFINE lTamTexto LONG $0
        DEFINE lTamSom LONG $0
        DEFINE szTexto STRING

        DEFINE lIdx LONG $0
        DEFINE lUltimaPos LONG $0
        DEFINE lPosTemp LONG $0

__Ini:  IF CUR_LINE > NUM_LINES GOTO __End
        SCAN szLinha
        SPLIT szLinha TAB
        GETVAL iIdx 1
        GETVAL lFlagRegistro 2
        GETVAL szNomeSom 3
        GETVAL lVariacaoVolume 4
        GETVAL lVariacaoTom 5
        GETVAL lPosInicialTexto
        GETVAL lTamTexo
        GETVAL lTamSom
        SCAN szLinha
        SPLI NEW_LINE
        GETVAL szTexto
        WRITE lFlagRegistro $4
        WRITE szNomeSom $16
        WRITE lVariacaoVolume $4
        WRITE lVariacaoTom $4
        WRITE lPosInicialTexto $4
        STRLEN lTamTexto szTexto
        WRITE lTamTexto $4
        WRITE lTamSom $4
        SET lUltimaPos POSITION
        GOTOPOS lUltimaPos
        WRITE szTexto lTamTexto
        GOTO __Ini
__End:
```

Figura 3.15: Código para a incorporação dos textos no arquivo de recurso

4 CONCLUSÃO

Este trabalho apresentou o desenvolvimento de uma ferramenta para auxílio na tradução de jogos de computador, possibilitando aos tradutores extrairem, traduzirem e incorporarem os textos traduzidos de volta aos arquivos de recurso dos jogos.

A seção 2 avaliou outras ferramentas que são utilizadas na tarefa de tradução, expondo seus pontos fortes e fracos, e no final foi feita uma comparação com a ferramenta desenvolvida, mostrando as funcionalidades que foram incluídas e aquelas que existem apenas em outras ferramentas.

A criação da linguagem e do módulo de interpretação foi um ponto importante para a ferramenta, pois possibilitou que vários arquivos de recurso diferentes fossem abertos e seus dados extraídos, não prendendo a ferramenta a apenas um tipo de estrutura.

A interface criada também facilitou que a comparação de textos, antes feita por ferramentas caras ou pela utilização de várias janelas abertas, fosse realizada em apenas uma janela central, comparando até 3 arquivos ao mesmo tempo, linha por linha.

Diversas funcionalidades podem ser incluídas na ferramenta para ampliar e melhorar a sua utilização na tradução de jogos. A seção 4.1 apresenta algumas melhorias que podem ser feitas na ferramenta.

4.1 Trabalhos futuros propostos

São váris as possibilidades de mudanças e melhorias da ferramenta. Abaixo são destacadas e comentadas algumas sugestões para futuros trabalhos que podem tornar a ferramenta mais robusta e com maior desempenho.

- Adicionar a possibilidade de analisar as estruturas dos arquivos de recursos através de um visualizador em hexadecimal. Tal funcionalidade ampliaria a utilização dessa ferramenta no processo de tradução de jogos.

- Melhorar o sistema de incorporação dos textos para que funcione com casos mais complexos de diálogos, como diálogos separados em mais de uma linha ou arquivo. Tal melhoria ajudaria no trabalho em conjunto, facilitando a incorporação de textos através de vários arquivos, cada um sendo traduzido por uma pessoa diferente.

- Incorporar os códigos utilizados em *plugins* dentro da ferramenta, para as extensões de arquivos de recurso mais comuns. Isso facilitaria o uso da ferramenta pela primeira vez, sem a necessidade de procurar um plugin para o arquivo já reconhecido.

- Adicionar um sistema de automação para a criação de *plugins*, com uma interface visual simples e, ao mesmo tempo, robusta.

REFERÊNCIAS

Alexa. **Traffic Details for**: gamevicio.com.br/. Disponível em: <http://www.alexa.com/data/details/traffic_details/gamevicio.com.br>.

AZEVEDO, T. **Jogo original no Brasil**: progressos significativos. Disponível em: <http://jogos.uol.com.br/reportagens/ultnot/ult2240u118.jhtm>.

BERNAL-MERINO, M. A. **Localization and the Cultural Concept of Play**. Disponível em: <http://www.gamecareerguide.com/features/454/localization_and_the_cultural_.php >.

BOS, B. **XML in 10 points**. Disponível em: <http://www.w3.org/XML/1999/XML-in-10-points>.

BRAUN, D. **Vendas de PCs no Brasil crescem 23% e chegam a 10,1 milhões em 2007**. Disponível em: <http://idgnow.uol.com.br/computacao_pessoal/2007/12/06/idgnoticia.2007-12-06.5643442136/>.

CHANDLER, H. **The Game Localization Handbook (Game Development Series)**. Rockland, MA, USA: Charles River Media, Inc., 2004.

DEITEL, H. M.; DEITEL, P. J. **Java How to Program**. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

ECMA International. **C# Language Specification**. Disponível em: <http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-334.pdf>.

FRAUCHES, S. **Aperte o botão início (ou 'Eu jogo em português')**. Disponí-

vel em: <http://www.overmundo.com.br/overblog/aperte-o-botao-inicio-ou-eu-jogo-em-portugues>.

MAAS, C. **Free Hex Editor XVI32**. Disponível em: <http://www.chmaas.handshake.de/delphi/freeware/xvi32/xvi32.htm>.

MARRIOT, M. **Jogos de computador mostram sinais de retorno**. Disponível em: <http://ultimosegundo.ig.com.br/new_york_times/2007/04/23/jogos_de_computador_mostram_sinais_de_retorno__762187.html>.

Ministério da Educação e do Desporto. Secretaria de Educação Fundamental. **Parâmetros curriculares nacionais** : terceiro e quarto ciclos do ensino fundamental: língua estrangeira. Brasília: [s.n.], 1998. Disponível em: <http://portal.mec.gov.br/seb/arquivos/pdf/pcn_estrangeira.pdf>.

SCHILDT, H. **C**: the complete reference. Berkeley, CA, USA: Osborne/McGraw-Hill, 1985.

Scooter Software. **Beyoud Compare**. Disponível em: <http://www.scootersoftware.com/compare-text-files.php>.

STROUSTRUP, B. **The C++ programming language (2nd ed.)**. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1991.

The XeNTaX Foundation. **About**. Disponível em: <http://www.xentax.com/?page_id=2>.

TROELSEN, A. **Pro C# 2008 and the .NET 3.5 Platform, Fourth Edition (Pro Series)**. Berkely, CA, USA: Apress, 2007.

WATSON, M. **Game Extractor**. Disponível em: <http://www.watto.org/extract/info.html>.

Wikipedia. **Fan translation of video games**. Disponível em: <http://en.wikipedia.org/wiki/Fan_translation_of_computer_and_video_games>.

Wikipedia. **Linguagem de Montagem**. Disponível em: <http://pt.wikipedia.org/wiki/Linguagem_de_montagem>.

YUNKER, J. **Video Game Localization Becoming Big Business**. Disponível em: <http://goingglobal.corante.com/archives/2005/12/06/video_game_localization_becoming_big_business.php>.

APÊNDICE A - O IDE VISUAL C# 2008 EXPRESS EDITION

Para o desenvolvimento da ferramenta foi utilizada o ambiente de desenvolvimento integrado (IDE) da Microsoft chamado *Visual C# 2008 Express Edition*, que é uma versão mais simples do ambiente *Visual Studio*, caracterizado pelo nome Express Edition, sendo esse ambiente totalmente gratuíto.

Possui janelas laterais de propriedades, mostradas na Figura 4.1, e informações sobre o projeto, entre outras, que podem ser configuradas como janelas flutuantes ou fixas. É possível abrir vários arquivos do projeto na tela principal, e conforme o tipo do arquivo aberto, a ferramenta se auto-adapta, mostrando as janelas referentes ao tipo de arquivo (design ou código-fonte). Contém também uma janela de objetos visuais, possibilitando a rápida criação de formulários contendo botões, ícones e áreas de texto.

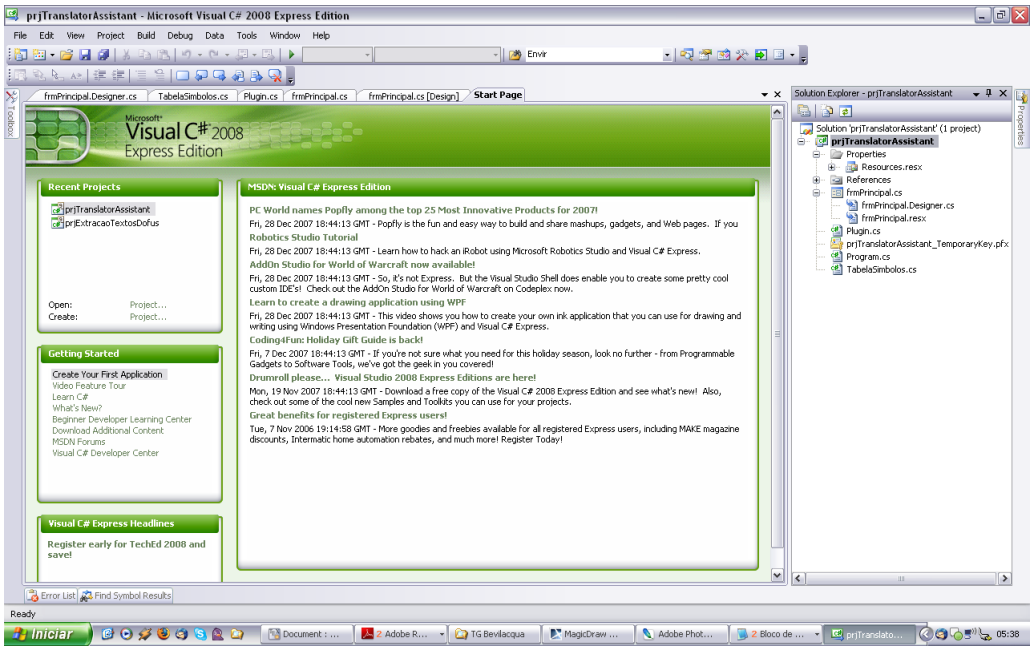

Figura 4.1: Janela inicial do IDE Visual C# Express Edition

Possui depurador integrado, facilitando a correção em tempo de execução do código

escrito. Seu menu principal possui botões para a rápida criação de esqueletos de aplicações, sendo tanto aplicações contendo janelas como aplicações em console. Na Figura 4.2 aparece um exemplo de construção de formulários (a tela inicial da ferramenta Translator Assistant). É possível ver o menu de objetos à esquerda, contendo todos os objetos visuais que podem ser adicionados ao formulário.

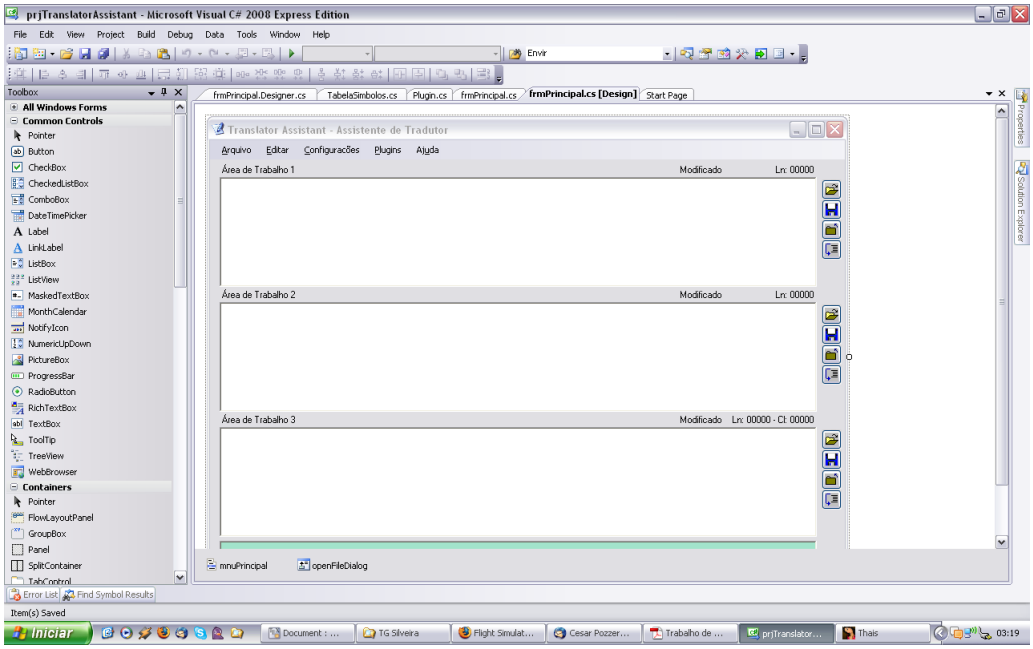

Figura 4.2: Construindo formulários com o IDE Visual C# Express Edition

APÊNDICE B - EXEMPLO DE CÓDIGO-FONTE EM C#

O código-fonte da Figura 4.3 é um exemplo de código-fonte em C#. É possivel observar a existência de uma enumeração, chamada *TipoVariavel* e uma classe, chamada *ItemTabelaSimbolos*.

Uma das características da linguagem C#, as propriedades, podem ser vistas através da classe *ItemTabelaSimbolos*. As propriedades *Valor*, *Nome* e *Tipo* são utilizadas no código como variáveis, mas executam o código correspondente quando é atribuido um valor ou quando a propriedade é lida.

Também é possível observar o formato utilizado pela linguagem para geração da documentação do código. São utilizandas TAGs em XML para a definição das documentações. Essas TAGs são analisadas pelo gerador de documentação e no final é criado um arquivo contendo as informações definidas, em um arquivo HTML ou outro que o usuário desejar.

Observa-se que o código é muito parecido com as linguagens JAVA e C++. De fato, a linguagem C# foi criada baseando-se nessas citadas, além de herdar características de outras linguagens, como a facilidade das propriedades da linguagem Delphi.

```
using System;
using System.Collections.Generic;
using System.Text;

namespace prjTranslatorAssistant {

    enum TipoVariavel {
        LONG,
        DOUBLE,
        STRING,
        LABEL
    }


    class ItemTabelaSimbolos {
        private string m_szNome = "";
        private object m_Valor = null;
        private TipoVariavel m_Tipo = TipoVariavel.LONG;

        /// <summary>
        /// Construtor
        /// </summary>
        /// <param name="a_szNome">Nome completo do arquivo de plugin
                </param>
        /// <param name="a_Tipo">Nome completo do arquivo a ser
                extraido os dados</param>
        /// <param name="a_Valor">Área onde os dados extraidos serÃ£o
                mostrados</param>
        public ItemTabelaSimbolos(string a_szNome, TipoVariavel a_Tipo,
                object a_Valor) {
            this.m_szNome = a_szNome;
            this.m_Tipo = a_Tipo;
            this.m_Valor = a_Valor;
        }

        public object Valor {
            get { return m_Valor; }
            set { m_Valor = value; }
        }

        public string Nome {
            get { return m_szNome; }
            set { m_szNome = value; }
        }

        public TipoVariavel Tipo {
            get { return m_Tipo; }
            set { m_Tipo = value; }
        }
    }
}
```

Figura 4.3: Exemplo de código-fonte em C#