

**UNIVERSIDADE FEDERAL DE SANTA CATARINA**  
**DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA**  
**CURSO DE BACHARELADO EM CIÊNCIAS DA COMPUTAÇÃO**

**TRABALHO DE CONCLUSÃO DE CURSO**



**Projeto e Implementação de um Game  
estilo Adventure**

**Autores**

Victor Simon Lee  
Rafael Silva e Souza

**Orientador**

Renato Cislighi

Florianópolis, Junho de 2003.

UNIVERSIDADE FEDERAL DE SANTA CATARINA  
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

TRABALHO DE CONCLUSÃO DE CURSO

Título: Projeto e Implementação de um Game estilo Adventure

Autores: Victor Simon Lee  
Rafael Silva e Souza

Submetido ao corpo docente do Departamento de Informática e Estatística da Universidade Federal de Santa Catarina como um dos requisitos para a obtenção do título de Bacharel em Ciências da Computação.

Orientador:

---

Prof. Mst. Renato Cislaghi

Banca Examinadora:

---

Prof. Dr. José Mazzucco Jr.

---

Prof. Dr. Roberto Willrich

---

Bel. Rafael Leite

# Sumário

Resumo .....	1
Abstract.....	3
Lista de Abreviaturas e Siglas .....	5
Lista de Figuras .....	7
1. Introdução .....	9
1.1 Apresentação .....	10
1.2 Justificativas .....	12
1.3 Objetivos.....	13
1.4 Metodologia .....	14
2. Tecnologias Necessárias .....	16
2.1 Concurrent Version System – CVS.....	21
3. Implementação do Sistema.....	24
3.1 Requisitos Funcionais .....	25
3.2 Modelagem Conceitual.....	27
3.3 Arquitetura do sistema .....	31
3.3.1 Arquivo de Configuração .....	32
3.3.2 Game Loop .....	35
3.4 O Interpretador de Comandos .....	39
3.5 Algoritmo para busca do caminho.....	47
3.5.1 Algoritmos Existentes .....	48
3.5.2 Algoritmo Implementado.....	50
3.6 Telas Virtuais .....	53
3.7 Níveis de Profundidade .....	57
3.8 Esquema de Transições de Tela .....	59
3.9 Esquema de Objetivos .....	60
3.10 Esquema de Eventos .....	62
4. Conclusão.....	63
5. Referências Bibliográficas .....	66
6. Anexos.....	69
6.1 As 10 regras no desenvolvimento de Games.....	70
6.2 Game Design Document .....	71

6.3 Código-Fonte .....	81
6.4 Artigo sobre o trabalho .....	134

# **Resumo**

## **Resumo**

O presente trabalho de conclusão do curso de Ciências da Computação trata do projeto e implementação de um software de entretenimento eletrônico, mais especificamente, um game que segue o estilo "Adventure".

O principal objetivo deste trabalho consiste da aplicação dos conhecimentos adquiridos durante o curso de graduação, em uma aplicação real de tamanho e complexidades consideráveis.

# **Abstract**

## **Abstract**

The present work of conclusion of the Computer Science course deals with the project and implementation of a electronic entertainment software, more specifically, a game, which follows the adventure style.

The main objective of this work consists on the application of the knowledge acquired during the graduation course, in a real application with considerable size and complexity.



## **Lista de Abreviaturas e Siglas**

## **Lista de Abreviaturas e Siglas**

API: Application Programming Interface

CGA: Color Graphics Array

CVS: Concurrent Version System

GDD: Game Design Document

RAD: Rapid Application Development

GPL: General Public License

GDB: GNU Debugger

FSM: Finite State Machine

GAS: Gerador de Analisadores Sintáticos

XML: Extensible Markup Language

NPC: Non-Player Character

## **Lista de Figuras**

## Lista de Figuras

Figura 2.1: Software para trabalhar com CVS.....	23
Figura 3.1: Diagrama de Relacionamento entre as classes do sistema....	27
Figura 3.2: Sistema genérico. ....	31
Figura 3.3: Documento XML para arquivo de configuração. ....	34
Figura 3.4: Game Loop tradicional. ....	35
Figura 3.5: Gramática utilizada para a interpretação dos comandos.....	40
Figura 3.6: Autômato Finito utilizado na análise léxica. ....	41
Figura 3.7.: Interação entre classes para a análise sintática. ....	41
Figura 3.8: Ações a serem executadas pela classe Tjogo. ....	46
Figura 3.9: Algoritmo simples para busca do caminho ....	47
Figura 3.10: Algoritmo não consegue achar a saída.....	49
Figura 3.11: Algoritmo de Busca em Largura.....	50
Figura 3.12: Algoritmo implementado. ....	51
Figura 3.13: Estratégia adotada para a escolha de uma nova direção.....	51
Figura 3.14: Software para validar algoritmo de busca do caminho. ....	53
Figura 3.15: Exemplo de cenário do game. ....	53
Figura 3.16: Exemplo de tela "atual".....	54
Figura 3.17: Exemplo de Tela "interna". ....	55
Figura 4.18: Exemplo de Tela virtual sobre a atual.....	56
Figura 3.19: Dependências entre objetivos. ....	60

# **1. Introdução**

## 1.1 Apresentação

A história dos games eletrônicos divide-se em vários períodos. Períodos estes caracterizados pela complexidade dos games e que eram delimitadas, principalmente, pelo que os computadores podiam fazer e pela criatividade dos seus criadores.

Bem antes de placas aceleradoras 3D, bem antes de games multi-jogadores, numa época em que gráficos 2D com 256 cores eram considerados a última tecnologia e a maioria dos jogadores ainda sacrificava seus olhos nas quatro cores de um monitor CGA, os games "Adventure" dominavam uma grande parcela do mercado.

Não mais em modo texto como nos primórdios, onde o jogador lia uma descrição do cenário em que se encontrava e digitava que ações tomar, mas um pouco mais avançados, com grande parte da tela mostrando o cenário e o personagem que o jogador incorporava. Os movimentos do personagem eram controlados pelo teclado (mais tarde, por cliques do mouse) e suas ações continuavam sendo digitadas (isso também foi modificado com o passar dos anos, dispensando completamente o teclado).

Numa época em que os games não tinham enredos bem definidos, os games "Adventure" criavam enredos mais complexos, que se dividiam em sequências. Séries inteiras como os títulos King's Quest, Space Quest, Police Quest, Leisure Suit Larry, da produtora Sierra, ou The Secret Of Monkey Island da Lucasfilm Games fizeram história.

Esse trabalho de conclusão do curso de Ciências da Computação é uma volta aqueles tempos. Uma tentativa de recriar aqueles games

antigos em estilo Adventure que marcaram época.

## 1.2 Justificativas

Quando a dúvida do que fazer para o trabalho de conclusão de curso surgiu aos membros da equipe, a solução não demorou muito a aparecer.

Tendo sido assíduos jogadores de games no estilo adventure, a idéia dos membros foi imediatamente a de implementar um game que seguisse a mesma linha. Mas não podia ser um game qualquer, deveria ser um game que empregasse o máximo possível de conteúdo das disciplinas que foram vistas no decorrer do curso.

Então, um game seguindo o estilo "Adventure" foi o escolhido pois foi esse tipo de game era o preferido pela equipe na época em que a mesma começou a se aprofundar no mundo da informática, e o que sempre despertou mais curiosidade sobre como tais games eram desenvolvidos.

Empregando desde simples estruturas de dados até técnicas para a interpretação de comandos, a implementação desse game não só satisfaz os membros da equipe, mas também demonstra uma série de conhecimentos adquiridos no decorrer do curso.



## **1.3 Objetivos**

Este trabalho tem como objetivos:

- o aprendizado das técnicas utilizadas para o projeto e implementação de games de computador no estilo "Adventure", assim como o aprendizado das ferramentas necessárias;
- a elaboração de um Game Design Document, o projeto de um game no estilo "Adventure" e sua consequente implementação;

## 1.4 Metodologia

Para alcançarmos os objetivos propostos neste trabalho, adotamos a seguinte metodologia de trabalho:

- estudos sobre a biblioteca gráfica Allegro;

A biblioteca gráfica Allegro (<http://www.allegro.cc>) consiste em uma API (Application Programming Interface) multimídia de propósito geral. Um estudo a respeito desta API, suas funcionalidades e a realização de alguns experimentos com a mesma são um passo fundamental dentro de nosso projeto.

- estudos e elaboração de um "Game Design Document";

O projeto e implementação de games de computador envolve uma etapa denominada "Game Design Document" (GDD), onde são especificadas de maneira informal os vários aspectos do game a ser desenvolvido (tais como requisitos funcionais, aspectos artísticos, enredo, etc).

Em projetos de games de computador, o papel de um GDD é fundamental para que a equipe envolvida no projeto (artistas, músicos, programadores, etc) tenham todos a mesma idéia a respeito do software que está sendo desenvolvido. Isto é, em um projeto com diversos colaboradores, é comum encontrar-se pessoas com idéias e opiniões diversas. Portanto, a existência de um GDD serve principalmente para que todos os envolvidos no projeto saibam o que e como deve ser feito, e os objetivos que devem ser alcançados pela equipe.

- elaboração do projeto do sistema;

Nesta etapa de nosso trabalho é realizado o projeto do sistema em

si, levando-se em consideração aspectos computacionais tais como modelagem conceitual do sistema, diagramas necessários, etc.

- implementação e testes do sistema projetado;

Nesta última etapa de nosso trabalho é feita a codificação do game proposto seguindo-se as especificações elaboradas nas etapas anteriores.

## **2. Tecnologias Necessárias**

Várias tecnologias poderiam ser utilizadas para se alcançar os objetivos deste trabalho. Dentre as várias opções de linguagens escolhemos utilizar o C++.

### **Linguagem C++:**

O C++ é uma linguagem criada a partir da extensão do C, para incorporar comportamentos de linguagens orientadas a objetos. Qualquer programa funcional em C é um programa funcional em C++, mas o C++ traz muito mais recursos inerentes à programação orientada a objeto, como o encapsulamento e ocultação de dados, a herança e reutilização e o polimorfismo.

Com certeza um dos principais motivos da escolha do C++ por parte da equipe foi o fato de já termos um relacionamento de longa data com o C, e, após aprender sobre orientação de objetos no decorrer no curso, percebemos suas vantagens na criação de aplicações mais robustas e de fácil manutenção.

Outra vantagem do C++ é a sua grande difusão. Existem compiladores C++ em praticamente todas as plataformas existentes no mercado, o que dá portabilidade ao presente trabalho, sendo independente de plataforma.

Contudo, era necessário um ambiente de trabalho intuitivo e que livrasse a equipe de tarefas mecânicas. O ambiente escolhido foi o Bloodshed Dev-C++.

### **Ambiente Bloodshed Dev-C++:**

O Bloodshed Dev-C++ é um ambiente de trabalho distribuído sob a GNU GPL (General Public License), ou seja, é um software de código aberto, e traz muitos recursos. O Bloodshed Dev-C++ gerencia projetos, mostra árvores com as classes do projeto, tem um debugger integrado (utilizando o GDB, discutido adiante), um sistema de pacotes (com

atualizações automáticas), além de importar projetos do Microsoft Visual C++ e exportar seus próprios projetos para o Microsoft Visual C++.

Seu editor tem funcionalidades já comuns em vários outros produtos, como o Syntax Highlighting - diferentes partes do código fonte são mostradas com cores diferentes -, Code Completion - que mostra os membros de um objeto ao se digitar o nome do mesmo -, indentação facilitada e Bookmarks - que permite marcar linhas no código para rápida referência.

### **Debugger GDB:**

O GDB é o Debugger mais utilizado no mundo do código aberto. Muito poderoso, o GDB traz vários recursos para depurar um programa, também é um programa de código aberto e tem a vantagem de rodar nas mais diferentes plataformas.

Por ser um programa em modo texto o GDB não é o depurador mais fácil de se aprender. Felizmente o Bloodshed Dev-C++ faz uma bela integração com o GDB, criando uma interface intuitiva com o GDB, mas ao mesmo tempo sem limitar suas funcionalidades: há uma caixa onde se permite mandar comandos diretos para o GDB.

### **Biblioteca gráfica Allegro:**

Com o ambiente pronto para o desenvolvimento em C++, a equipe procurou por uma biblioteca para facilitar no desenvolvimento de um game.

Criada por Shawn Hargreaves e vários colaboradores, a Allegro é uma biblioteca para programação de games, de código aberto e multiplataforma - há versões para Windows, Unix, BeOS, QNX, entre outros.

A Allegro é de simples uso e expansível, através de módulos e

pacotes de extensão. Contempla desde operações básicas de desenho, como pintar pixéis, linhas, círculos, até outras operações não tão básicas, como desenhos com splines bézier e iluminação/translucência.

Permite acesso fácil a joysticks, teclado e mouse; lê e escreve arquivos comprimidos com o algoritmo LZSS; tem rotinas para criar interfaces gráficas com o usuário, entre outras funções.

Como se não bastasse, sua API é simples de ser aprendida, sem perder em flexibilidade, e o Bloodshed Dev-C++ a tem como um pacote para fácil instalação e linkagem.

Tudo isso fez com que a Allegro fosse a biblioteca perfeita para a implementação desse trabalho, que passou a ser totalmente feito com ferramentas de código aberto e multiplataforma - o que o transforma num trabalho de código aberto e também multiplataforma.

Infelizmente, apesar do trabalho em si ser totalmente criado com ferramentas de código aberto, certos passos no processo de sua criação foram feitos em programas comerciais.

Para a implementação de aplicações que não poderiam tomar muito tempo de projeto, como os testes preliminares (por exemplo, o teste do algoritmo de busca do caminho) e a criação de utilitários para o game (como o editor de telas internos), viu-se a necessidade de utilizar o Borland Delphi, que é um RAD (Rapid Application Development - Desenvolvimento Rápido de Aplicações) baseado em Object Pascal.

A principal vantagem é que poucas linhas de código precisam ser incorporadas a um projeto, sendo que a maioria das ações é realizada num ambiente intuitivo aponte-e-clique.

Infelizmente a extrema facilidade de criação de uma aplicação em

Delphi traz consigo certas desvantagens. Uma delas sendo o preço da aplicação - um objetivo implícito desse trabalho é utilizar ferramentas de código aberto para gerar um programa de código aberto.

Mas a principal desvantagem de se utilizar o Delphi é a perda do código livre de plataforma. Mesmo existindo agora o Kylix, que permite que projetos possam ser compilados em uma plataforma como o Linux, não houve a possibilidade de testar essas aplicações-teste em tal ambiente para garantir sua compatibilidade.

Felizmente, as aplicações-teste e os utilitários criados em Delphi não comprometem o trabalho como um todo, por serem apenas um artifício na facilitação da compreensão do problema em mãos e/ou de tarefas repetitivas.



## 2.1 Concurrent Version System – CVS

O CVS é um sistema de controle de versões de arquivos muito difundido entre a comunidade do código aberto (Open Source). O sistema consiste em guardar arquivos, assim como todo o histórico de modificações nos mesmos, de um modo que vários desenvolvedores possam trabalhar nos mesmos sem criar problemas para os demais.

O sistema funciona da seguinte forma: os arquivos são adicionados ao repositório, onde ficam à disposição dos desenvolvedores. No caso desse projeto foram adicionados ao repositório não somente os arquivos com o código-fonte do game, mas também os arquivos com documentações, designs e os diferentes capítulos dessa monografia.

Esses arquivos permanecem visíveis em modo de leitura para todos os desenvolvedores. Ao aparecer a necessidade de fazer alguma mudança num arquivo, o desenvolvedor deve fazer um check-out do arquivo, o que faz com que o sistema forneça uma cópia de leitura/escrita ao desenvolvedor, e "tranque" o arquivo, não permitindo a mais nenhum desenvolvedor realizar o check-out do mesmo. Apesar disso, a cópia de leitura não alterada continua visível a todos os demais desenvolvedores. Isso faz com que um desenvolvedor possa trabalhar num determinado arquivo sem inserir erros de compilação no projeto de outros desenvolvedores.

Quando o desenvolvedor estiver satisfeito com as mudanças, ele faz um check-in do arquivo. Nessa operação, o CVS verifica quais as mudanças existentes entre o arquivo no repositório e o novo arquivo. Essas mudanças são guardadas num histórico - onde podem ser visualizadas e/ou desfeitas -, o arquivo é liberado para check-out por qualquer outro desenvolvedor e o novo arquivo aparece em modo de

leitura para todos.

Uma sessão comum no CVS baseia-se em um desenvolvedor conectando-se ao sistema e puxando as novas versões dos arquivos (se existir alguma). Daí ele pode decidir em quais arquivos ele irá trabalhar, e faz o check-out do mesmo - desde que não estejam "trancados" por outro desenvolvedor, faz as mudanças e realiza, por fim, o check-in do arquivo, liberando-o para alteração por outros desenvolvedores. Algumas versões do CVS permitem que vários desenvolvedores façam o check-out simultâneo, e, ao fazer o check-in, as mudanças de ambos são colocadas no arquivo. Um problema ocorre caso os dois desenvolvedores mexem na mesma parte do arquivo: o último desenvolvedor a fazer o check-in deve então decidir manualmente como o arquivo deve ficar.

O CVS foi uma alternativa perfeita para o trabalho do grupo, que pôde trabalhar à distância sem se preocupar em atrapalhar os avanços dos outros integrantes, e ao mesmo tempo não se preocupando em manchar algum código, já que todas as versões dos arquivos são guardadas, há sempre a possibilidade de se voltar a uma versão anterior, ou de verificar onde os erros foram introduzidos.

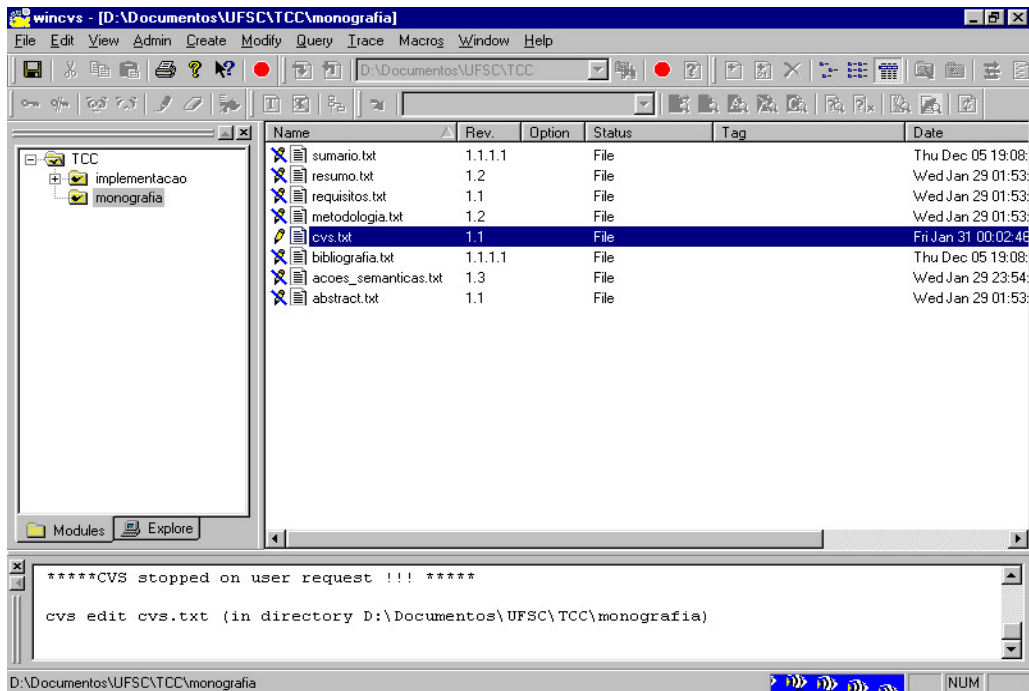


Figura 2.1: Software para trabalhar com CVS.

O site Sourceforge (<http://www.sourceforge.net>) disponibiliza o serviço de CVS, além de vários outros serviços como mailing lists e forums, para os desenvolvedores Open Source. O grupo utilizou esse serviço gratuito para desenvolver esse game.

O cliente CVS escolhido pela equipe para se conectar ao servidor CVS do Sourceforge foi o WinCVS. De código aberto, o programa é intuitivo e de fácil utilização.

### **3. Implementação do Sistema**

### 3.1 Requisitos Funcionais

O game proposto neste trabalho deverá atender aos seguintes requisitos funcionais:

- O game deverá seguir o estilo "Adventure" (ou Aventura). Seguindo este estilo, o jogador deverá controlar um personagem através do mouse e comandos via teclado;
- Como o game seguirá o estilo "Adventure" (aventura), não possuirá fases ou rodadas;
- O personagem poderá ir de um cenário a outro quando alcançar os limites do cenário em que se encontra;
- Os limites de cada cenário, assim como outras propriedades dos mesmos, poderão ser alterados através de um editor apropriado;
- O jogador poderá controlar a posição do personagem através do mouse. Ao clique do mouse, o personagem deverá mover-se de sua posição atual até a posição final, indicada pelo jogador através do mouse. O personagem deverá desviar-se dos obstáculos que estiverem em seu caminho;
- As ações a serem executadas pelo personagem serão indicadas pelo jogador através de comandos enviados via teclado;
- O personagem poderá carregar um número limitado de objetos, os quais poderão ser visualizados pelo jogador através de uma interface gráfica;
- Para vencer o game o jogador deverá alcançar uma série de objetivos, obedecendo uma ordem pré-estabelecida;
- A apresentação visual do game se dará através de imagens pseudo-3D;
- Deverão existir ferramentas externas ao game, com o intuito de auxiliar a criação do mesmo, tais como editor para os cenários e

softwares para testes/validações de determinados algoritmos;

- O objetivo do game será fazer o personagem pegar três objetos e colocá-los em um compartimento, obedecendo a uma determinada sequência;

## 3.2 Modelagem Conceitual

Nesta seção é abordada a modelagem conceitual das classes envolvidas no projeto.

De acordo com a modelagem elaborada para o desenvolvimento do game proposto neste trabalho, foram identificadas dez classes (TJogo, TTela, TPersonagem, TObjeto, TInterpretador, Tlexico, Ttoken, Trespota, Tinterface e TObjetivo). As classes assim como os relacionamentos entre as mesmas estão detalhadas na figura 3.1.

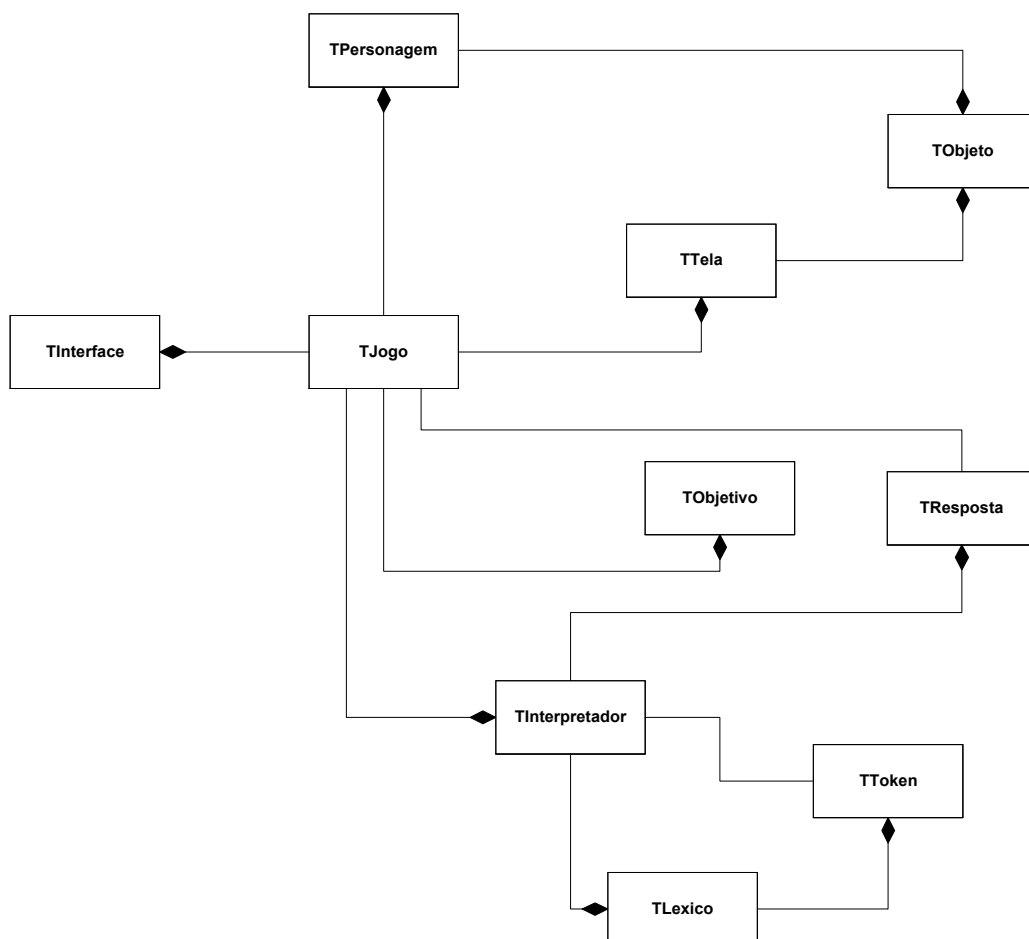


Figura 3.1: Diagrama de Relacionamento entre as classes do sistema.

Descrição das classes:

**TJogo:** A classe TJogo é a principal classe do sistema, podendo ser analisada como sendo uma classe "gerente". Isto quer dizer que através da classe TJogo é realizada a interação do jogador com o game propriamente dito.

Esta classe tem como propriedades instâncias das classes TPersonagem, TInterpretador e TTela. Além de um relacionamento com a classe TResposta, objeto este devolvido pela classe TInterpretador como resposta à uma chamada ao método "interpretar(string frase)" (método responsável pela interpretação de um comando enviado pelo jogador ao sistema). Maiores detalhes a respeito do funcionamento da interpretação de um comando e o relacionamento entre as classes TJogo e TInterpretador estão presentes no capítulo 3.2 (O Interpretador de Comandos).

**TPersonagem:** A classe TPersonagem tem como objetivo modelar o comportamento do personagem do game, controlado pelo jogador.

Esta classe tem como uma de suas propriedades uma lista de instâncias da classe TObjeto. Esta lista representa os objetos que estão sendo carregados pelo personagem. Por exemplo, se o jogador em um determinado instante envia um comando ao personagem, instruindo-o a pegar um objeto presente na tela, este objeto será então inserido na lista de objetos do personagem.

**TTela:** A classe TTela representa as propriedades presentes nas telas(ou cenários) do game, tais como: quais objetos estão em cada tela, quais eventos estão associados à cada região de cada tela, etc.



**TObjeto:** Esta classe representa os objetos presentes no game.

**TInterpretador:** Esta classe representa o interpretador de comandos, responsável por analisar os comandos enviados pelo jogador à classe TJogo.

Cada comando enviado pelo jogador é analisado pela classe Interpretador e uma instância da classe TResposta é enviada como resposta à classe TJogo. A classe TJogo executa então a ação apropriada.

**TLexico:** Classe responsável pela análise léxica dos comandos analisados pelo interpretador de comandos. Esta classe é propriedade da classe TInterpretador, é uma classe auxiliar, utilizada para fornecer ao interpretador de comandos os tokens presentes nos comandos sendo analisados.

**TResposta:** Uma instância da classe TResposta é devolvida pelo interpretador de comandos à classe TJogo como resposta ao método "interpretar(string frase)". Esta instância contém dados relacionados ao resultado obtido com a análise do comando em questão como, por exemplo, qual ação deverá ser executada pela classe TJogo, se ocorreu algum problema com a interpretação do comando, etc.

**TObjetivo:** A classe TJogo contém uma lista de instâncias da classe TObjetivo, sendo que cada um destes representa um determinado objetivo que deve ser realizado pelo jogador. Cada instância da classe TObjetivo poderá conter referências a outras instâncias desta mesma classe, representando-se desta maneira uma dependência entre os diversos objetivos do game. Desta maneira, pode-se representar a necessidade de uma ordem para a realização dos diversos objetivos.

**Tinterface:** Esta classe é responsável pelas ações de desenho na tela. Todas as outras classes que desejam desenhar na tela devem fazer

chamadas a métodos desta classe. Esta é a única classe que tem acesso as funções da biblioteca Allegro.

### 3.3 Arquitetura do sistema

Durante a fase de desenvolvimento do projeto, percebeu-se a possibilidade de criá-lo como um sistema genérico de games estilo adventure, e não apenas como sendo um game único e específico.

O uso da expressão sistema genérico refere-se à possibilidade de elaborar-se diferentes games utilizando-se sempre o mesmo software (arquivo executável). Este software contém toda a lógica do game, isto é, o interpretador de comandos, os algoritmos responsáveis pelos movimentos dos personagens, os algoritmos para a interface com o usuário etc. Todos estes algoritmos implementados de forma genérica, dependendo de instruções externas ao sistema (um arquivo de configuração e recursos de imagens, sons, etc), possibilitam diferentes comportamentos do sistema e então diferentes games.

A figura 3.2 ilustra o funcionamento geral do software atuando como um sistema genérico.

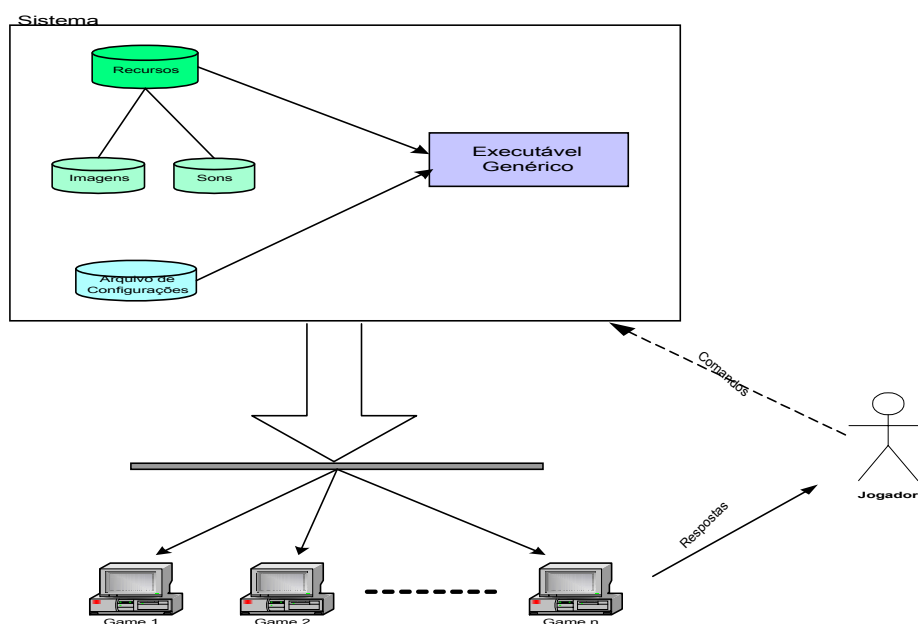


Figura 3.2: Sistema genérico.

Todos os algoritmos do software foram implementados de maneira genérica, portanto os dados específicos para cada game (personagens, sons, telas, telas virtuais, objetivos a serem cumpridos, etc) devem ser fornecidos através de um arquivo de configuração. Na seção de inicialização do sistema, este arquivo de configuração é lido assim como os recursos carregados para a memória. O arquivo de configuração pode ser gerado utilizando-se o editor de games descrito neste capítulo. A criação de diferentes games a partir de diferentes arquivos de configuração será melhor detalhada na seção Arquivo de Configuração.

Os diferentes recursos (tais como sons e imagens por exemplo) utilizados em um game podem ser gerados através de ferramentas externas ao sistema (tais como editores de imagem e som) e depois incorporados em arquivos especiais, a serem lidos pelo game em sua fase de inicialização.

### **3.3.1 Arquivo de Configuração**

Os diferentes recursos tais como sons e imagens são integrados ao sistema através do arquivo de configurações, este arquivo contém as informações sobre as telas e objetos do sistema assim como todas as suas propriedades.

Para a elaboração de um arquivo de configuração resolveu-se utilizar o padrão XML. Adotou-se este padrão por tornar mais prática a criação e validação do arquivo de configuração devido às ferramentas já existentes atualmente, as quais são largamente utilizadas para se trabalhar com documentos XML.

Um documento XML utilizado como arquivo de configuração de um game contém todas as informações necessárias tais como título do game,

quais telas estão presentes neste game, quais objetos cada tela contém, etc. A Figura 3.3 ilustra um exemplo de um documento XML que poderia ser utilizado para configurar um pequeno game.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/css" href="estilo_xml.css" ?>

<!DOCTYPE JOGO
[
  <!ELEMENT JOGO (TELA+, JOGADOR, OBJETIVO+, EVENTO*)>
  <!ATTLIST JOGO titulo CDATA "">

  <!ELEMENT TELA (OBJETO*, PERSONAGEM*, CENARIO*, TRANSICAO*)>
  <!ATTLIST TELA id NMTOKEN #REQUIRED>

  <!ELEMENT OBJETO (#PCDATA)>
  <!ATTLIST OBJETO id NMTOKEN #REQUIRED posX NMTOKEN #REQUIRED posY NMTOKEN #REQUIRED
tipo (objeto | container) "objeto">

  <!ELEMENT PERSONAGEM (#PCDATA)>
  <!ATTLIST PERSONAGEM id NMTOKEN #REQUIRED>

  <!ELEMENT CENARIO (#PCDATA)>
  <!ATTLIST CENARIO id NMTOKEN #REQUIRED posX NMTOKEN #REQUIRED posY NMTOKEN
#REQUIRED nivel (0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
16 ) "0">

  <!ELEMENT TRANSICAO (#PCDATA)>
  <!ATTLIST TRANSICAO cor NMTOKEN #REQUIRED destino NMTOKEN #REQUIRED>

  <!ELEMENT JOGADOR (#PCDATA | OBJETO)*>
  <!ATTLIST JOGADOR posX NMTOKEN #REQUIRED posY NMTOKEN #REQUIRED>

  <!ELEMENT OBJETIVO (#PCDATA | DEPEND)*>
  <!ATTLIST OBJETIVO id NMTOKEN #REQUIRED idAcao NMTOKEN #REQUIRED>

  <!ELEMENT DEPEND (#PCDATA)>
  <!ATTLIST DEPEND idObjetivo NMTOKEN #REQUIRED >

  <!ELEMENT EVENTO (#PCDATA)>
  <!ATTLIST EVENTO tipo (som | animacao) "som" idAcao NMTOKEN #REQUIRED idObjeto
NMTOKEN #REQUIRED idObjetoLugar NMTOKEN #REQUIRED>
]>
```

*Figura 3.3: Documento XML para arquivo de configuração (parte 1/2).*

```

<JOGO titulo="TCC Quest">
  <!-- Telas: Cada tela do jogo pode conter objetos, personagens e cenarios.-->
  <TELA id="0">
    <OBJETO id="0" posX="50" posY="30" tipo="objeto">Pedra</OBJETO>
    <OBJETO id="1" posX="100" posY="80" tipo="container">Caixa</OBJETO>

    <PERSONAGEM id="1">Victor</PERSONAGEM>
    <CENARIO id="0" posX="50" posY="30" >Arvore</CENARIO>

    <!--Transicoes definem para quais telas o jogador pode ir atraves da tela atual-->
    <TRANSICAO cor="255:255:0" destino="1">Transicao para tela: 1</TRANSICAO>
  </TELA>

  <TELA id="1">
    <OBJETO id="2" posX="50" posY="30" tipo="objeto">Pedra</OBJETO>
    <OBJETO id="3" posX="100" posY="80" tipo="container">Caixa</OBJETO>
    <TRANSICAO cor="255:255:1" destino="0">Transicao para tela: 0</TRANSICAO>
  </TELA>

  <!-- O Jogo pode ter apenas um Jogador. Cada Jogador pode iniciar o jogo
  carregando alguns objetos.-->
  <JOGADOR posX="50" posY="30">Jogador
    <OBJETO id="4" posX="20" posY="10" tipo="container">Carteira</OBJETO>
    <OBJETO id="5" posX="20" posY="10" tipo="objeto">Dinheiro</OBJETO>
  </JOGADOR>

  <!--O Jogo possui uma serie de objetivos que devem ser cumpridos pelo jogador: -->
  <!--Cada objetivo pode estar dependendo de outros objetivos para que possam ser
  cumpridos-->
  <OBJETIVO id="0" idAcao="3">Objetivo 0
  </OBJETIVO>
  <OBJETIVO id="1" idAcao="5">Objetivo 1
    <DEPEND idObjetivo="1">^... dependente do objetivo 0</DEPEND>
  </OBJETIVO>

  <!--O Jogo possui diversos eventos, disparados de acordo com determinadas acoes e
  objetos-->
  <EVENTO tipo="som" idAcao="3" idObjeto="0" idObjetoLugar="-1">
    Evento: som;
    Acao: 3;
    Objeto: 0;
    ObjetoLugar: nao;
  </EVENTO>
</JOGO>

```

*Figura 3.3: Documento XML para arquivo de configuração (parte 2/2).*

### 3.3.2 Game Loop

Tradicionalmente todos os games desenvolvidos possuem uma mesma estrutura básica em seu Loop principal conforme ilustrado na Figura 3.4. Este esquema, dividido em seções, é bastante flexível no sentido de poder-se agrupar algumas etapas em uma única ou dividir alguma etapa em outras menores, sendo que cada projeto de game deve analisar suas necessidades e tomar as decisões apropriadas no design do projeto. Uma decisão errada neste sentido pode acarretar em diversos problemas durante o estágio de desenvolvimento do projeto, podendo até tornar inviável a codificação do mesmo.

Neste projeto utilizamos uma estrutura de Loop bastante tradicional e enxuta, como sugerido em [10].

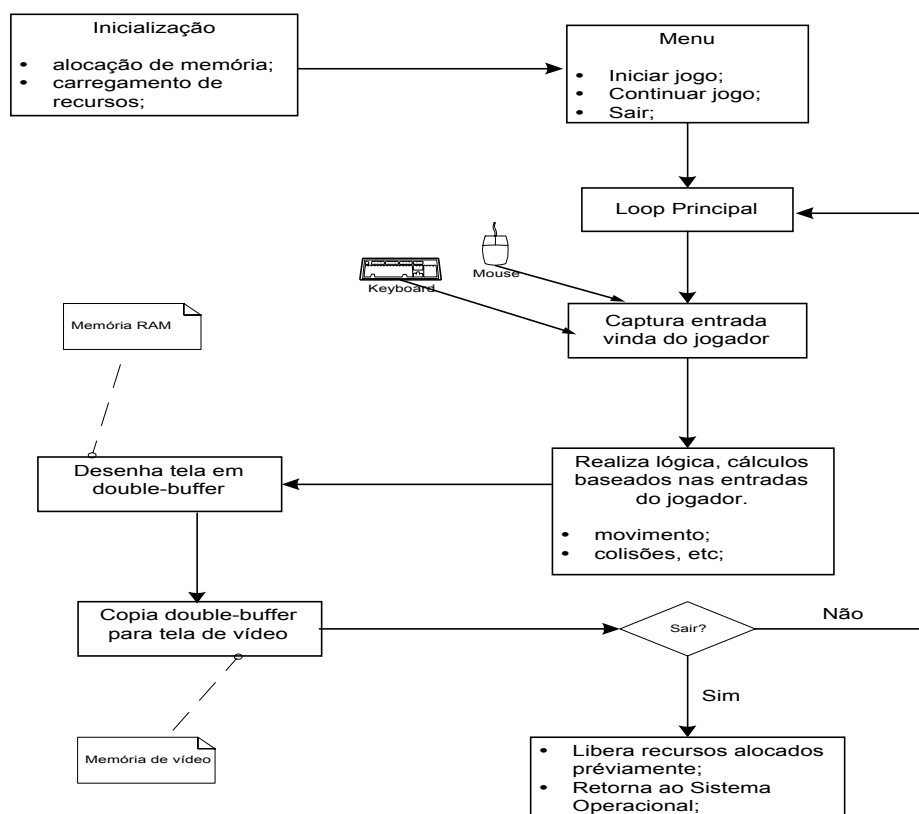


Figura 3.4: Game Loop tradicional.

O Game Loop pode ser representado computacionalmente através de uma máquina de estados finitos - FSM, onde cada estado representa um estágio do esquema da Figura 3.4.

As seções representadas na Figura 3.4 são explicadas com detalhes a seguir:

**Seção 1 - Inicialização:** Nesta seção, os recursos do game devem ser carregados do disco e a memória necessária para tal deve ser devidamente alocada. Além disto, todas as estruturas de dados necessárias devem ser inicializadas. Após estas inicializações, o game pode passar para o estado seguinte;

**Seção 2 - Menu Principal:** Neste estado, é apresentado um menu de opções ao jogador e o game entra em um loop infinito, aguardando uma resposta do jogador. As opções mais comuns são: Iniciar, Continuar, Créditos e Sair;

**Seção 3 - Início do Loop Principal:** Caso o jogador tenha escolhido a opção "Iniciar" quando na seção 2, o Loop Principal do game terá início. Este loop abrange as seções de 4 à 8, onde na seção 8 faz-se uma verificação para o caso do jogador ter sinalizado sua intenção de finalizar o game;

**Seção 4 - Entrada de dados:** Nesta etapa é feita uma leitura dos buffers dos dispositivos de entrada configurados para o game em questão (em nosso sistema teclado e mouse). Os dados capturados nesta etapa são armazenados em variáveis apropriadas, para serem utilizados no próximo estado.

**Seção 5 - Lógica do game:** Com base nos dados capturados no estado anterior, a lógica do game é computada. Esta lógica envolve toda a



computação que não produz saída em vídeo, como por exemplo cálculos para verificação de colisões, posicionamento de objetos, etc.

**Seção 6 - Desenho da Tela em Buffer:** Tendo sido calculadas todas as informações a respeito do estado do game nesta interação do loop, inicia-se a etapa de desenho do próximo quadro de animação do game. Para evitar problemas de "Flickering" – quando ocorrem algumas "piscadas" no vídeo durante o desenho do mesmo – o próximo quadro de animação é desenhado primeiro em um buffer em memória (que não aparece no vídeo) onde a velocidade de desenho não é tão crucial.

**Seção 7 - Cópia do Buffer para Tela:** Após o próximo quadro de animação ter sido completamente desenhado em um buffer na memória, pode-se copiar este buffer de uma só vez utilizando rotinas bastante rápidas, onde esta área de memória RAM é toda copiada para a memória de vídeo, produzindo então a imagem do próximo quadro de animação sem problemas de "Flickering".

**Seção 8 - Verifica sinal de saída:** Neste estado faz-se apenas uma leitura de uma variável para determinar se o jogador decidiu abandonar o sistema. Em caso positivo, o próximo estado será "Liberação de recursos alocados" senão o loop voltará para novamente para o estado "Entrada de dados".

**Seção 9 - Liberação de recursos alocados:** Este último estado é acessado apenas quando o jogador decide abandonar o sistema. Faz-se então uma liberação de todos os recursos alocados no primeiro estado e retorna o comando para o sistema operacional.

Através da descrição das seções envolvidas em um Game Loop, podemos concluir que um game é, basicamente, um loop infinito no qual se processa todos os objetos e elementos do game e então desenha-se o

próximo quadro de animação [9].

### 3.4 O Interpretador de Comandos

Tendo em vista que no game em questão as ações a serem executadas pelo personagem serão transmitidas pelo jogador através de comandos escritos (via teclado), existe a necessidade da implementação de um Interpretador de Comandos. Este capítulo dedica-se ao esclarecimento de seu funcionamento, assim como o método escolhido para sua implementação.

Se o jogador desejar instruir o personagem do game a executar uma determinada ação como, por exemplo, pegar um certo objeto(uma pedra) presente no cenário atual, o seguinte comando(sem as aspas) poderia ser enviado via teclado pelo jogador: "pegue a pedra". O game (mais especificamente, a classe TJogo) seria responsável por receber tal comando e realizar uma análise/validação sobre a mesma.

Para realizar esta validação sobre o comando enviado pelo jogador, a classe TJogo utiliza-se do Interpretador de Comandos (representado pela classe TInterpretador).

O Interpretador de Comandos é ativado através do método "analiseSintatica(string frase)". Este método recebe como parâmetro uma frase, que representa o comando em si e retorna uma instância da classe TResposta.

As relações existentes entre estas três classes podem ser observadas no diagrama da Figura 3.1.

Para a análise sintática dos comandos foi implementado um parser descendente recursivo (um parser preditivo - LL1). E, para a geração do algoritmo de análise sintática, foi utilizado um Gerador de Analisadores

Sintáticos, mais especificamente o GAS (Gerador de Analisadores Sintáticos). A gramática utilizada pelo analisador sintático é apresentada na Figura 3.5.

```
Vn = { <comando> <ver> <obj> <personagem> <para-persn> <lugar> <frase>
<falar> }

Vt = { olhe veja examine descreva pegue largue abra feche cheire coma use
chute fale de entregue coloque converse ria descanse durma grite a o ao
para na no com literal id }

S = <comando>

P = {
<comando> ::= olhe <ver> #1 | veja <ver> #1 | examine <obj> #2 | descreva
<obj> #2 | pegue <obj> #3 | largue <obj> #4 | abra <obj> #5 | feche <obj>
#6 | cheire <obj> #7 | coma <obj> #8 | use <obj> #9 | chute <obj> #10 | de
<obj> #12 <personagem> #13 | entregue <obj> #12 <personagem> #13 | coloque
<obj> #12 <lugar> #14 | ria #15 | descanse #16 | durma #17 | fale <falar>
<frase> #11 | grite <falar> <frase> #18 | converse <falar> #21 | ajuda #25;

<ver> ::= <obj> | î ;
<obj> ::= a #19 id | o #20 id ;
<personagem> ::= ao #20 id | a #19 id | para <para-persn> ;
<para-persn> ::= o #20 id | a #19 id | id ;
<lugar> ::= na #19 id | no #20 id ;
<falar> ::= com <para-persn> | î ;
<frase> ::= literal | î ;
}
```

*Figura 3.5: Gramática utilizada para a interpretação dos comandos.*

Já para a análise léxica dos comandos foi implementado um analisador léxico de acordo com a especificação representada pelo autômato finito determinístico apresentado na Figura 3.6.

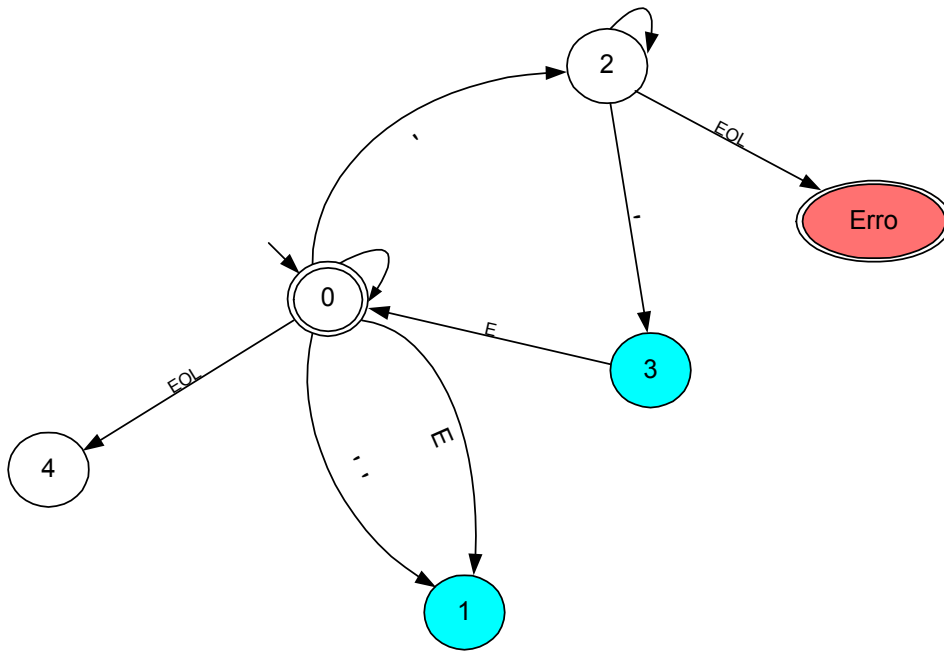


Figura 3.6: Autômato Finito utilizado na análise léxica.

A interpretação dos comandos no game é realizada através de um analisador sintático (TInterpretador), o qual utiliza-se de um analisador léxico (TLexico) para a obtenção dos tokens da linguagem. Ao ser invocado, o método "analiseSintatica(string frase)" interage com as classes TLexico e TResposta, como pode ser observado pela Figura 3.7.

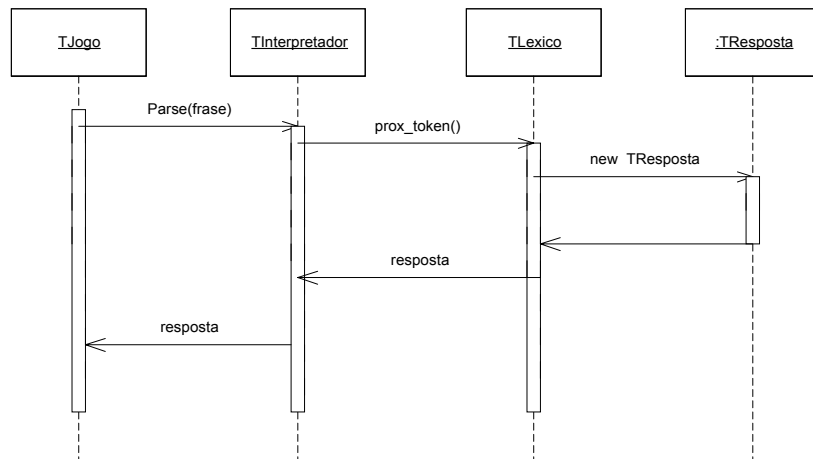


Figura 3.7.: Interação entre classes para a análise sintática.

Assim, a interpretação do comando acima se daria da seguinte maneira:

1. Chamada ao método `analiseSintatica("pegue a pedra");`
2. A classe `TInterpretador` faria chamadas à classe `TLexico`, a qual retornaria os tokens referentes à frase analisada;
3. Criação e retorno de uma instância da classe `TResposta`;

O objeto retornado pela chamada ao analisador sintático contém, no caso de uma sentença sintaticamente correta, o número da ação que deverá ser executada pela classe `TJogo`. Esta ação pode ser comparada a uma ação semântica em um compilador tradicional.

A Figura 3.8 contém as ações possíveis a serem executadas pela classe `TJogo`.

```
#1: Se o personagem está carregando o objeto em questão então
  Se o sexo do objeto em questão = SEXO_OBJ_ATUAL então
    Se o objeto em questão atende algum objetivo através desta ação então
      Marca o objetivo correspondente como cumprido
      Executa o evento correspondente
    Senão{ não faz nada }
    Mostra imagem do objeto
  Senão mostra mensagem avisando que o personagem não está com o objeto em questão
Senão

Se o objeto em questão está presente na tela atual então
  Se o sexo objeto em questão = SEXO_OBJ_ATUAL então
    Mostra mensagem descritiva resumida do objeto
  Senão mostra mensagem de que não encontrou tal objeto
Senão mostra mensagem de que não encontrou tal objeto

#2: Se o personagem está carregando o objeto em questão então
  Se o sexo do objeto em questão = SEXO_OBJ_ATUAL então
    Se o objeto atende algum objetivo através desta ação então
      Marca o objetivo correspondente como cumprido
      Executa o evento correspondente
    Senão{ }
    Mostra imagem do objeto
  Senão mostra mensagem avisando que o personagem não está com o objeto em questão
Senão
```

*Figura 3.8: Ações a serem executadas pela classe `TJogo` (parte 1/5).*

```

Se o objeto em questão está presente na tela atual então
  Se o sexo objeto em questão = SEXO_OBJ_ATUAL então
    Mostra mensagem descritiva completa do objeto, junto com sua imagem
  Senão mostra mensagem de que não encontrou tal objeto
Senão mostra mensagem de que não encontrou tal objeto

#3: Se o objeto em questão está presente na tela atual então
  Se o sexo do objeto em questão = SEXO_OBJ_ATUAL então
    Se o objeto em questão pode ser pego então
      Se o personagem estiver próximo o suficiente então
        Se o personagem tiver espaço suficiente em seu inventário então
          Se o objeto em questão atende algum objetivo através desta ação então
            Marca o objetivo correspondente como cumprido
            Executa o evento correspondente
            Senão{ não faz nada }
          Senão mostra mensagem de que não pode carregar o objeto
        Senão mostra mensagem de que não está próximo o suficiente para pegar o obj
      Senão mostra mensagem de que o objeto não pode ser pego
    Senão mostra mensagem de que não encontrou o objeto em questão
  Senão mostra mensagem de que não encontrou o objeto em questão

#4: Se o personagem está carregando o objeto em questão então
  Se o sexo do objeto em questão = SEXO_OBJ_ATUAL então
    Se o objeto em questão atende algum objetivo através desta ação então
      Marca o objetivo correspondente como cumprido
      Executa o evento correspondente
      Senão{ não faz nada }
    Se o objeto em questão pode ser largado então
      Coloca o objeto em questão na lista de objetos pertencentes a tela atual
      Remove o objeto da lista de objetos presentes no inventário do personagem
    Senão mostra mensagem de que o objeto em questão não pode ser largado
  Senão mostra mensagem de que o personagem não está carregando tal objeto
  Senão mostra mensagem de que o personagem não está carregando tal objeto

#5: Se o objeto em questão está na tela atual então
  Se o sexo do objeto em questão = SEXO_OBJ_ATUAL então
    Se o objeto em questão pode ser aberto então
      Seta objeto como aberto
      Dispara evento associado a esta ação
    Senão mostra mensagem de que o objeto em questão não pode ser aberto
  Senão mostra mensagem de que o personagem não está carregando tal objeto
Senão mostra mensagem de que o personagem não está carregando tal objeto

#6: Se o objeto em questão está na tela atual então
  Se o sexo do objeto em questão = SEXO_OBJ_ATUAL então
    Se o objeto em questão pode ser fechado então
      Seta objeto como fechado

```

*Figura 3.8: Ações a serem executadas pela classe TJogo (parte 2/5).*

```

Dispara evento associado a esta ação
    Senão mostra mensagem de que o objeto em questão não pode ser fechado
    Senão mostra mensagem de que o personagem não está carregando tal objeto
    Senão mostra mensagem de que o personagem não está carregando tal objeto

#7: Se o personagem está carregando o objeto em questão então
    Se o sexo do objeto em questão = SEXO_OBJ_ATUAL então
        Se o objeto em questão atende algum objetivo através desta ação então
            Marca o objetivo correspondente como cumprido
            Executa o evento correspondente
            Senão{ não faz nada }
        Mostra mensagem com descrição do cheiro do objeto
        Senão mostra mensagem de que o personagem não está carregando tal objeto
        Senão mostra mensagem de que o personagem não está carregando tal objeto

#8: Se o personagem está carregando o objeto em questão então
    Se o sexo do objeto em questão = SEXO_OBJ_ATUAL então
        Se o objeto em questão atende algum objetivo através desta ação então
            Marca o objetivo correspondente como cumprido
            Executa o evento correspondente
            Senão{ não faz nada }
        Senão mostra mensagem de que o personagem não está carregando tal objeto
        Senão mostra mensagem de que o personagem não está carregando tal objeto

#9: Se o personagem está carregando o objeto em questão então
    Se o sexo do objeto em questão = SEXO_OBJ_ATUAL então
        Se o objeto em questão atende algum objetivo através desta ação então
            Marca o objetivo correspondente como cumprido
            Executa o evento correspondente
            Senão{ não faz nada }
        Se o objeto em questão pode ser usado então
            Dispara evento associado a esta ação
            Senão mostra mensagem de que o objeto em questão não pode ser usado
            Senão mostra mensagem de que o personagem não está carregando tal objeto
            Senão mostra mensagem de que o personagem não está carregando tal objeto

#10: Se o personagem está carregando o objeto em questão então
    Se o sexo do objeto em questão = SEXO_OBJ_ATUAL então
        Se o objeto em questão atende algum objetivo através desta ação então
            Marca o objetivo correspondente como cumprido
            Executa o evento correspondente
            Senão{ não faz nada }
        Senão mostra mensagem de que o personagem não está carregando tal objeto
        Senão mostra mensagem de que o personagem não está carregando tal objeto

#11: Se o tamanho da frase for menor que o tamanho máximo permitido então
    Mostra um balão de diálogo(falando) com a frase digitada pelo jogador

```

*Figura 3.8: Ações a serem executadas pela classe Tjogo (parte 3/5).*



```

Senão mostra mensagem de que a frase tem que ser menor

#12: Se o personagem está carregando o objeto em questão então
    Se o sexo do objeto em questão = SEXO_OBJ_ATUAL então
        Se o objeto em questão não pode ser colocado então
            mostra mensagem de que o objeto não pode ser colocado
            Senão{ não faz nada}
        Senão mostra mensagem de que o personagem não está carregando tal objeto
    Senão mostra mensagem de que o personagem não está carregando tal objeto

#13: Se o objeto_personagem em questão está presente na tela atual então
    Se o sexo do objeto_personagem em questão = SEXO_OBJ_ATUAL então
        Se o personagem está perto o suficiente do obj_personagem em questão então
            Se o objeto_personagem for do tipo container então
                Se o objeto em questão atende algum objetivo através desta ação então
                    Marca o objetivo correspondente como cumprido
                    Executa o evento correspondente
                    Senão{ não faz nada }
                objeto_personagem recebe objeto
                dispara evento associado a esta ação
            Senão mostra mensagem de que o objeto_personagem não é recipiente
        Senão mostra mensagem de que o personagem não está perto o suficiente
    Senão mostra mensagem de que o personagem não está carregando tal objeto
    Senão mostra mensagem de que o personagem não está carregando tal objeto

#14: Se o objeto_lugar em questão está presente na tela atual então
    Se o sexo do objeto_lugar em questão = SEXO_OBJ_ATUAL então
        Se o personagem está perto o suficiente do objeto_lugar em questão então
            Se o objeto_lugar for do tipo container então
                objeto_lugar recebe objeto
                dispara evento associado a esta ação
            Senão mostra mensagem de que o objeto_lugar não é recipiente
        Senão mostra mensagem de que o personagem não está perto o suficiente
    Senão mostra mensagem de que o personagem não está carregando tal objeto
    Senão mostra mensagem de que o personagem não está carregando tal objeto

#15: Dispara ação associada a este evento

#16: Dispara ação associada a este evento

#17: Dispara ação associada a este evento

#18: Se o tamanho da frase for menor que o tamanho máximo permitido então
    Mostra um balão de diálogo(gritando) com a frase digitada pelo jogador
    Senão mostra mensagem de que a frase tem que ser menor

#19: A ser executada diretamente pelo interpretador. Seta SEXO_OBJ_ATUAL= FEMININO;

```

**Figura 3.8:** Ações a serem executadas pela classe TJogo (parte 4/5).

```
#20: A ser executada diretamente pelo interpretador. Seto SEXO_OBJ_ATUAL= MASCULINO;

#21: Se o objeto_personagem em questão está presente na tela atual então
    Se o sexo do objeto_personagem em questão = SEXO_OBJ_ATUAL então
        Se o personagem está perto o suficiente do obj_personagem em questão então
            Se o objeto em questão atende algum objetivo através desta ação então
                Marca o objetivo correspondente como cumprido
                Executa o evento correspondente
            Senão{ não faz nada }
            objeto_personagem recebe objeto
            dispara evento associado a esta ação
        Senão mostra mensagem de que o personagem não está perto o suficiente para
executar esta ação
            Senão mostra mensagem de que o personagem não está carregando tal objeto
            Senão mostra mensagem de que o personagem não está carregando tal objeto
```

*Figura 3.8: Ações a serem executadas pela classe TJogo (parte 5/5).*

## 3.5 Algoritmo para busca do caminho

Entre todos os tipos de decisões envolvidas na Inteligência Artificial aplicada em games de computador, talvez a mais comum delas seja a de achar o melhor caminho (algoritmos de PathFinding). Estes algoritmos são responsáveis por encontrar uma boa rota para mover uma determinada entidade (o personagem, no caso deste projeto) de um ponto a outro de um determinado cenário.

Apesar de parecer uma tarefa trivial, ela não o é. Algoritmos para busca do caminho freqüentemente são bastante complexos. Atualmente existem diversos algoritmos para busca do caminho disponíveis, alguns muito eficientes e outros nem tanto. Neste capítulo serão apresentados os principais algoritmos para busca do caminho, e detalhado o algoritmo escolhido para implementar as funções de busca do caminho em nosso game.

Desenvolver um algoritmo que faça com que um personagem de um game mova-se de um ponto até outro não parece ser um desafio muito grande. E na verdade não é. Pode-se escrever um algoritmo simples, como mostrado na Figura 3.9, o qual ignora a presença de obstáculos até que os encontre (colida com um deles):

```
enquanto nao chegou no destino
  escolha a direcao do destino
  se a direcao escolhida estiver livre
    mova-se para lah
  senao
    escolha outra direcao seguindo uma estrategia de escolha qualquer
```

*Figura 3.9: Algoritmo simples para busca do caminho*

Neste caso um simples algoritmo como o demonstrado na Figura 3.9 não resolve o problema com muita eficiência, pois o personagem ao ser interrompido pelo obstáculo, poderia perder um tempo considerável até que uma direção correta fosse escolhida. O que se deseja é um algoritmo onde o personagem, ao encontrar um obstáculo em seu caminho, altere a sua rota, desviando de tal obstáculo a um custo (tempo) aceitável.

### 3.5.1 Algoritmos Existentes

Alguns algoritmos que consideram a existência e o desvio de possíveis obstáculos são descritos a seguir [4].

**Movimento em uma direção randômica:** Se os obstáculos forem pequenos e convexos, o personagem poderia desviar-se do obstáculo movendo-se em uma direção qualquer (escolhida aleatoriamente), diferente daquela a qual está se tentando ir. O problema surge quando se depara com um obstáculo grande ou côncavo. Nestes casos o algoritmo se torna bastante ineficiente.

Uma saída seria projetar os cenários do game de modo que contenha apenas obstáculos pequenos e/ou convexos.

**Seguir em torno do obstáculo:** Existem outras técnicas que podemos usar caso o objeto seja grande demais. Podemos fazer, por exemplo, o equivalente a percorrer o obstáculo tateando-se os lados do mesmo até que o obstáculo tenha sido contornado.

O problema desta técnica é decidir quando se deve parar de contornar o obstáculo. Uma heurística simplista poderia ser parar de contornar o objeto quando se alcançar a direção em que estava seguindo quando do início da busca. Esta heurística funcionaria para a maioria dos casos, mas não todos eles.

A Figura 3.10 a seguir ilustra um caso onde este algoritmo não seria nem um pouco eficiente.

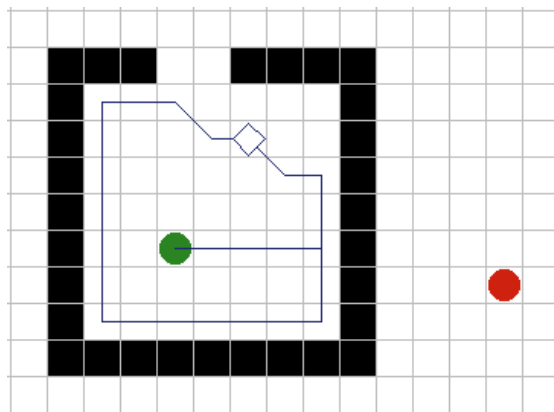


Figura 3.10: Algoritmo não consegue achar a saída.

Além das técnicas de desvio do obstáculo quando de uma colisão com o mesmo, existe outra classe de algoritmos, os quais consideram o cálculo do caminho inteiro a ser seguido antes mesmo de se iniciar o movimento. Muitos destes algoritmos utilizam-se de técnicas pertencentes as áreas de Teoria dos Grafos e Inteligência Artificial.

Um dos vários algoritmos que consideram o cálculo do caminho inteiro antes de qualquer movimento é descrito a seguir.

**Busca em largura:** Este método considera como nó inicial o ponto de partida, depois examina todos os nós vizinhos imediatos, depois todos os nós em dois níveis adiante, depois três níveis, e assim por diante, até que um nó alvo seja encontrado. Tipicamente, os nós vizinhos de cada nó a ser examinado são colocados em uma lista de nós "abertos" (geralmente

uma lista FIFO). A Figura 4.11 exemplifica o algoritmo descrito:

```
lista: open;
n, n1, s: no;

s.pai:= Null;
coloca s na lista open;
enqto open <> vazio faça
inicio
  retira n da lista open;
  constroi caminho;
  result:= sucesso;
fim;
para cada sucessor n1 de n
```

*Figura 3.11: Algoritmo de Busca em Largura.*

O algoritmo descrito na Figura 3.11 funcionará para todos os casos, apesar de não ser muito eficiente pois, ao invés de ir diretamente para o ponto destino, ele testa todas as direções primeiro.

Existem vários algoritmos pertencentes a esta segunda classe de algoritmos de busca do caminho, entre eles podemos citar o algoritmo de Dijkstra (semelhante ao algoritmo da Figura 3.11 que, ao invés de uma lista FIFO, utiliza uma lista com prioridades), o algoritmo de busca em profundidade, e o citado na literatura especializada como sendo o melhor algoritmo de todos o A\* (leia-se "A estrela").

### **3.5.2 Algoritmo Implementado**

Para implementar o algoritmo de busca do caminho no game proposto neste trabalho, a equipe decidiu utilizar um algoritmo

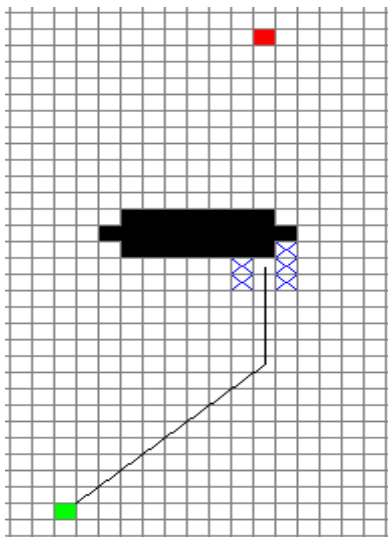
pertencente à primeira classe descrita neste trabalho.

O algoritmo funciona exatamente da maneira descrita na Figura 3.9 (algoritmo simples), sendo que a estratégia adotada para a escolha de uma nova direção a ser tomada quando de uma colisão com um obstáculo é descrita na Figura 3.12.

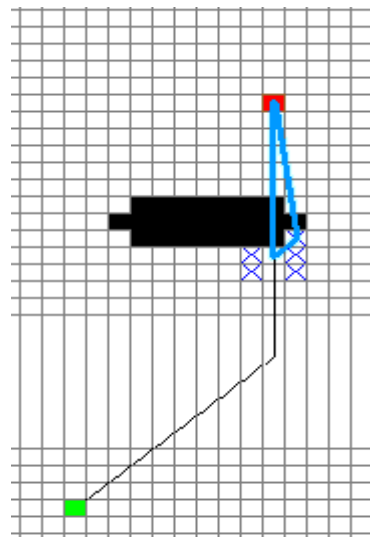
```
para cada ponto ao redor do ponto atual faca
  se o ponto ainda nao foi visitado ou testado entao
    traca um triangulo retangulo utilizando como vertices: o ponto atual, o
    ponto sendo testado e o ponto destino;
    guarda o valor da hipotenusa;
  fim se;
fim faca;
```

*Figura 3.12: Algoritmo implementado.*

A figura 3.13 exemplifica o descrito:



Foi encontrado um obstáculo no meio da trajetória.



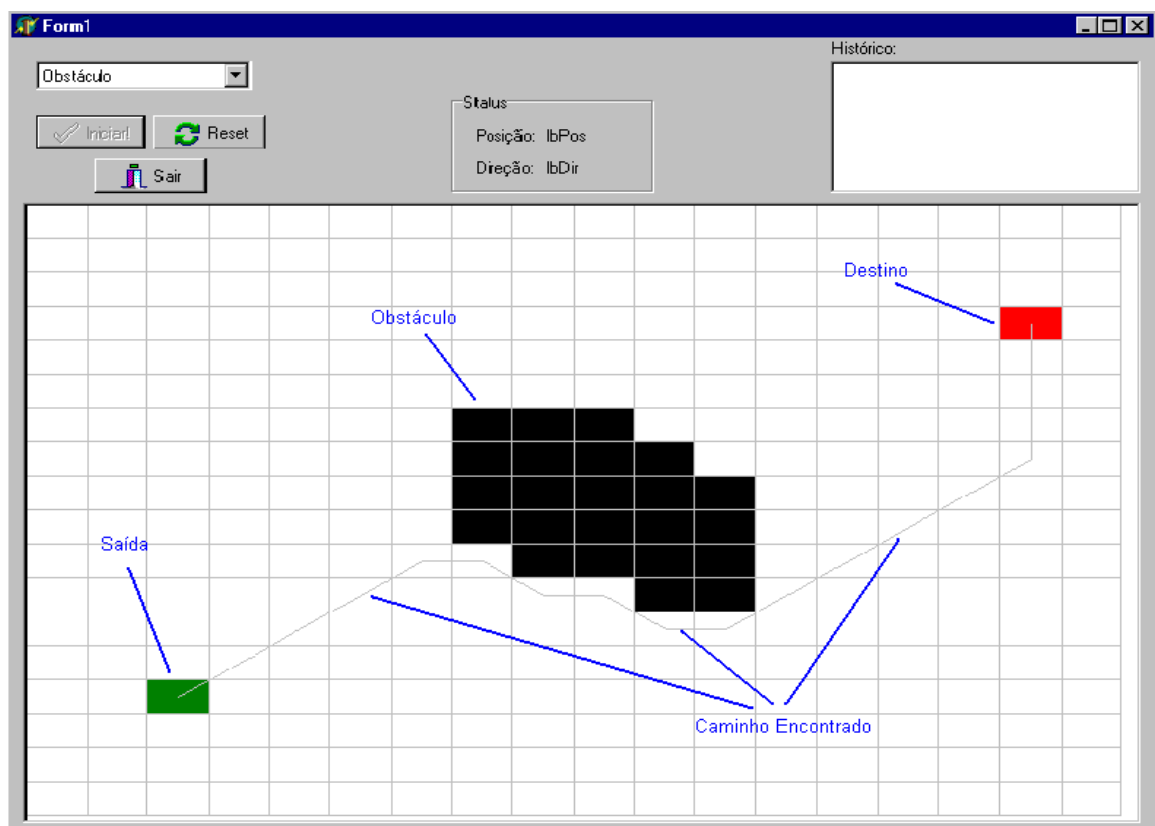
Para cada ponto vizinho ao ponto atual a hipotenusa de um triangulo retângulo é calculada.

*Figura 3.13: Estratégia adotada para a escolha de uma nova direção.*

Este algoritmo não possui a mesma eficiência dos algoritmos da segunda classe mas, por outro lado, não apresenta tantos problemas como os algoritmos descritos na primeira classe. Uma vez que o game proposto neste trabalho não necessita de um algoritmo tão eficiente para a busca do caminho, optou-se pela implementação deste algoritmo, onde conseguimos obter um equilíbrio entre eficiência e complexidade/tempo de desenvolvimento.

Para validar e testar o algoritmo escolhido desenvolveu-se um pequeno aplicativo utilizando uma ferramenta RAD (Rapid Application Development), no caso, Delphi. Utilizando este aplicativo, pode-se testar o comportamento do algoritmo diante de diferentes situações, envolvendo obstáculos de vários tamanhos e formas.

A figura 3.14 ilustra o aplicativo em andamento.



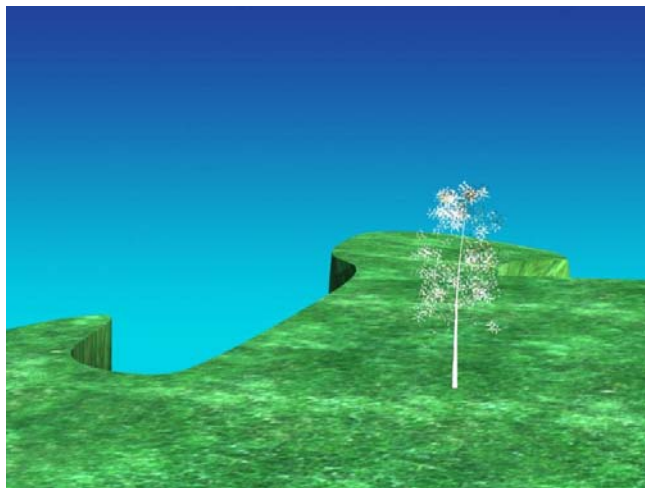


*Figura 3.14: Software para validar algoritmo de busca do caminho.*

### **3.6 Telas Virtuais**

O game em questão se passa em diferentes cenários, cada um representado por uma tela.

Em cada um desses cenários o jogador poderá, através do Personagem, interagir com os objetos que nele se encontram. Ou seja, o Personagem deve estar ciente de existirem obstáculos ao longo do caminho.

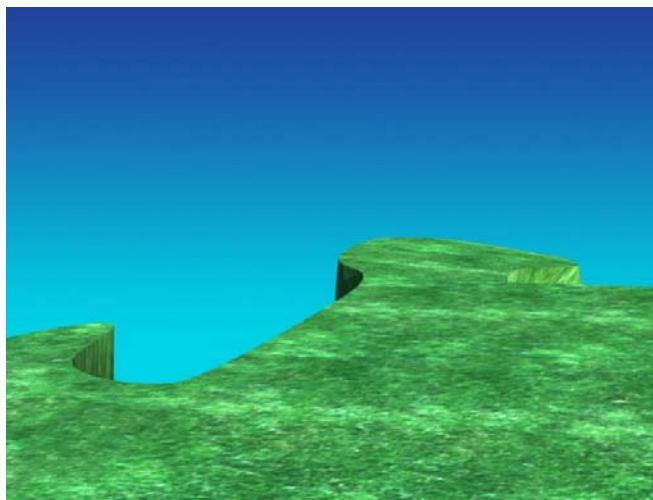


*Figura 3.15: Exemplo de cenário do game.*

Além disso existem ações especiais que devem ser tomadas em determinados locais do cenário como, por exemplo, a transição para outro cenário, a ativação de alguma animação ou a execução de um efeito sonoro – estas duas últimas referem-se ao disparo de um evento.

O cenário em si, que aparece para o jogador, nada mais é que uma imagem. Esta é colocada ao fundo, no que chamamos de Tela Atual. Os

objetos e o Personagem são desenhados no topo da Tela Atual.



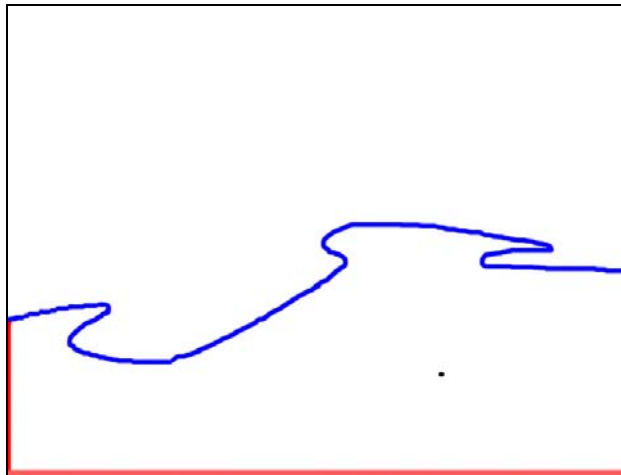
*Figura 3.16: Exemplo de tela "atual".*

A Tela Atual não tem nenhuma informação especial, sendo apenas a imagem do local, sem os objetos que nele se encontram. Por essa razão apareceu a necessidade de uma outra imagem com as informações sobre o cenário.

Essa imagem com as informações nunca aparece para o jogador, sendo apenas uma imagem residente na memória com as mesmas dimensões da Tela Atual e que é utilizada como referência para tomar decisões sobre ações especiais e obstáculos em tela.

Por nunca aparecer ao jogador, a equipe resolveu chamar essa imagem de Tela virtual.

A Tela virtual é uma imagem totalmente branca - cor que nada significa ao game - onde são pintados, com cores especiais, os locais onde existem obstáculos - nesse caso, com a cor preta, no caso de transições de tela, com tons de vermelho e obstáculos passíveis de serem desviados (com a utilização do algoritmo de busca do caminho) com a cor verde.



*Figura 3.17: Exemplo de Tela "interna".*

O game então, ao movimentar o Personagem pela tela, verifica se o mesmo colidiu com alguma área não-branca da Tela virtual. Havendo a colisão, verifica-se qual é a cor da área de colisão, e toma-se a ação necessária.

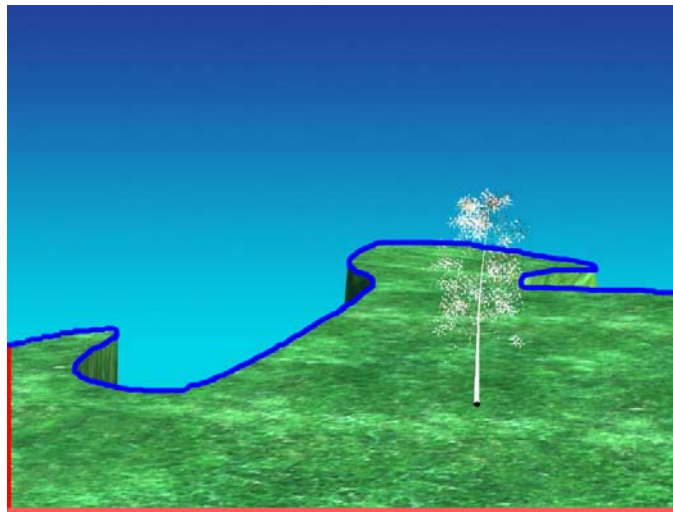
No caso de uma cor preta, o Personagem colidiu com um obstáculo não-contornável. Ou seja, não há como desviar do mesmo. Nesse caso, o personagem simplesmente conclui sua movimentação, ficando parado nesse local.

Se, ao invés da cor preta, for encontrada uma cor verde, sabe-se que o personagem colidiu com um obstáculo contornável. Nesse caso o algoritmo para busca do caminho é acionado, procurando um caminho ao redor do obstáculo para que o Personagem possa atingir seu ponto de destino. Já nos casos de ações especiais as decisões dependem de outros fatores.

Se o Personagem colidir com uma área de tom vermelho, que significa uma transição de tela, é necessário que se procure, na lista de transições do objeto da tela, qual a tela relacionada com aquele

determinado tom. E após isso, se faz a transição para essa tela (as transições de tela serão vistas com maiores detalhes na seção “Esquema de transições de tela”).

Combinando as informações da Tela Atual com a Tela virtual - que, novamente, nunca aparece ao jogador - temos todas as informações necessárias sobre o cenário atual, para que o Personagem possa interagir com o mesmo.



*Figura 3.18: Exemplo de Tela virtual sobre a atual.*

## 3.7 Níveis de Profundidade

Para deixar o game visualmente mais interessante, a apresentação do personagem não pode ser sempre a mesma.

A idéia geral é que, quando mais próximo da parte inferior da tela, o Personagem deve parecer maior, e quando mais perto da parte superior, menor, criando-se assim uma ilusão de distância.

Além disso, dependendo da posição do Personagem, o mesmo poderá estar na frente ou atrás de objetos que estão espalhados pela tela. Para resolver esse problema, adotou-se a idéia de Níveis de Tela.

Cada tela é dividida em 16 níveis, começando pelo nível 0 no canto superior da tela e descendo até o décimo sexto, cada qual com um valor de profundidade e um tamanho na tela em linhas (30 linhas).

A primeira coisa que deve-se pensar é na relação do Personagem com objetos. Quando os objetos e o Personagem são desenhados na tela, a ordem em que isso acontece é ditada pelo nível.

Começa-se pelo primeiro nível, descendo até o último. Dessa forma os objetos que se encontram mais longe são desenhados primeiro, e os que estão mais perto são desenhados por cima dos que estão mais longe. Esse algoritmo é conhecido como o algoritmo do pintor.

O próximo item a ser considerado é o tamanho do Personagem. Se o personagem estiver no primeiro nível, ele deve ser menor do que se estivesse no décimo nível... e muito menor do que se estivesse no último nível.

A idéia é de que tem-se um desenho do Personagem que é escalado de acordo com o nível em que o mesmo se encontra.

O valor da escala em que o Personagem é transformado é dado pelo valor de profundidade do nível atual. Cada tela pode ter valores de profundidade diferentes para cada nível. E os níveis podem até ter o mesmo valor de profundidade -- permitindo que coloquemos objetos na frente ou atrás do jogador sem preocupação com o tamanho do personagem se alterando.

Por fim, cada nível tem um valor de quantas linhas ocupa. A tela do game tem por definição 480 linhas, que podem ser divididas de várias formas entre os níveis. Sempre se trabalhando do canto inferior até o canto superior da tela.

O nível 0 sempre começa na linha 0 (linha mais superior da tela). Se o valor de linhas do nível 1 é 100, o mesmo irá da linha 0 até a 99. E o nível 1 começará a partir da linha 100.

Isto oferece uma grande flexibilidade ao game, pois permite que tenhamos cenários com o horizonte mais baixo ou mais alto.

### 3.8 Esquema de Transições de Tela

As telas virtuais também mantêm informações a respeito das transições entre telas. Estas transições (marcadas em tons de vermelho nas telas virtuais) tornam possível a interação do jogador com o ambiente em que o personagem se encontra.

Cada objeto instância da classe TTela possui uma lista de estruturas de dados chamadas de "transicoesTela". Esta estrutura contém as informações necessárias para se efetuar uma transição de tela. Como em uma mesma tela podem existir várias transições, as mesmas precisam se diferenciar na tela virtual de alguma maneira. Por esta razão optou-se pela marcação das transições na tela virtual em tons de vermelho, isto é, se em uma determinada tela necessitamos de três transições (cada uma para uma tela destino diferente) utilizamos três tons de vermelho distintos, assim como a cor "R:250, G:0, B:0" para a primeira transição, "R:240, G:0, B:0" para a segunda transição e R:230, G:0, B:0 para a terceira transição.

Desta maneira, o algoritmo responsável pela leitura dos pixels nas telas virtuais identificará as três situações como sendo a de uma transição de tela (vermelho, não importando o tom) e fará a chamada ao método apropriado para uma transição de tela.

O método responsável por efetuar a transição de telas irá buscar na lista de estruturas de dados "transicoesTela" o elemento que possuir uma transição associada ao tom de vermelho identificado pela tela virtual. Encontrado tal elemento, este contém então informação a respeito da tela de destino da transição a qual poderá então ser efetuada.

### 3.9 Esquema de Objetivos

Para que o jogador possa chegar ao final do game, é necessário que uma série de objetivos sejam cumpridos. Estes objetivos podem ser desde pegar ou largar um determinado objeto, ou qualquer outra ação que o personagem possa realizar, estando esta ação relacionada ou não com algum objeto.

Para tanto, a classe TJogo mantém uma lista de objetos instâncias da classe TObjetivo. Cada um destes objetos possui informações a respeito de quais outros objetivos este depende para que possa ser cumprido, quantos pontos o jogador irá marcar caso cumpra este objetivo, etc.

Como cada objetivo contém informação a respeito de quais outros objetivos este depende, é possível estabelecer uma cadeia de dependências, ou uma lista com a ordem exata dos objetivos a serem cumpridos (Figura 3.19). Portanto, o personagem não poderá cumprir objetivos fora da ordem pré estabelecida, o que poderia comprometer o andamento do game.

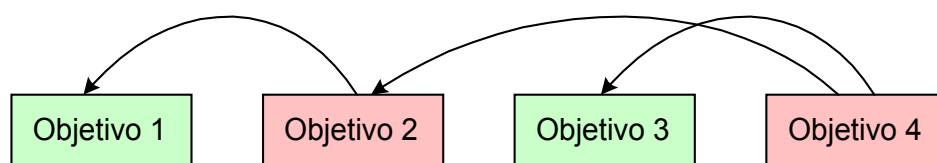


Figura 3.19: Dependências entre objetivos.

Na Figura 3.19 podemos observar que os objetivos 1 e 3 não dependem de nenhum outro objetivo para que os mesmos possam ser cumpridos, já o objetivo 2 depende do cumprimento do objetivo 1, assim como o objetivo 4 depende do cumprimento dos objetivos 2 e 3.



Pode-se imaginar o seguinte cenário: um game onde o objetivo seria o personagem pegar dois objetos do chão (sendo um objeto preto e outro azul) e colocá-los dentro de um recipiente obedecendo uma certa ordem tal como primeiro o objeto azul e em seguida o preto, por exemplo. Tendo sido este objetivo definido, o personagem não poderia ser capaz de colocar os objetos no recipiente na ordem inversa. Esta ordem é garantida através da lista de objetivos, e das dependências entre eles.

Assim, quando o personagem for executar alguma ação que resultaria no cumprimento de um objetivo, primeiramente é feita uma verificação se o mesmo pode ou não ser cumprido devido a presença de alguma dependência. Se o objetivo não depender de nenhum outro ele será marcado como "Objetivo Cumprido", senão uma mensagem será exibida ao jogador avisando que tal objetivo não pode ser cumprido ainda.

Além do esquema apresentado anteriormente, existe a necessidade de proibir o cumprimento de determinados objetivos, pois estes poderiam inviabilizar o cumprimento dos demais. Citando o exemplo anterior, se neste game o personagem fosse capaz de "destruir" os objetos que ele estivesse carregando consigo (através de um comando como: "destrua <obj>"), não poderia ser permitido que o personagem destruísse o objeto azul ou o preto. Pois, se o fizesse, não haveria mais como completar todos os objetivos previstos para o game.

Por este motivo, a classe TJogo mantém também uma lista de objetivos ditos "proibidos". Objetivos estes que, caso o personagem tente realizá-los, será impedido pelo sistema, que também alertará o jogador.

A realização de um objetivo está diretamente relacionada com as possíveis ações a serem realizadas pelo personagem. Para tanto, cada objeto da classe TObjetivo contém também informações a respeito de qual ação pode realizá-lo.

### 3.10 Esquema de Eventos

Assim como nos esquemas de transições de telas e de objetivos, a classe TJogo mantém também uma lista com os eventos possíveis de serem disparados em determinadas situações.

Os disparos de eventos estão diretamente associados com as ações semânticas do interpretador de comandos. Cada elemento da lista de eventos mantida pela classe TJogo contém informações sobre qual o tipo de evento será disparado (a execução de um efeito sonoro ou de uma animação, por exemplo), qual ação semântica deverá estar envolvida, assim como qual objeto(s). Logo, cada ação semântica tem a possibilidade de disparar um ou mais eventos quando o personagem interage com o sistema através da execução de determinadas ações semânticas.

Como exemplo, podemos imaginar a situação onde um dos elementos da lista de eventos tem associado a si a ação número #1 (disparada após o comando "olhe <obj>"), o objeto de nome "pedra" e o seu tipo de evento é execução de um efeito sonoro. Assim, quando o jogador executar o comando "olhe a pedra" a classe TJogo, verificará se em sua lista de eventos existe algum elemento que pode ser disparado e, se houver, executará o tipo de evento apropriado (neste caso a execução de um efeito sonoro).

Após a execução de cada ação semântica (as ações semânticas são executadas diretamente pela classe TJogo) a classe TJogo faz a verificação para o caso de um evento poder ser disparado, assim como acontece para o caso do cumprimento de objetivos.

## **4. Conclusão**

## 4. Conclusão

O mercado de jogos de computador é um mercado que movimentava bilhões de dólares por ano. E, apesar de existirem algumas universidades ao redor do mundo dedicadas ao seu ensino, a criação desses jogos é uma atividade que não segue muitos padrões e que requer uma gama dos mais variados tipos de conhecimentos.

É reconfortante saber que, ao final do curso de graduação em Ciências da Computação da Universidade Federal de Santa Catarina, como alunos pudemos acumular os conhecimentos necessários para a realização de qualquer tipo de atividade relacionada à informática, incluindo a construção de um jogo que, antes do curso, mostrava-se impossível.

Além disso, outros conhecimentos adquiridos auxiliaram muito na conclusão desse trabalho. Um exemplo seriam os conhecimentos adquiridos na área de engenharia de software, que mostraram como trabalhar num projeto relativamente grande com mais de um desenvolvedor, sem atropelos e mantendo um código final limpo e conciso.

É por esses conhecimentos que o trabalho final se tornou algo bastante flexível, que pode ser alterado a qualquer momento, e sem muito esforço, para outro objetivo qualquer, seja ele mudar a história do jogo ou mesmo alterar os gráficos de duas dimensões para três dimensões.

A finalização desse trabalho trouxe consigo uma imensa gratificação em termos da dedicação dispensada ao curso, e ao que foi recebido em troca por essa dedicação. É um sonho antigo a implementação de um jogo dessa natureza, e, apesar de ser bastante ultrapassado, os

conhecimentos adquiridos quebram as barreiras do que é possível ser feito.

## **5. Referências Bibliográficas**

## 5. Referências Bibliográficas

- [1] The C++ Resources Network. Disponível em:  
<<http://www.cplusplus.com>>. Acesso em: 01/09/02.
- [2] C, C++ and Java. Disponível em:  
<<http://www.doc.ic.ac.uk/lab/cplus/>>. Acesso em: 15/10/02.
- [3] References vs Pointers. Disponível em:  
<<http://www.embedded.com/story/OEG20010311S0024>>. Acesso em:  
10/09/02.
- [4] GameDev. Disponível em: <<http://www.gamedev.net/>>. Acesso em:  
20/11/02.
- [5] GamaSutra - Creating A Great Design Document. Disponível em:  
<[http://www.gamasutra.com/features/19970912/design\\_doc.htm](http://www.gamasutra.com/features/19970912/design_doc.htm)>.  
Acesso em: 20/11/02.
- [6] Allegro.cc. Disponível em: <<http://www.allegro.cc/>>. Acesso em:  
01/09/02.
- [7] StickSauce.com. Disponível em:  
<<http://www.sticksauce.com/tutorials/programming/cplusplus>>. Acesso  
em: 06/04/03.
- [8] C++ FAQ Lite. Disponível em: <<http://new-brunswick.net/workshop/c++/faq/>>. Acesso em: 09/04/03.
- [9] The Allegro GUI Clinic. Disponível em:  
<<http://agdn.netfirms.com/gui/>>. Acesso em: 20/10/02.
- [10] Lamothe, André **Windows Game Programming for Dummies**.  
Foster City, CA: IDG Books WorldWide, inc.
- [11] Lamothe, André **Game Programming in 21 Days**. Indianapolis, IN:  
Sams Publishing.
- [12] Lamothe, André **Tricks of Game Programming Gurus**.  
Indianapolis, IN: Sams Publishing.
- [13] Anderson, Greg **More Tricks of Game Programming Gurus**.  
Indianapolis, IN: Sams Publishing.

[14] Aho, Alfred V. **Compiladores Princípios, Técnicas e Ferramentas**. São Paulo, SP: Livros Técnicos e Científicos.



## **6. Anexos**

## 6.1 As 10 regras no desenvolvimento de Games

São apresentadas abaixo algumas regras veneradas por programadores de games. Claro que algumas destas regras não precisam ser seguidas para que se consiga desenvolver um bom game, mas elas representam a necessidade e o cuidado que se deve ter com a performance e a complexidade envolvida ao se desenvolver um game, por mais simples que este seja [10][11][12][13].

1. Faça funcionar primeiro, otimize o código depois.
2. Se funcionou na tela então funcionou!
3. Sempre existe um jeito melhor e mais otimizado de se implementar um algoritmo.
4. Você pode usar variáveis globais, mas apenas se necessário.
5. Nunca use instruções GOTO.
6. Tente não utilizar operações com números em ponto flutuante. Inteiros são BEM mais rápidos.
7. Use funções INLINE para pequenas funções que são chamadas sempre.
8. Não passe um monte de variáveis para uma função. Coloque-as em uma estrutura (ou objeto) e passe um ponteiro para ela.
9. Programadores de verdade fazem o seu melhor trabalho entre 1:00 e 5:00 horas da manhã.
10. Coloque comentários em todo o seu código-fonte!

## **6.2 Game Design Document**

O anexo a seguir é o Game Design Document elaborado para este game demonstrativo. Como já comentado anteriormente, um game design document contém todas as informações (em “alto nível”) a respeito do game proposto. Este documento possui diversas finalidades que vão de material introdutório como proposta de desenvolvimento a documentação e definição de requisitos funcionais do sistema [5].

**Game Design Document para:**

# TCC Quest

The Quest for Diploma

Versão # 1.00

# Sumário

Visão Geral do Game .....	4
Filosofia .....	4
Perguntas Comuns .....	4
O que é o game?.....	4
Por que criar esse game? .....	4
Aonde o game se passa? .....	4
O que eu controlo? .....	5
Qual é o enfoque principal?.....	5
O que é diferente?.....	5
Características .....	6
O Mundo do Game.....	7
Visão Geral.....	7
O Mundo Físico .....	7
Visão Geral.....	7
Locações Chave .....	7
Movimentação .....	7
Objetos.....	7
Sistema de Renderização.....	8
Visão Geral.....	8
Renderização 2D .....	8
Engine do Game.....	8
Visão Geral.....	8
Linha de Comando.....	8
Busca do Caminho.....	9
Objetivos .....	9
Inventário .....	9
Cenários .....	10
Personagens do Game .....	11

Visão Geral.....	11
O Herói.....	11
A Coruja .....	11
Interface Com o Usuário.....	12
Visão Geral.....	12
Cenário.....	12
Inventário .....	12
Informações .....	12
Linha de Comando.....	13
Caixas de Diálogo.....	13
Efeitos de Som .....	14
Visão Geral.....	14
O Game .....	15
Visão Geral.....	15
História.....	15
Horas de Game .....	15
Condições de Vitória .....	15

## **Visão Geral do Game**

### **Filosofia**

Esse game é basicamente uma forma de aplicar os conhecimentos adquiridos no Curso de Ciências da Computação da Universidade Federal de Santa Catarina em um sonho do passado: fazer games do tipo adventure gráfico.

### **Perguntas Comuns**

#### **O que é o game?**

Preso no próprio sonho, o nosso herói deve dar presentes a uma coruja que é a única criatura com o poder de fazê-lo acordar.

#### **Por que criar esse game?**

Esse game está sendo criado como trabalho de conclusão do Curso de Ciências da Computação da Universidade Federal de Santa Catarina, e envolve o uso de vários conhecimentos adquiridos no mesmo.

#### **Aonde o game se passa?**

O game se passa nos sonhos do Herói. É um mundo sem muitas regras, como nos sonhos de qualquer pessoa, em que num momento você está num lugar, e no momento seguinte, em outro completamente diferente.

## **O que eu controlo?**

O Herói, que pode explorar o seu próprio sonho coletando objetos.

## **Qual é o enfoque principal?**

É a busca dos objetos para satisfazer a Coruja. Alguns dos objetos são simples de encontrar, mas outros estão escondidos por quebra-cabeças. A idéia é a de fazer o Herói explorar o mundo.

## **O que é diferente?**

É um retorno ao antigo estilo adventure gráfico clássico. Onde era necessário pensar mais no modo como se quer fazer algo, para ser digitado, do que no quebra-cabeças em si.



## **Características**

- Analisador léxico, sintático e semântico;
- Entrada de comandos via teclado, o que aumenta as opções de game;
- Movimentação com um clique de mouse;
- Vários quebra-cabeças, que podem ser resolvidos em qualquer ordem;
- Imagens com 32-bit de cores;
- Interface simples e intuitiva;

## **O Mundo do Game**

### **Visão Geral**

O mundo é o sonho do Herói. Logo não há muitas regras onde o mesmo pode estar e aonde o mesmo pode ir.

## **O Mundo Físico**

### **Visão Geral**

O mundo é dividido em telas. Cada tela é um cenário diferente, onde o Herói pode andar e interagir.

## **Locações Chave**

Existem quatro lugares chave no game. Esses quatro lugares não tem muito em comum entre si, pois na verdade são diferentes partes do sonho do Herói. Em cada um deles existe um objeto que a Coruja quer.

## **Movimentação**

Com o clique do mouse, o Herói anda pelo cenário, desviando dos obstáculos. Ao chegar num canto de tela, o mesmo pode ser transportado para outro cenário.

## **Objetos**

Existem 5 objetos nesse game: uma flor, uma caixa, uma pedra, uma folha de papel e um martelo. A Coruja quer esses objetos, menos a

caixa.

## **Sistema de Renderização**

### **Visão Geral**

O game é em duas dimensões, e a tela do mesmo é dividida em duas partes: na maior parte, a superior, o cenário do game é desenhado, junto com os personagens e os objetos do mesmo. Na parte inferior, temos a interface, com a linha de comando e o inventário, além de outras informações, como os pontos feitos pelo jogador.

### **Renderização 2D**

A biblioteca Allegro será utilizada para desenhar a tela. O game será em duas dimensões, mas com uma visão em "perspectiva", simulando um ambiente real. Não será possível mudar o ponto de vista de cada cenário do game.

## **Engine do Game**

### **Visão Geral**

A engine do game é composta por diferentes módulos, que serão criados de tal forma que novos games possam ser implementados sem muita alteração no código-fonte.

### **Linha de Comando**

O game possui um analisador léxico, um analisador sintático e um analisador semântico que são utilizados para interpretar os comandos

dados pelo jogador na linha de comando. Dessa forma, o usuário tem uma grande interação com o game, podendo fazer vários tipos de ações diferentes, desde que a linguagem aceite o mesmo.

Apesar da linguagem simples do primeiro game, a mesma segue um padrão de ações mínimas, como ter verbos como "olhar" e "pegar", que são normais em todos os games adventure, e pode ser facilmente expandida para aceitar novas ações, objetos e personagens.

## **Busca do Caminho**

Com o clique do mouse numa área do cenário, o jogador pode fazer o Herói se mover até lá. Para isso, o game terá um algoritmo de busca do caminho, o qual tenta achar o menor caminho possível para levar o Herói do ponto de origem ao de destino, ao mesmo tempo em que contorna qualquer obstáculo que haja no caminho.

## **Objetivos**

O game permite que o usuário tenha vários objetivos, os quais podem ser dependentes um do outro, ou seja, para completar um objetivo o usuário deve primeiro completar um outro. E o game é ganho quando o jogador completa todos os objetivos primários.

Cada objetivo será colocado numa estrutura de dados, com dados como o modo como o mesmo pode ser completado, se o mesmo já foi completado, e se o mesmo é dependente de outro objetivo. Todos os objetivos serão então colocados numa lista encadeada.

## **Inventário**

O Herói pode carregar vários objetos com ele durante suas

aventuras. Os mesmos serão guardados numa lista encadeada simples.

## **Cenários**

O game possuirá uma lista com todos os cenários do game. Cada cenário do game traz consigo diversas informações, que são organizadas numa estrutura. Entre essas informações, temos os gráficos – uma imagem para ser mostrada na tela, e outra com informações especiais, como áreas onde o Herói pode andar, onde o mesmo não pode andar, e onde o mesmo irá ativar algum evento especial, como, por exemplo, a troca de cenários.

Além disso, outros dados devem ser guardados pelo cenário, como que personagens estão nele e quais objetos estão espalhados pelo mesmo.

## **Personagens do Game**

### **Visão Geral**

Existem dois personagens no game: o Herói e a Coruja.

### **O Herói**

O Herói é o personagem central do game, e é controlado pelo jogador. Cabe a ele realizar todas as ações para completar o game.

### **A Coruja**

A Coruja é um personagem não controlado pelo jogador (o chamado NPC), e serve basicamente para receber os objetos do Herói e encerrar o game.

## **Interface Com o Usuário**

### **Visão Geral**

A interface com o usuário é dividida em duas partes: na superior, e maior parte, tem-se o cenário, os personagens e os objetos, e na inferior tem-se, além de informações gerais, o inventário e a linha de comando.

### **Cenário**

Na parte superior da tela o cenário é desenhado, a partir da imagem dada pela estrutura de dados do cenário atual. Aqui também são desenhados os personagens e os objetos, de acordo com a ordem de profundidade dos mesmos no cenário.

Clicando em qualquer parte do cenário faz-se com que o Herói se mova até aquela área, se for permitido.

### **Inventário**

Uma caixa com todos os itens do inventário é desenhada, onde o usuário pode dar dois cliques e receber mais informações sobre o mesmo através de uma caixa de diálogo.

### **Informações**

Uma barra com algumas informações – como o score atual – será desenhada.

## **Linha de Comando**

Uma linha de comando onde o usuário pode digitar o que o Herói deve fazer. A linha de comando deve permitir que o usuário possa rapidamente acessar comandos já dados anteriormente, e que também possa utilizar comandos de edição de texto comuns (backspace, inserção, e por aí vai).

## **Caixas de Diálogo**

As caixas de diálogo padrão da Allegro serão alteradas para que possam aceitar mais opções, como quebra de linha e a visualização de imagens. Através dessas caixas será possível a comunicação do game com o usuário, contando a história ou explicando o que está acontecendo.



## **Efeitos de Som**

### **Visão Geral**

O game deverá ter alguns sons para criar o mundo, como, por exemplo, grilos numa noite, e um ocasional som quando um evento for efetuado, como, por exemplo, um som para marcar o momento em que o usuário completa um objetivo.

## **O Game**

### **Visão Geral**

O usuário controla o Herói em vários cenários diferentes, resolvendo quebra-cabeças e recolhendo objetos, para então entregá-los para a Coruja, que o permitirá acordar do sonho em que ele se encontra preso.

### **História**

Preso no próprio sonho, sem conseguir acordar, o Herói logo descobre que a Coruja é sua saída de lá. Só ela pode fazê-lo acordar, mas ela quer algo em troca.

O Herói deve agora explorar os seus próprios sonhos para recolher os objetos que a Coruja requer. Para assim, finalmente, acordar.

### **Horas de Game**

Como é um game de demonstração, esse é para ser um game curto, que demore no máximo 5 minutos para sua conclusão. Dessa forma o mesmo pode ser mostrado na íntegra durante a apresentação do TCC.

### **Condições de Vitória**

Quando o Herói entregar os quatro objetos (uma flor, uma pedra, uma folha de papel e um martelo) para a Coruja, o mesmo estará concluindo todos os objetivos do game.

## 6.3 Código-Fonte

É apresentado nesta seção o código-fonte desenvolvido para este projeto de conclusão de curso.

Para que o processo de compilação tenha sucesso, são necessários: o compilador GNU GCC, e a biblioteca gráfica Allegro. Para a execução do game são necessários os arquivos de recursos (sons, imagens, ícones etc).

Todas as ferramentas utilizadas, assim como os arquivos de recursos estão gravados no CD-Rom anexado a este documento.

## Arquivo: projeto.cpp

```
#include <string>
#include <iostream>
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <allegro.h> // Necessário para END_OF_MAIN

#include "./includes/resources/objetosTela.h"
#include "./includes/resources/objetosInvent.h"
#include "./includes/resources/objetosCenario.h"
#include "./includes/resources/objetosPersn.h"
#include "./includes/resources/screens.h"
#include "./includes/resources/cursors.h"
#include "./includes/resources/soundFX.h"
#include "./includes/uJogo.hpp"
#include "./includes/uPersonagem.hpp"

bool fps= false;
bool redraw= false;

void getFrames(){
    fps= true;
};

void getRedraw(){
    redraw= true;
};

int main(int argc, char *argv[]){
int frames;

    bool finalizar = false;
    string comando = "";

    Tjogo jogo("TCC Quest");
    jogo.interface->inicializaGame();
    jogo.carregaResources();
    jogo.getPersonagem()->setPosicao(400, 280, 8);
    TTela tela0(TELA0);
        tela0.setAdicionaTransicao(makecol(250, 0, 0), TELA1, 430, 100, 8);
        tela0.setAdicionaTransicao(makecol(240, 0, 0), TELA2, 730, 400, 16);
    TTela tela1(TELA1);
        tela1.setAdicionaTransicao(makecol(240, 0, 0), TELA0, 180, 460, 16);
    TTela tela2(TELA2);
        tela2.setAdicionaTransicao(makecol(240, 0, 0), TELA0, 30, 340, 8);
//OBJETOS DO JOGO:
    TObjeto obj2(OBJETO_CONTAINER, CAIXA);
        obj2.setNome("caixa");
        obj2.setSexo(FEM);
        obj2.setPosicao(192, 292, 5);
        obj2.setPosicaoJogo(192, 292);
        obj2.setEstaAbertoOuFechado(FECHADO);
    TObjeto obj3(OBJETO_OBJETO, FLOR);
        obj3.setNome("flor");
        obj3.setSexo(FEM);
        obj3.setPosicao(320, 380, 5);
        obj3.setPosicaoJogo(320, 380);
    TObjeto coruja(OBJETO_PERSONAGEM, CORUJA);
        coruja.setNome("coruja");
        coruja.setSexo(FEM);
        coruja.setPosicao(300, 240, 0);
        coruja.setPosicaoJogo(300, 200);
//cenarios na TELA0:
    TObjeto cenario_T0_porta(OBJETO_CENARIO, T0_porta);
        cenario_T0_porta.setPosicao(405, 154, 7);
    TObjeto cenario_T0_poleiro(OBJETO_CENARIO, T0_poleiro);
        cenario_T0_poleiro.setPosicao(265, 240, 10);
    TObjeto cenario_T0_ponte_esq(OBJETO_CENARIO, T0_ponte_esq);
        cenario_T0_ponte_esq.setPosicao(203, 320, 11);
```

```

TObjeto cenario_T0_ponte_dir(OBJETO_CENARIO, T0_ponte_dir);
    cenario_T0_ponte_dir.setPosicao(275, 328, 12);
TObjeto cenario_T0_poste(OBJETO_CENARIO, T0_poste);
    cenario_T0_poste.setPosicao(390, 292, 13);
TObjeto cenario_T0_arvore(OBJETO_CENARIO, T0_arvore);
    cenario_T0_arvore.setPosicao(0, 210, 13);
TObjeto cenario_T0_arbusto(OBJETO_CENARIO, T0_arbusto);
    cenario_T0_arbusto.setPosicao(0, 445, 15);
TObjeto cenario_T0_arbusto_galho(OBJETO_CENARIO, T0_arbusto_galho);
    cenario_T0_arbusto_galho.setPosicao(246, 379, 15);
//cenarios na TELA1:
TObjeto cenario_T1_arvore_meio(OBJETO_CENARIO, T1_arvore_meio);
    cenario_T1_arvore_meio.setPosicao(199, 24, 7);
TObjeto cenario_T1_canto_esq(OBJETO_CENARIO, T1_canto_esq);
    cenario_T1_canto_esq.setPosicao(0, 0, 7);
TObjeto cenario_T1_arvore1(OBJETO_CENARIO, T1_arvore1);
    cenario_T1_arvore1.setPosicao(674, 19, 8);
TObjeto cenario_T1_arvore2(OBJETO_CENARIO, T1_arvore2);
    cenario_T1_arvore2.setPosicao(600, 21, 8);
TObjeto cenario_T1_arvore_galho(OBJETO_CENARIO, T1_arvore_galho);
    cenario_T1_arvore_galho.setPosicao(357, 48, 8);
TObjeto cenario_T1_pedra_meio(OBJETO_CENARIO, T1_pedra_meio);
    cenario_T1_pedra_meio.setPosicao(350, 401, 13);
TObjeto cenario_T1_pedra(OBJETO_CENARIO, T1_pedra);
    cenario_T1_pedra.setPosicao(0, 286, 15);
//cenarios na TELA2:
TObjeto cenario_T2_arbusto(OBJETO_CENARIO, T2_arbusto);
    cenario_T2_arbusto.setPosicao(200, 143, 9);
TObjeto cenario_T2_feno(OBJETO_CENARIO, T2_feno);
    cenario_T2_feno.setPosicao(3, 217, 9);

//OBJETIVOS DO JOGO:
//TObjetivo(int id, int idAcao, int idObj, int idObjLugar, int iPts, bool necess,
//          bool repetir);
TObjetivo objet1(1, 3, FLOR, -1, 1, true, true);
TObjetivo objet2(2, 3, CAIXA, -1, 1, true, true);
TObjetivo objet3(3, 14, FLOR, CAIXA, 2, true, false);
TObjetivo objet4(4, 13, CAIXA, CORUJA, 5, true, false);
    objet4.setDependencia(3);
TObjetivo objet5(5, 1, FLOR, -1, 3, false, true);
TObjetivo objet6(6, 6, CAIXA, -1, 3, false, true);
TObjetivo objet7(7, 5, CAIXA, -1, 1, false, true);
//OBJETIVOS PROIBIDOS:
TObjetivo objetProibido1(8, 13, FLOR, CORUJA, -1, false, false); //entregar a flor
TObjetivo objetProibido2(9, 3, FLOR, CAIXA, -1, false, false); //pegar da caixa
//MONTA AS TELAS NO JOGO:
jogo.setAdicionaTela(&tela0);
    //objetos nesta tela:
    jogo.setAdicionaObj(TELA0, &obj2);

    //objetos cenario nesta tela:
    jogo.setAdicionaCenario(TELA0, &cenario_T0_porta);
    jogo.setAdicionaCenario(TELA0, &cenario_T0_poleiro);
    jogo.setAdicionaCenario(TELA0, &cenario_T0_ponte_esq);
    jogo.setAdicionaCenario(TELA0, &cenario_T0_ponte_dir);
    jogo.setAdicionaCenario(TELA0, &cenario_T0_poste);
    jogo.setAdicionaCenario(TELA0, &cenario_T0_arvore);
    jogo.setAdicionaCenario(TELA0, &cenario_T0_arbusto);
    jogo.setAdicionaCenario(TELA0, &cenario_T0_arbusto_galho);

    //personagens nesta tela:
    jogo.setAdicionaPersn(TELA0, &coruja);

jogo.setAdicionaTela(&tela1);
    //objetos cenario nesta tela:
    jogo.setAdicionaCenario(TELA1, &cenario_T1_arvore_meio);
    jogo.setAdicionaCenario(TELA1, &cenario_T1_canto_esq);
    jogo.setAdicionaCenario(TELA1, &cenario_T1_arvore1);
    jogo.setAdicionaCenario(TELA1, &cenario_T1_arvore2);
    jogo.setAdicionaCenario(TELA1, &cenario_T1_arvore_galho);
    jogo.setAdicionaCenario(TELA1, &cenario_T1_pedra_meio);
    jogo.setAdicionaCenario(TELA1, &cenario_T1_pedra);

jogo.setAdicionaTela(&tela2);

```

```

//objetos cenario nesta tela:
jogo.setAdicionaCenario(TELA2, &cenario_T2_arbusto);
jogo.setAdicionaCenario(TELA2, &cenario_T2_feno);

//objetos nesta tela:
jogo.setAdicionaObj(TELA2, &obj3);

//objetivos para se chegar ao final do jogo:
jogo.setAdicionaObjetivo(&objet1, OBJETIVO_NORMAL);
jogo.setAdicionaObjetivo(&objet2, OBJETIVO_NORMAL);
jogo.setAdicionaObjetivo(&objet3, OBJETIVO_NORMAL);
jogo.setAdicionaObjetivo(&objet4, OBJETIVO_NORMAL);
jogo.setAdicionaObjetivo(&objet5, OBJETIVO_NORMAL);
jogo.setAdicionaObjetivo(&objet6, OBJETIVO_NORMAL);
jogo.setAdicionaObjetivo(&objet7, OBJETIVO_NORMAL);

//objetivos que nao podem ser cumpridos, para nao invalidar objetivos anteriores:
jogo.setAdicionaObjetivo(&objetProibido1, OBJETIVO_PROIBIDO);
jogo.setAdicionaObjetivo(&objetProibido2, OBJETIVO_PROIBIDO);

//EVENTO DE TOCAR SOM (PARA QDO FAZ PONTO):
//(int id, int tipo, posicao pos, int idAcao, int idObj, int idObjLugar, bool repetir);
struct posicao pos;
TEvento eventoScore0(0, EVENTO_SOM, pos, 1, FLOR, -1, false);
TEvento eventoScore1(1, EVENTO_SOM, pos, 3, FLOR, -1, false);
TEvento eventoScore2(2, EVENTO_SOM, pos, 3, CAIXA, -1, false);
TEvento eventoScore3(3, EVENTO_SOM, pos, 14, FLOR, CAIXA, false);
TEvento eventoScore4(4, EVENTO_SOM, pos, 13, CAIXA, CORUJA, false);

jogo.setAdicionaEvento(&eventoScore0);
jogo.setAdicionaEvento(&eventoScore1);
jogo.setAdicionaEvento(&eventoScore2);
jogo.setAdicionaEvento(&eventoScore3);
jogo.setAdicionaEvento(&eventoScore4);
jogo.interface->mudaCursor(CURSOR_SETA);

int iX= 0;
int iY= 0;

//install_int_ex(getFrames, SECS_TO_TIMER(1));
install_int_ex(getRedraw, BPS_TO_TIMER(80));

int game_state= STATE_MENU;
bool quit_game= false;

while(!quit_game){
switch(game_state){
case STATE_MENU:
game_state= jogo.doMenuPrincipal();
break;//menu

case STATE_PLAY:
frames= 0;

// Loop principal... (playing)
finalizar= false;
//para iniciar o jogo com o personagem parado:
jogo.iniciaJogo();
iX= jogo.getPersonagem()->getPosicaoX();
iY= jogo.getPersonagem()->getPosicaoY();

while (!finalizar) {
// Vemos se foi pressionado algo... no caso de ter sido
// mandado um comando, o executamos
if (jogo.interface->verificaTeclado(comando)) {

if (comando == "sair") { // volta para o menu
game_state= STATE_MENU;
finalizar= true;
}else

jogo.executaComando(comando.c_str());

comando = "";
}
}
}
}

```

```

        if (jogo.getFimDeJogo()){
            game_state= STATE_WIN;
            finalizar= true;
        }
    }
    if ( jogo.getMudouDeTela() ){
        jogo.setMudaPosPersonagemTela(iX, iY);
        jogo.setMudouDeTela(false);
    }

    // monta string de informações para esse momento
    string valor;
    char bufferConversao[20];
    string novaInfo = "Score ";
    valor = itoa(jogo.getScore(SCORE_ATUAL),bufferConversao, 10);
    novaInfo += valor + " de ";
    valor = itoa(jogo.getScore(SCORE_TOTAL),bufferConversao, 10);
    novaInfo += valor;
    jogo.setLinhaInformacoes(novaInfo);

    if ( redraw ){
        //realiza logica do jogo, antes de atualizar a tela
        jogo.interface->verificaMouse(iX, iY);
        jogo.movePersonagem(iX, iY);

        //monta a tela no backBuffer, antes de copia-la para o video
        jogo.interface->montaBackScreen(jogo.getTela(), jogo.getPersonagem(),
jogo.getLinhaInformacoes());

        // Ao final do loop, damos um flip na tela para mostrá-la
        jogo.interface->mostraTela(false);

        redraw= false;
    }
    /*frames++;
    if ( fps ){
        cout << "\n" << frames;
        frames= 0;
        fps= false;
    }
    */
}
    highcolor_fade_out(30, NULL);
    rest(500);
break;//play

case STATE_CONTINUAR:

break;

case STATE_CREDITOS:
    jogo.doCreditos();
    game_state= STATE_MENU;
break; //credits

case STATE_WIN:
    jogo.doGanhou();
    game_state= STATE_MENU;
break;

case STATE_QUIT:
    quit_game= true;
break; //quit
}
} //while true
return 0;
}
END_OF_MAIN();

```

## Arquivo: uAbertura.hpp

```
class TAbertura{
```

```

protected:

public:
    TAbertura();
    ~TAbertura();

    void carregaResources();
};

```

## Arquivo: uAbertura.cpp

```

#include "../includes/uAbertura.hpp"

TAbertura::TAbertura(){
};

TAbertura::~~TAbertura(){
};

```

## Arquivo: uEvento.hpp

```

#ifndef __EVENTO_HPP__
#define __EVENTO_HPP__

#include "../uDefinicoesTipos.h"

class TEvento{

protected:

    int id; //id dentro do resource
    int iTipo; //SOM OU ANIMACAO..
    //para o caso do evento necessitar de uma posicao inicial:
    posicao pos;

    int iAcao; //acao q dispara esse evento
    int iObj; //objeto q deve estar envolvido na acao para q o evento dispare
    int iObjLugar; //objeto container q deve estar envolvido na acao para q o evento dispare
    bool bJaDisparou;
    bool bPodeRepetir;

public:
    TEvento(int id, int tipo, posicao pos, int idAcao, int idObj, int idObjLugar,
            bool repetir);
    ~TEvento();

    //SETTERS:
    void setJaDisparou();

    //GETTERS:
    int getTipo();
    int getId();
    posicao getPos();
    bool getPodeRepetir();
    bool getJaDisparou();

    bool getDisparaEvento(int idAcao, int idObj, int idObjLugar);
};

#endif;

```

## Arquivo: uEvento.cpp

```

#include "../includes/uEvento.hpp"

```



```

TEvento::TEvento(int id, int tipo, posicao pos, int idAcao, int idObj, int idObjLugar,
                bool repetir){
    this->id= id;
    this->iTipo= tipo;
    this->pos= pos;

    this->iAcao= idAcao;
    this->iObj= idObj;
    this->iObjLugar= idObjLugar;

    this->bPodeRepetir= repetir;
    this->bJaDisparou= false;
};

TEvento::~TEvento(){

};

//SETTERS:
void TEvento::setJaDisparou(){
    this->bJaDisparou= true;
};

//GETTERS:
int TEvento::getTipo(){
    return this->iTipo;
};

int TEvento::getId(){
    return this->id;
};

posicao TEvento::getPos(){
    return this->pos;
};

bool TEvento::getDisparaEvento(int idAcao, int idObj, int idObjLugar){
    if (this->iAcao == idAcao && this->iObj == idObj
        && this->iObjLugar == idObjLugar)
        return true;

    return false;
};

bool TEvento::getPodeRepetir(){
    return this->bPodeRepetir;
};

bool TEvento::getJaDisparou(){
    return this->bJaDisparou;
};

```

## Arquivo: uInterface.hpp

```

#ifndef __TINTERFACE_HPP__
#define __TINTERFACE_HPP__

#include <allegro.h>
#include <string>
#include "./uFuncoesAux.h"
#include "./uConstantesInterface.h"
#include "./uTela.hpp"
#include "./uMedidor.hpp"
#include "./uPrompt.hpp"

int d_fecha_dialogo(int msg, DIALOG *d, int c);

class TInterface{
private:
    BITMAP * backScreen;

```

```

//recursos:
DATAFILE *resCursors; //cursos do mouse
DATAFILE *resObjetos;
DATAFILE *resObjetosTela;
DATAFILE *resObjetosPersn;
DATAFILE *resObjetosInvent;
DATAFILE *resObjetosCenario;
DATAFILE *resScreens; //--> telas.
DATAFILE *resInterface; //imagens para a interface (inventario, etc..)
DATAFILE *resJogador[4];

posicao posInventario[20];

//gauge:
TMedidor *medidor;

TPrompt *prompt;
public:
TInterface(string titulo);
~TInterface();

void inicializaGame();

bool setaModoGrafico(int width, int height, int virtual_width = 0, int virtual_height = 0);

void mostraMensagem(string textoStr, int iTipo= -1, int id= -1);

void mostraMsgInit(string msg); //msgs de inicializacao do jogo

bool verificaTeclado(string & comando);

bool verificaMouse(int &x, int &y);

void mostraTela(bool fade);

//monta a tela, com seus sprites personagem, etc
void montaBackScreen(TTela *telaAtual, TPersonagem *persn, string LinhaInformacoes);

void mudaCursor(int qualCursor);

void desenhaLinhaInformacoes(string LinhaInformacoes);

//desenha invent. baseados nos objetos que o personagem esta carregando:
void desenhaInventario(TPersonagem *persn);

//para outras classes chamarem:
void avancaMedidor();
void destroiMedidor();

void apagaBackScreen();
};

#endif

```

## Arquivo: uInterface.cpp

```

#include "../includes/uPersonagem.hpp"
#include "../includes/resources/intro.h"
#include "../includes/resources/jogador_n.h"
#include "../includes/resources/jogador_s.h"
#include "../includes/resources/jogador_l.h"
#include "../includes/resources/jogador_o.h"
#include "../includes/resources/interface.h"
#include "../includes/resources/cursors.h"
#include "../includes/uInterface.hpp"

TInterface::TInterface(string titulo){
    this->prompt = new TPrompt;

    //inicializa o array de posicoes dos itens do inventario:

```

```

for (int i=0; i<20; i++){
    posInventario[i].posX= OFFSET_X + 15 + ((LARGURA_ITEM+5)*i+1);
    if (i<11)
        posInventario[i].posY= OFFSET_Y + 20;
    else
        posInventario[i].posY= OFFSET_Y + 25 + ALTURA_ITEM;
}

set_window_title( (char *)titulo.c_str() );

setaModoGrafico(RESOLUCAO_X,RESOLUCAO_Y);

set_mouse_range(0, 0, RESOLUCAO_X-1, RESOLUCAO_Y-127);

//demais inicializacoes foram para a classe TJogo, porque eh a la a ser criada!
}

TInterface::~TInterface(){
    delete(this->prompt);

    destroy_bitmap(backScreen);

    allegro_exit();
}

bool TInterface::setaModoGrafico(int width, int height, int virtual_width, int virtual_height){
    bool resultado;

    // Seta a profundidade
    set_color_depth(RESOLUCAO_DEPTH);

    // Seta o modo gráfico
    resultado = set_gfx_mode(GFX_AUTODETECT,
        width, height, virtual_width, virtual_height) == 0;

    // Caso o usuário teclasse alt_tab ou similar, o programa pára.
    set_display_switch_mode(SWITCH_PAUSE);

    backScreen = create_bitmap(width, height);

    prompt->setaTela(backScreen);

    return resultado;
}

void TInterface::mostraMensagem(string texto, int iTipo, int id){

    BITMAP *imagem= NULL;

    FONT * fonteAntiga = NULL;

    fonteAntiga = font;

    font = (FONT *) resInterface[FONTEIALOGO].dat;

    //<chuncho> para evitar do mouse ficar "marcado" na tela anterior...
    show_mouse(NULL);
    blit(backScreen, screen,0,0,0,0, RESOLUCAO_X, RESOLUCAO_Y-ALTURA_INVENT);
    //</chuncho>

    // DEBUG!!!
    if (texto.substr( 0, 5 ) == "DEBUG") {
        imagem = (BITMAP *) this->resObjetosInvent[1].dat;
    }
    // FIM DEBUG!!!

    if (id > -1)
        switch (iTipo){
            case IMG_OBJETO: imagem= (BITMAP *) this->resObjetos[id].dat; break;
            //case IMG_PERSN: imagem= (BITMAP *) this->resObjetos[id].dat; break; //future work! mostrar
            rosto de personagens, etc..
        }
}

```

```

// Mostra mensagem descritiva com botão de okay.
int nroLinhas = 0;
int maxLinhas = 25;
int posQuebra, posAtual;
string Linhas[maxLinhas];

// Vemos qual é o tamanho da box com o texto...
int strWidth;
int strHeight = text_height(font) + TEXTBOX_BORDER_SPACE;

int larguraMaxima = MAX_DIALOG_TEXT_WIDTH;
int comecoTexto = DIALOG_BORDER_SPACE;

if (imagem != NULL) {
    larguraMaxima -= imagem->w + 20;
    comecoTexto += imagem->w + 20;
}

// Fazemos o processamento do texto... (vemos se existe \n, se é muito grande...etc)
for (posAtual = 0; posAtual < maxLinhas; posAtual++)
    Linhas[ posAtual ] = "";

bool caracterInvalidoApagado;
bool caracterValido;
strWidth = 0;
posQuebra = -1;
posAtual = 0;
int posTabs = 0; // Marca quantos espaços de tab estão sendo processados.
// Se houver quebra de linha quando estiver
maior que zero, apaga próximos caracteres

while ((texto.length() > 0) && (nroLinhas <= maxLinhas)) {

    caracterValido = ((texto[0] >= 32) && (texto[0] <= 125));
    caracterInvalidoApagado = false;

    // Quebra de linha forçada
    if ((texto[0] == '\n') || (texto[0] == 13)) {
        if (gui_strlen(Linhas[ nroLinhas ].c_str()) > strWidth) {
            strWidth = gui_strlen(Linhas[ nroLinhas ].c_str());
        }

        posAtual = 0;
        posQuebra = -1;
        nroLinhas++;
        strHeight += text_height(font);
        caracterValido = false;
    }

    // Caracter de tabulação...
    if ((texto[0] == '\t') || (texto[0] == 9)) {

        texto.erase(0, 1);
        caracterInvalidoApagado = true;

        int indiceTab;

        indiceTab = Linhas[ nroLinhas ].length() % TAB_ESPACOS;
        indiceTab = TAB_ESPACOS - indiceTab;

        posTabs = indiceTab + posTabs;

        for (; indiceTab > 0; indiceTab--)
            texto = " " + texto;

        caracterValido = false;
    }

    if (!caracterValido) {

        if (!caracterInvalidoApagado)
            texto.erase(0, 1);

    } else {

```

```

if (gui_strlen(string(Linhas[ nroLinhas ] + texto[0]).c_str()) >= larguraMaxima) {
    if (posQuebra > 0) {
        if (posTabs > 0) {
            texto.erase(0, posTabs);
            posTabs = 0;
        }
        texto = Linhas[ nroLinhas ].substr( posQuebra + 1, Linhas[ nroLinhas ].length() ) +
texto;
        Linhas[ nroLinhas ].erase(posQuebra + 1, Linhas[ nroLinhas ].length() );
    };

    strWidth = larguraMaxima;

    posAtual = 0;
    posQuebra = -1;
    nroLinhas++;
    strHeight += text_height(font);
} else {
    if (texto[0] == ' ') {
        if (posTabs > 0) {
            posTabs--;
        }
        posQuebra = posAtual;
    }
    Linhas[ nroLinhas ] += texto[0];
    texto.erase(0, 1);
    posAtual++;
}
}

if (gui_strlen(Linhas[ nroLinhas ].c_str()) > strWidth) {
    strWidth = gui_strlen(Linhas[ nroLinhas ].c_str());
}

nroLinhas++;

if (strWidth < MIN_DIALOG_TEXT_WIDTH) {
    strWidth = MIN_DIALOG_TEXT_WIDTH;
}

if (imagem != NULL) {
    if (strHeight < TEXTBOX_BORDER_SPACE + imagem->h) {
        strHeight = TEXTBOX_BORDER_SPACE + imagem->h;
    }
}

if (strHeight > MAX_DIALOG_TEXT_HEIGHT) {
    strHeight = MAX_DIALOG_TEXT_HEIGHT;
}

int diagHeight = (2 * DIALOG_BORDER_SPACE) + strHeight;
int diagWidth = (2 * DIALOG_BORDER_SPACE) + strWidth;

if (imagem != NULL) {
    diagWidth += imagem->w + 20;
}

// Criamos a caixa...
int indiceControle = 0;

indiceControle = 4 + nroLinhas;
if (imagem != NULL) {
    indiceControle++;
}

```

```

};

DIALOG mensagem[indiceControle];

/* (dialog proc)
   (x) (y)
   (w) (h)
   (fg) (bg) (key) (flags)
   (dl) (d2) (dp) (dp2) (dp3)
*/

indiceControle = 0;

mensagem[indiceControle++] = (DIALOG) { d_shadow_box_proc,
  0, 0,
  diagWidth, diagHeight,
  GUI_COR_FG, GUI_COR_FG, 0, 0,
  0, 0, NULL, NULL, NULL };

mensagem[indiceControle++] = (DIALOG) { d_shadow_box_proc,
  DIALOG_BORDER, DIALOG_BORDER,
  diagWidth - (2 * DIALOG_BORDER), diagHeight - (2 *
DIALOG_BORDER),

  GUI_COR_BG, GUI_COR_BG, 0, 0,
  0, 0, NULL, NULL, NULL };

if (imagem != NULL) {
  mensagem[indiceControle++] = (DIALOG) { d_bitmap_proc,
  DIALOG_BORDER_SPACE, DIALOG_BORDER_SPACE,
  imagem->w, imagem->h,
  0, 0, 0, 0,
  0, 0, (void *) imagem, NULL, NULL };
}

for (int indice = 0; indice < nroLinhas; indice++) {
  mensagem[indice + indiceControle] = (DIALOG) { d_text_proc,
  começoTexto, DIALOG_BORDER_SPACE + text_height(font) *
indice,
  strWidth, text_height(font),
  GUI_COR_TEXTO_DIALOGO, GUI_COR_BG, 0, 0,
  0, 0, (void *)Linhas[indice].c_str(), NULL, NULL };
}

indiceControle += nroLinhas;

mensagem[indiceControle++] = (DIALOG) { d_fecha_dialogo,
  0, 0,
  0, 0,
  0, 0, 0, 0,
  0, 0, NULL, NULL, NULL };

mensagem[indiceControle++] = (DIALOG) { NULL, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, NULL, NULL, NULL };

centre_dialog(mensagem);

popup_dialog(mensagem, 0);

font = fonteAntiga;
}

bool TInterface::verificaTeclado(string & comando) {
  // Retorna true se foi pressionado ENTER
  return prompt->atualiza(comando);
}

bool TInterface::verificaMouse(int &x, int &y){
  if (mouse_needs_poll())
    poll_mouse();

  //se jogador apertou botao:
  if (mouse_b & 1){
    x= mouse_x;
    y= mouse_y;
    return true;
  }
}

```

```

    }

    return false;
};

void TInterface::mostraTela(bool fade){
    // Mostra o back buffer...
    if (fade){
        highcolor_fade_in(backScreen, 30);
    }else{
        acquire_screen();
        blit(backScreen, screen,0,0,0,0, RESOLUCAO_X, RESOLUCAO_Y);
        release_screen();
    }
};

void TInterface::mostraMsgInit(string msg){
    static int y= 0;

    y+= 10;

    textout(screen, font, msg.c_str(), 5, y, makecol(255,255,255));
};

void TInterface::montaBackScreen(TTela *telaAtual, TPersonagem *persn, string LinhaInformacoes){
    /* TODO (victorleel#9#): mudar o modo de desenho da tela. Ao inves de
        desenha-la o tempo todo, desenhar apenas qdo
        necessario (p/ evitar Flickering)*/

    int i, ii, dir;
    bool impPersn= false;
    TObjeto *tmp;
    BITMAP *imgPersn;

    //lo desenha o fundo:
    masked_blit((BITMAP *) resScreens[telaAtual->getIdImagem()].dat, backScreen, 0,0,0,0, RESOLUCAO_X,
RESOLUCAO_Y);

    //desenha os objetos q estao na tela atual:
    for (i=0; i<MAX_OBJETOS; i++){
        if ( (tmp= telaAtual->getObjeto(i)) != NULL ){
            draw_sprite(backScreen, (BITMAP *) resObjetosTela[tmp->getId()].dat,
tmp->getPosicaoX()-((BITMAP *) resObjetosTela[tmp->getId()].dat->w/2),
tmp->getPosicaoY()-((BITMAP *) resObjetosTela[tmp->getId()].dat->h);
        }
    }

    //desenha personagem, objetos na ordem correta:
    for (i=0; i<MAX_OBJETOS_CENARIO; i++){
        if ( (tmp= telaAtual->getCenario(i)) != NULL ){
            if ( tmp->getNivelProfundidade() < persn->getNivelProfundidade() ){
                //imprime objeto_cenario:
                draw_sprite(backScreen, (BITMAP *) resObjetosCenario[tmp->getId()].dat,
tmp->getPosicaoX(), tmp->getPosicaoY());
            }else{
                //imprime personagem:
                dir= persn->getDirecao();
                imgPersn= (BITMAP *) resJogador[dir][persn->getFrame(dir)].dat;
                draw_sprite(backScreen, imgPersn, persn->getPosicaoX()-(imgPersn->w/2),
persn->getPosicaoY()-imgPersn->h);
                impPersn= true;
                //agora termina de imprimir os cenarios:
                for (ii= i; ii<MAX_OBJETOS_CENARIO; ii++){
                    if ( (tmp= telaAtual->getCenario(ii)) != NULL ){
                        draw_sprite(backScreen, (BITMAP *) resObjetosCenario[tmp->getId()].dat,
tmp->getPosicaoX(), tmp->getPosicaoY());
                    }
                    break;//sai do loop mais externo
                }
            }else
                break; // se for NULL, eh pq acabou a lista de objetos_cenario
        }
    }
    if (!impPersn){
        //se ainda nao imprimiu personagem, o faz:
        dir= persn->getDirecao();
    }
}

```

```

    imgPersn= (BITMAP *) resJogador[dir][persn->getFrame(dir)].dat;
    draw_sprite(backScreen, imgPersn, persn->getPosicaoX()-(imgPersn->w/2),
    persn->getPosicaoY()-(imgPersn->h);
}

//desenha os objetos_persn:
for (i=0; i<MAX_OBJETOS_PERSN; i++)
    if ( (tmp= telaAtual->getPersn(i)) != NULL ){
        draw_sprite(backScreen, (BITMAP *) resObjetosPersn[tmp->getId()].dat,
        tmp->getPosicaoX()-((BITMAP *) resObjetosTela[tmp->getId()].dat->w/2),
        tmp->getPosicaoY()-((BITMAP *) resObjetosTela[tmp->getId()].dat->h);
    }

// Desenha interface...
draw_sprite(backScreen, (BITMAP *) resInterface[INTERFACE].dat, INTERFACE_X, INTERFACE_Y);

// Desenha o score
desenhaLinhaInformacoes(LinhaInformacoes);

//agora desenha o inventario:
desenhaInventario(persn);

// E o prompt...
prompt->desenhaNaTela();

//necessario pq cada vez q dah um blit, o mouse eh apagado...
show_mouse(backScreen);
};

void TInterface::mudaCursor(int qualCursor){
    show_mouse(NULL);

    set_mouse_sprite((BITMAP *) resCursors[qualCursor].dat);

    show_mouse(screen);
};

void TInterface::inicializaGame(){
    DATAFILE *intro= load_datafile("./resources/intro.dat");

    show_mouse(NULL);

    blit((BITMAP *) intro[0].dat, screen, 0, 0, 0, 0, 800, 600);
    this->medidor= new TMedidor(20);

    if ( (this->resScreens= load_datafile("./resources/screens.dat")) == NULL ){
        allegro_message("nErro fatal ao carregar arquivo de recurso - screens!!");
        exit(1);
    }
    this->medidor->avanca();
    rest(200);

    if ( (this->resObjetos= load_datafile("./resources/objetos.dat")) == NULL ){
        allegro_message("Erro fatal ao carregar arquivo de recurso - objetos!!");
        exit(1);
    }
    this->medidor->avanca();
    rest(200);

    if ( (this->resObjetosTela= load_datafile("./resources/objetosTela.dat")) == NULL ){
        allegro_message("Erro fatal ao carregar arquivo de recurso - objetosTela!!");
        exit(1);
    }
    this->medidor->avanca();
    rest(200);

    if ( (this->resObjetosInvent= load_datafile("./resources/objetosInvent.dat")) == NULL ){
        allegro_message("Erro fatal ao carregar arquivo de recurso - objetosInvent!!");
        exit(1);
    }
    this->medidor->avanca();
    rest(200);
}

```



```

if ( (this->resObjetosCenario= load_datafile("./resources/objetosCenario.dat")) == NULL ){
    allegro_message("Erro fatal ao carregar arquivo de recurso - objetosCenario!!");
    exit(1);
}
this->medidor->avanca();
rest(200);

if ( (this->resObjetosPersn= load_datafile("./resources/objetosPersn.dat")) == NULL ){
    allegro_message("Erro fatal ao carregar arquivo de recurso - objetosPersonagem!!");
    exit(1);
}
this->medidor->avanca();
rest(200);

if ( (this->resJogador[0]= load_datafile("./resources/jogador_n.dat")) == NULL ){
    allegro_message("Erro fatal ao carregar arquivo de recurso - Jogador_n!!");
    exit(1);
}
this->medidor->avanca();
rest(200);

if ( (this->resJogador[1]= load_datafile("./resources/jogador_l.dat")) == NULL ){
    allegro_message("Erro fatal ao carregar arquivo de recurso - Jogador_l!!");
    exit(1);
}
this->medidor->avanca();
rest(200);

if ( (this->resJogador[2]= load_datafile("./resources/jogador_s.dat")) == NULL ){
    allegro_message("Erro fatal ao carregar arquivo de recurso - Jogador_s!!");
    exit(1);
}
this->medidor->avanca();
rest(200);

if ( (this->resJogador[3]= load_datafile("./resources/jogador_o.dat")) == NULL ){
    allegro_message("Erro fatal ao carregar arquivo de recurso - Jogador_o!!");
    exit(1);
}
this->medidor->avanca();
rest(200);

if ( (this->resInterface= load_datafile("./resources/interface.dat")) == NULL ){
    allegro_message("Erro fatal ao carregar arquivo de recurso - Interface!");
    exit(1);
}
this->medidor->avanca();
rest(200);

if ( (this->resCursors= load_datafile("./resources/cursors.dat")) == NULL ){
    allegro_message("Erro fatal ao carregar arquivo de recurso - cursor!");
    exit(1);
}
this->medidor->avanca();
rest(200);

//libera memoria dos recursos de abertura:
unload_datafile(intro);
};

void TInterface::desenhaInventario(TPersonagem *persn){
    int posicao= 0;
    TObjeto *tmp;

    //lo desenha a moldura de fundo:
    // Removido por Rafael em 14/Jun/2003 : draw_sprite(backScreen, (BITMAP *)
resInterface[INVENTARIO].dat, OFFSET_X, OFFSET_Y);

    //agora desenha os itens que o personagem esta carregando
    for (int i=0; i< 20; i++){
        if ( (tmp= persn->getObjeto(i)) != NULL ){
            draw_sprite(backScreen, (BITMAP *) resObjetosInvent[tmp->getId()].dat,
                posInventario[i].posX, posInventario[i].posY);
        }
    }
}

```

```

};

void TInterface::avancaMedidor(){
    this->medidor->avanca();
};

void TInterface::destroiMedidor(){
    delete(this->medidor);
};

int d_fecha_dialogo(int msg, DIALOG *d, int c) {

    switch(msg) {
        case MSG_XCHAR:
            return D_CLOSE;
    }

    return D_OK;
};

void TInterface::desenhaLinhaInformacoes(string LinhaInformacoes) {
    int velhoModo = text_mode(-1);
    textout_centre(backScreen, font, LinhaInformacoes.c_str(), (INFO_X_FINAL - INFO_X_INICIO) /
2, INFO_Y_INICIO + 3, GUI_COR_TEXTO);
    text_mode(velhoModo);
};

void TInterface::apagaBackScreen(){
    clear(backScreen);
};

```

## Arquivo: uInterpretador.hpp

```

#ifndef __TINTERPRETADOR_HPP__
#define __TINTERPRETADOR_HPP__

#include "./uLexico.hpp"
#include "./uResposta.hpp"

class TInterpretador{

public:
    //CONSTRUTORES:
    TInterpretador();
    ~TInterpretador();
    TResposta* AnaliseSintatica(string frase);

protected:
    typedef void (TInterpretador::*PACAO_PARSER)(TToken *token); //define um tipo ponteiro para
fcao membro

    TLexico *scan; // nome do objeto SCANNER
    PACAO_PARSER ACOES_DO_PARSER[MAX_ACOES_PARSER]; //array de ponteiros para funcoes-membro
    TResposta *resp;
    string frase;

    bool EhTerminal(int codigo);
    bool EhNaoTerminal(int codigo);
    bool EhAcaoSemantica(int codigo);
    bool EhAcaoDoParser(int codigo);

    // ACOES "SEMANTICAS" A SEREM EXECUTADAS DIRETAMENTE PELO PARSER:
    void acao19(TToken *token);
    void acao20(TToken *token);
    void acao21(TToken *token);
    void acao22(TToken *token);
    void acao23(TToken *token);
};

#endif

```

## Arquivo: uInterpretador.cpp

```
#include "../includes/uConstantesMax.h"
#include "../includes/uInterpretador.hpp"
#include "../includes/uConstantesGramatica.hpp"

TInterpretador::TInterpretador(){
    this->scan = new TLexico();

    ACOES_DO_PARSER[0]= &TInterpretador::acao19;
    ACOES_DO_PARSER[1]= &TInterpretador::acao20;
    ACOES_DO_PARSER[2]= &TInterpretador::acao21;
    ACOES_DO_PARSER[3]= &TInterpretador::acao22;
    ACOES_DO_PARSER[4]= &TInterpretador::acao23;
}

// Destrutor da Classe TPARSER
TInterpretador::~TInterpretador(){
    delete(scan);
}

// Funcoes Auxiliares *****

bool TInterpretador::EhTerminal(int codigo){
    return ((codigo >= PRIM_T) && (codigo<=ULTI_T));
}

bool TInterpretador::EhNaoTerminal(int codigo){
    return ((codigo>=PRIM_NT) && (codigo<=ULTI_NT));
}

bool TInterpretador::EhAcaoSemantica(int codigo){
    return ((codigo>=PRIM_ACAO) && (codigo<PRIM_ACAO+NRO_ACOES_SEMANTICAS));
}

bool TInterpretador::EhAcaoDoParser(int codigo){
    return ((codigo>=PRIM_ACAO_PARSER) && (codigo<=ULTI_ACAO_PARSER));
}

// ACOES "SEMANTICAS" A SEREM EXECUTADAS DIRETAMENTE PELO PARSER:
void TInterpretador::acao19(TToken *token){
    this->resp->setSexo(0, SEXO_FEMININO);
}

void TInterpretador::acao20(TToken *token){
    this->resp->setSexo(0, SEXO_MASCULINO);
}

void TInterpretador::acao21(TToken *token){
    this->resp->setSexo(1, SEXO_FEMININO);
};

void TInterpretador::acao22(TToken *token){
    this->resp->setSexo(1, SEXO_MASCULINO);
};

void TInterpretador::acao23(TToken *token){
    this->resp->setFrase(token->getPalavra());
};

/////FUNCAO PRINCIPAL:
TResposta* TInterpretador::AnaliseSintatica(string frase){
    int PilhaDeAS[511];
    int numeroAcoes, codigo, topo, inicio, atual, fim, inicioProducao,
        fimProducao, iIndice;
    bool continuar= true;

    topo= 1;
    PilhaDeAS[topo]= DOLAR;           // Inicializando a pilha com dolar
```

```

topo++;
PilhaDeAS[topo]= SIMBOLO_INICIAL; // Simbolo inicial da gramatica
TToken token;
iIndice= 0;

// Algoritmo para executar a analise sintatica
this->resp= new TResposta();
this->scan->setFraseAtual(frase);

token= this->scan->getProxToken();
codigo= token.getId();

while (continuar){
    if (EhTerminal(PilhaDeAS[topo]))
        if (PilhaDeAS[topo] == codigo){ // reconhe o simbolo
            if (PilhaDeAS[topo] == IDENTIFICADOR)
                this->resp->setNome(iIndice++, token.getPalavra());
            // Retira terminal do topo da pilha
            topo--;

            while (EhAcaoSemantica(PilhaDeAS[topo])){//chama acao semantica
                if (EhAcaoDoParser(PilhaDeAS[topo])){
                    (*this.*ACOES_DO_PARSER[PilhaDeAS[topo]-PRIM_ACAO_PARSER])(&token); //chama acao
semantica do proprio parser...
                }else //se for acao do parser, jah chama fcao correspondente
                    this->resp->setAcao(PilhaDeAS[topo]-ULTI_NT);
                topo--;
            }

            if (continuar){
                // Proximo simbolo do AL
                token= this->scan->getProxToken();
                codigo= token.getId();
                continuar= (codigo != 0);
            }
        }else // erro sintatico
            continuar= false;
    else
        if (PilhaDeAS[topo] == SENTENCA_VAZIA)
            topo--; // retira sentenca vazia da pilha
        else
            if (EhNaoTerminal(PilhaDeAS[topo])){
                inicio= ACOES[PilhaDeAS[topo]-PRIM_NT].inicio;
                fim= ACOES[PilhaDeAS[topo]-PRIM_NT+1].inicio;
                // fazer pesquisa binaria dentro das acoes possiveis para o NT
                fim--;
                atual= fim;
                do{
                    if (codigo < TABELA[atual].codigo)
                        fim= atual;
                    else
                        if (codigo > TABELA[atual].codigo)
                            inicio= atual+1;
                        else{
                            inicio= atual;
                            fim= atual;
                        }
                    atual= (fim+inicio)/2;
                }while (fim > inicio);

                if (codigo == TABELA[atual].codigo){
                    // Substituir o nao terminal pela sua producao
                    inicioProducao= PRODUCOES[TABELA[atual].producao];
                    fimProducao= PRODUCOES[TABELA[atual].producao+1];
                    do{ // coloca o primeiro simbolo do lado direito no topo
                        fimProducao= fimProducao-1;
                        PilhaDeAS[topo]= GRAMATICA[fimProducao];
                        topo++;
                    }while (fimProducao != inicioProducao);
                    topo--;
                }else
                    continuar= false;
            }else
                if (EhAcaoSemantica(PilhaDeAS[topo])){//chama acao semantica

```

```

        if (EhAcaoDoParser(PilhaDeAS[topo])){
            (*this.*ACOES_DO_PARSER[topo])(&token); //chama acao semantica do proprio
parser...
        }else //marca acao semantica a ser executada por TJogo
            this->resp->setAcao(PilhaDeAS[topo]-ULTI_NT);
            topo--;
        }else
            if (PilhaDeAS[topo] == DOLAR)
                continuar= false; // FIM DA ANALISE
            else
                topo--; // simbolo sem acao
    }//while
    if (PilhaDeAS[topo] == DOLAR)
        this->resp->setStatus(SINTATICO_OK);
    else
        this->resp->setStatus(SINTATICO_ERRO);
    return resp;
}

```

## Arquivo: uJogo.hpp

```

#ifndef __TJOGO_HPP__
#define __TJOGO_HPP__

#include <math.h>

#include "../resources/soundFX.h"
#include "../resources/soundTrack.h"
#include "../uConstantes.h"
#include "../uConstantesMax.h"
#include "../uFuncoesAux.h"
#include "../uTela.hpp"
#include "../uObjetivo.hpp"
#include "../uPersonagem.hpp"
#include "../uInterpretador.hpp"
#include "../uInterface.hpp"

class TJogo{
protected:
    typedef void (TJogo::*pACAO)(TResposta *); //define um tipo ponteiro para fcao membro
    pACAO ACOES[MAX_ACOES]; //array de ponteiros para funcoes-membro

    string LinhaInformacoes;

    TTela *telas[MAX_TELAS];
    TObjetivo *objetivos[MAX_OBJETIVOS];
    TObjetivo *objetivosProibidos[MAX_OBJETIVOS]; //seta (acao, Obj, ObjLugar) q nao pode ser
realizado.
    TEvento *eventos[MAX_EVENTOS];
    TPersonagem jogador;
    TInterpretador interpretador;

    bool ptosVisitados[800][600]; //p/ pathFind. Eh grande.. eu sei, eu sei
    int niveisProfundidade[480]; //look-up table pra nivel de profundidade

    int iScoreAtual; //pontos no jogo no momento
    int iScoreTotal; //maximo de pontos q se pode fazer no jogo
    int iTelaAtual; //tela em que o personagem esta no momento
    bool bFimDeJogo; //acabou o jogo? true qdo cumpriu todos os objetivos necessarios
    bool bMudouDeTela; //para sinalizar qdo houver transicao de tela
        int iTelaAnt; //id da tela em que o personagem estava antes da transicao de tela
        //meio chuncho.. mas fazer o q??

    //RESOURCES:
    DATAFILE *resMenu;
    DATAFILE *resCreditos;
    DATAFILE *resSoundFX;
    DATAFILE *resSoundTrack;
    DATAFILE *resVirtualScreens;
    DATAFILE *resStringDescTela;
    DATAFILE *resStringDescObj;
    DATAFILE *resStringDescObjTela;

```

```

DATAFILE *resStringDescObjPersn;

//acoes semanticas:
void acao1(TResposta *resp);
void acao2(TResposta *resp);
void acao3(TResposta *resp);
void acao4(TResposta *resp);
void acao5(TResposta *resp);
void acao6(TResposta *resp);
void acao7(TResposta *resp);
void acao8(TResposta *resp);
void acao9(TResposta *resp);
void acao10(TResposta *resp);
void acao11(TResposta *resp);
void acao12(TResposta *resp); //n faz nada.. foi retirada da gramatica!
void acao13(TResposta *resp);
void acao14(TResposta *resp);
void acao15(TResposta *resp);
void acao16(TResposta *resp);
void acao17(TResposta *resp);
void acao18(TResposta *resp); //as acoes 19, 20, 21, 22 e 23 sao executadas
void acao24(TResposta *resp); // diretamente pelo parser...
void acao25(TResposta *resp);
/// FIM ACOES_SEMANTICAS

//decide se o personagem esta proximo de um objeto a ponto de pega-lo
bool personagemProximoSuficiente(TObjeto *obj);
//verifica se o objetivo jah pode ser cumprido atraves da ACAO, OBJETO, e
//OBJETO_CONTAINER fornecidos...
bool verificaObjetivos(int idAcao, int idObj, int idObjLugar);
bool verificaObjetivosProibidos(int idAcao, int idObj, int idObjLugar);

colisao getExisteObstaculo(int pX, int pY);
string getResString(int tipo, int id); //retorna uma string com descricao, cheiro, etc..
(depense do tipo)

void mudaTela(int idTela); //muda para outra tela
//para o pathfind:
int achaDirecao(int destx, int desty);
int achaNovaDirecao(int destx, int desty); //pathFind propriamente dito
bool tentaAndar(int direcao, int &destX, int &destY);
public:
TInterface *interface;

TJogo(string titulo);
~TJogo();

//executa comando enviado pelo usuario:
void executaComando(string comando);
void movePersonagem(int &destX, int &destY); //qdo chegou no destino, retorna true
void executaEvento(int idAcao, int idObj, int idObjLugar);
void carregaResources();
void iniciaJogo();
void doCreditos();
void doGanhou();
int doMenuPrincipal();

//SETTERS:
void setAdicionaTela(TTela *novaTela);
void setAdicionaObj(int idTela, TObjeto *obj); //coloca objeto em uma tela
void setAdicionaCenario(int idTela, TObjeto *objCenario);
void setAdicionaPersn(int idTela, TObjeto *objPersn);
void setAdicionaObjetivo(TObjetivo *objetivo, int iTipo);
void setAdicionaEvento(TEvento *evento);
//avisa p/ os objetivos q um objetivo foi cumprido, para q eles atualizem seus
//arrays de dependencias:
void setAvisaDependenciaCumprida(int iDep);
void setIncScoreAtual(int inc);
void setMudouDeTela(bool bSimOuNao);
void setMudaPosPersonagemTela(int &x, int &y); //muda a posicao do persn de acordo com a
nova tela
void setLinhaInformacoes(string novaInfo) { this->LinhaInformacoes = novaInfo; }

```

```

//GETTERS:
TTela *getTela() { return this->telas[iTelaAtual]; } //pega a tela atual
TTela *getTela(int idTela); //pega a tela com um id especifico
TPersonagem *getPersonagem() { return &jogador; }
bool getTodosObjetivosCumpridos();
bool getFimDeJogo() { return this->bFimDeJogo; }
bool getMudouDeTela() { return this->bMudouDeTela; }
int getScore(int iTelaAtual);
string getLinhaInformacoes() { return this->LinhaInformacoes; }

};

#endif

```

## Arquivo: uJogo.cpp

```

#include "../includes/uFuncoesAux.h"
#include "../includes/uConstantes.h"
#include "../includes/uJogo.hpp"
#include "../includes/resources/cursors.h"
#include "../includes/resources/menu.h"
#include "../includes/resources/creditos.h"

TJogo::TJogo(string titulo){
int i, nivel;

//inicializa allegro:
allegro_init();
install_keyboard();
install_mouse();
install_timer();

if (install_sound(DIGI_AUTODETECT, MIDI_AUTODETECT, NULL) < 0){
allegro_message("Erro fatal ao inicializar dispositivo de som!");
exit(1);
}

//soh agora constroi a interface, inicializando o modo grafico...
this->interface= new TInterface(titulo);

//inicializa todas as estruturas e variaveis da classe TJogo:
for (i=0; i<MAX_TELAS; i++)
this->telas[i]= NULL;
for (i=0; i<MAX_OBJETIVOS; i++)
this->objetivos[i]= NULL;
for (i=0; i<MAX_OBJETIVOS; i++)
this->objetivosProibidos[i]= NULL;
for (i=0; i<MAX_EVENTOS; i++)
this->eventos[i]= NULL;

for (i=0; i<800; i++) //reseta matriz de visitados
for (int x=0; x<600; x++)
this->ptosVisitados[i][x]= false;

//inicializa "look-up table" de nivel de profundidade (16 niveis):
nivel= -1;
for (i=0; i<480; i++){
if ( i % 30 == 0 )
nivel++;

this->niveisProfundidade[i]= nivel;
}

this->iScoreAtual= 0;
this->iScoreTotal= 0;
this->bFimDeJogo= false;
this->bMudouDeTela= false;

this->resSoundFX= NULL;

```

```

ACOES[0]= &TJogo::acao1;
ACOES[1]= &TJogo::acao2;
ACOES[2]= &TJogo::acao3;
ACOES[3]= &TJogo::acao4;
ACOES[4]= &TJogo::acao5;
ACOES[5]= &TJogo::acao6;
ACOES[6]= &TJogo::acao7;
ACOES[7]= &TJogo::acao8;
ACOES[8]= &TJogo::acao9;
ACOES[9]= &TJogo::acao10;
ACOES[10]= &TJogo::acao11;
ACOES[11]= &TJogo::acao12;
ACOES[12]= &TJogo::acao13;
ACOES[13]= &TJogo::acao14;
ACOES[14]= &TJogo::acao15;
ACOES[15]= &TJogo::acao16;
ACOES[16]= &TJogo::acao17;
ACOES[17]= &TJogo::acao18;
ACOES[18]= NULL; //esta acao sera executada diretamente no interpretador
ACOES[19]= NULL; //esta acao sera executada diretamente no interpretador
ACOES[20]= NULL; //esta acao sera executada diretamente no interpretador
ACOES[21]= NULL; //esta acao sera executada diretamente no interpretador
ACOES[22]= NULL; //esta acao sera executada diretamente no interpretador
ACOES[23]= &TJogo::acao24;
ACOES[24]= &TJogo::acao25;
};

TJogo::~TJogo(){
    for (int i=0; i<MAX_OBJETIVOS; i++)
        if (this->objetivos[i] != NULL)
            delete(this->objetivos[i]);
};

//ACOES SEMANTICAS:
void TJogo::acao1(TResposta *resp){
    TObjeto *obj;
    string msg;

    //se nome="" e status=-1, entao o comando foi: "Olhe".
    if (resp->getNome(0) == "" && resp->getSexo(0) < 0){
        if ( verificaObjetivos(1, -1, -1) && !verificaObjetivosProibidos(1, -1, -1) ){
            //executa evento correspondente, se tiver:
            executaEvento(1, -1, -1);
            interface->mostraMensagem(getResString(STR_DESC_TELA, iTelaAtual), -1, -1);
        }
    }else // no inventario:
        if ((obj= this->jogador.getObjeto(resp->getNome(0))) != NULL)
            if (obj->getSexo() == resp->getSexo(0)){
                //verifica se cumpriu algum objetivo atraves desta acao:
                if ( verificaObjetivos(1, obj->getId(), -1) && !verificaObjetivosProibidos(1, obj->getId(), -1) ){
                    //executa evento correspondente, se tiver:
                    executaEvento(1, obj->getId(), -1);
                    interface->mostraMensagem(getResString(STR_DESC_OBJ, obj->getId(), IMG_OBJETO, obj->getId()));
                }
            }else{
                msg= "\n\nHmm... Nao estou carregando ";
                if ( obj->getSexo() == FEM )
                    msg+= "nenhuma "; else msg+= "nenhum ";
                msg+= obj->getNome();
                interface->mostraMensagem(msg);
            }
        else{ //na tela:
            if ( (obj= this->telas[iTelaAtual]->getObjeto(resp->getNome(0))) != NULL &&
                obj->getSexo() == resp->getSexo(0) ){
                if ( verificaObjetivos(1, obj->getId(), -1) && !verificaObjetivosProibidos(1, obj->getId(), -1) ){
                    executaEvento(1, obj->getId(), -1);
                    interface->mostraMensagem(getResString(STR_DESC_OBJ_TELA, obj->getId()));
                }
            }else
                if ((obj= this->telas[iTelaAtual]->getPersn(resp->getNome(0))) != NULL &&
                    obj->getSexo() == resp->getSexo(0) ){

```



```

        if ( verificaObjetivos(1, obj->getId(), -1) && !verificaObjetivosProibidos(1, obj-
>getId(), -1) ){
            executaEvento(1, obj->getId(), -1);
            interface->mostraMensagem(getResString(STR_DESC_OBJ_PERSN, obj->getId()));
        }
    }
    else{
        msg= "\nHmm... Nao estou vendo "; msg+=retornaLetraSexo(resp->getSexo(0));
        msg+=" ' "; msg+= resp->getNome(0); msg+= "' por aqui...";
        interface->mostraMensagem(msg);
    }
}
};

void TJogo::acao2(TResposta *resp){
TObjeto *obj;
string msg;

    if ((obj= this->jogador.getObjeto(resp->getNome(0))) != NULL)
        if (obj->getSexo() == resp->getSexo(0)){
            if ( verificaObjetivos(2, obj->getId(), -1) && !verificaObjetivosProibidos(2, obj-
>getId(), -1) ){
                //executa evento correspondente, se tiver:
                executaEvento(2, obj->getId(), -1);
                interface->mostraMensagem(getResString(STR_DESC_OBJ, obj->getId(), IMG_OBJETO, obj-
>getId()));
            }
        }else{
            msg= "\nHmm... Nao estou carregando ";
            if ( obj->getSexo() == FEM )
                msg+= "nenhuma "; msg+= "nenhum ";
            msg+= obj->getNome();
            interface->mostraMensagem(msg);
        }
    else
        if ((obj= this->telas[iTelaAtual]->getObjeto(resp->getNome(0))) == NULL ||
obj->getSexo() != resp->getSexo(0)){
            msg= "\nHmm... Nao estou vendo "; msg+= retornaLetraSexo(resp->getSexo(0));
            msg+= " ' "; msg+= resp->getNome(0); msg+= "' por aqui...";
            interface->mostraMensagem(msg);
        }else
            if ( verificaObjetivos(2, obj->getId(), -1) && !verificaObjetivosProibidos(2, obj-
>getId(), -1) ){
                //executa evento correspondente, se tiver:
                executaEvento(2, -1, -1);
                interface->mostraMensagem(getResString(STR_DESC_OBJ_TELA, obj->getId()));
            }
        }
};

void TJogo::acao3(TResposta *resp){
TObjeto *obj, *tmp;
string msg;

    //se nome[1] for != "" entao eh pq esta tentando pegar objeto q esta dentro de outro obj
    if (resp->getNome(1).length() > 0){
        if ( (obj= this->jogador.getObjeto(resp->getNome(1))) == NULL &&
(obj= this->telas[iTelaAtual]->getObjeto(resp->getNome(1))) == NULL ||
obj->getSexo() != resp->getSexo(1) ){
            msg= "Hmm... Nao estou vendo "; msg+= retornaLetraSexo(resp->getSexo(1));
            msg+= " ' "; msg+= resp->getNome(1); msg+= "' por aqui...";
            interface->mostraMensagem(msg);
        }else
            if (obj->getTipoObjeto() == OBJETO_CONTAINER){
                if (obj->getEstaAbertoOuFechado() == ABERTO){
                    if ((tmp=obj->getRetiraObjeto(resp->getNome(0))) != NULL){
                        if ( verificaObjetivos(3, obj->getId(), tmp->getId()) &&
!verificaObjetivosProibidos(3, tmp->getId(), obj->getId()) ){
                            //executa evento correspondente, se tiver:
                            executaEvento(3, obj->getId(), tmp->getId());
                            this->jogador.setPossuiObjeto(tmp); //se tudo ok, agora o jogador carrega o obj
                            interface->mostraMensagem("Ok.");
                        }
                    }
                }else{
                    msg= retornaLetraSexo(resp->getSexo(0)); msg+= " ' "; msg+= resp->getNome(0);

```

```

        msg+= " ' nao esta dentro d"; msg+= retornaLetraSexo(resp->getSexo(1));
        msg+= " '"; msg+= resp->getNome(1); msg+= " '.";
        interface->mostraMensagem(msg);
    }
}
}else{
    msg= retornaLetraSexo(obj->getSexo()); msg+= " '"; msg+= obj->getNome();
    msg+= " ' esta fechad"; msg+= retornaLetraSexo(obj->getSexo()); msg+= " !";
    interface->mostraMensagem(msg);
}
}
}else{
    msg= "Ei! Nao se pode pegar nada de dentro de um";
    if (obj->getSexo() == FEM) msg+= 'a';
    msg+= " '"; msg+= obj->getNome();
    msg+= " !";
    interface->mostraMensagem(msg);
}
}
//senao, esta tentando pegar objeto q esta na tela:
}else
{
    if ((obj= this->telas[iTelaAtual]->getObjeto(resp->getNome(0))) == NULL ||
obj->getSexo() != resp->getSexo(0)){
        msg= "\nHmm... Nao estou vendo "; msg+= retornaLetraSexo(resp->getSexo(0));
        msg+= " '"; msg+= resp->getNome(0); msg+= " ' por aqui...";
        interface->mostraMensagem(msg);
    }
    }else
    {
        if (obj->getPodeSerPego()){
            if (this->personagemProximoSuficiente(obj)){
                if (this->jogador.getPossuiEspacoNoInventario()){
                    //PERSONAGEM PEGA O OBJETO!!
                    if ( verificaObjetivos(3, obj->getId(), -1) && !verificaObjetivosProibidos(3, obj-
>getId(), -1) ){
                        //executa evento correspondente, se tiver:
                        executaEvento(3, obj->getId(), -1);
                        this->jogador.setPossuiObjeto(this->telas[iTelaAtual]->getRetiraObjeto(obj-
>getNome()));
                        interface->mostraMensagem("Ok.");
                    }
                }
            }
            }else
            {
                interface->mostraMensagem("Nao consigo carregar mais nada!");
            }
        }
    }
    }else
    {
        interface->mostraMensagem("Hmmm.. Acho que nao estou perto o suficiente...");
    }
}
}
}
}
}
};

void TJogo::acao4(TResposta *resp){
TObjeto *obj;
string msg;

    if ((obj= this->jogador.getObjeto(resp->getNome(0))) == NULL ||
obj->getSexo() != resp->getSexo(0)){
        msg= "Hmm... Nao estou carregando "; msg+= retornaLetraSexo(resp->getSexo(0));
        msg+= " '"; msg+= resp->getNome(0); msg+= " '.";
        interface->mostraMensagem(msg);
    }
    }else
    {
        if (obj->getPodeSerLargado()){
            //executa evento correspondente, se tiver.
            //PERSONAGEM LARGA O OBJETO NO CHAO!!
            //seta a posicao do objeto para a mesma do personagem:
            if ( verificaObjetivos(4, obj->getId(), -1) && !verificaObjetivosProibidos(4, obj->getId(), -
1) ){
                //executa evento correspondente, se tiver:
                executaEvento(4, obj->getId(), -1);
                this->jogador.getObjeto(obj->getNome())->setPosicao(this->jogador.getPosicaoX(), this-
>jogador.getPosicaoY(), this->jogador.getNivelProfundidade());
                this->telas[iTelaAtual]->setPossuiObjeto(this->jogador.getRetiraObjeto(obj->getNome()));
                interface->mostraMensagem("Ok.");
            }
        }
    }
    }else
    {
        interface->mostraMensagem("Estranho... Nao consigo largar este objeto!!");
    }
}
};

```

```

void TJogo::acao5(TResposta *resp){
TObjeto *obj;
string msg;

if ( ((obj= this->telas[iTelaAtual]->getObjeto(resp->getNome(0))) == NULL &&
(obj= this->jogador.getObjeto(resp->getNome(0))) == NULL) ||
obj->getSexo() != resp->getSexo(0)){
    msg= "Hmm... Nao estou vendo "; msg+= retornaLetraSexo(resp->getSexo(0));
    msg+= " "; msg+= resp->getNome(0); msg+= " por aqui...";
    interface->mostraMensagem(msg);
}else
    if (obj->getPodeSerAbertoFechado()){
        if (this->personagemProximoSuficiente(obj) || (this->jogador.getObjeto(resp->getNome(0))) ==
obj){
            if (obj->getEstaAbertoOuFechado() == FECHADO){
                if ( verificaObjetivos(5, obj->getId(), -1) && !verificaObjetivosProibidos(5, obj-
>getId(), -1) ){
                    //executa evento correspondente, se tiver:
                    executaEvento(5, obj->getId(), -1);
                    //JOGADOR ABRE O OBJETO..
                    obj->setEstaAbertoOuFechado(ABERTO);
                    interface->mostraMensagem("Ok.");
                    //DISPARA POSSIVEL EVENTO ASSOCIADO A ESTA ACAO...
                }
            }else{
                msg= retornaLetraSexo(obj->getSexo()); msg+= " "; msg+= obj->getNome();
                msg+= "' ja esta abert"; msg+= retornaLetraSexo(obj->getSexo()); msg+= "!"";
                interface->mostraMensagem(msg);
            }
        }else
            interface->mostraMensagem("Hmmm.. Acho que nao estou perto o suficiente...");
    }else{
        msg= retornaLetraSexo(obj->getSexo()); msg+= " "; msg+= obj->getNome();
        msg+= "' nao pode ser abert"; msg+= retornaLetraSexo(obj->getSexo()); msg+= "!"";
        interface->mostraMensagem(msg);
    }
}
};

void TJogo::acao6(TResposta *resp){
TObjeto *obj;
string msg;

if ( ((obj= this->telas[iTelaAtual]->getObjeto(resp->getNome(0))) == NULL &&
(obj= this->jogador.getObjeto(resp->getNome(0))) == NULL) ||
obj->getSexo() != resp->getSexo(0)){
    msg= "Hmm... Nao estou vendo "; msg+= retornaLetraSexo(resp->getSexo(0));
    msg+= " "; msg+= resp->getNome(0); msg+= " por aqui...";
    interface->mostraMensagem(msg);
}else
    if (obj->getPodeSerAbertoFechado()){
        if (this->personagemProximoSuficiente(obj) || (this->jogador.getObjeto(resp->getNome(0))) ==
obj){
            if (obj->getEstaAbertoOuFechado() == ABERTO){
                if ( verificaObjetivos(6, obj->getId(), -1) && !verificaObjetivosProibidos(6, obj-
>getId(), -1) ){
                    //executa evento correspondente, se tiver:
                    executaEvento(6, obj->getId(), -1);
                    //JOGADOR FECHA O OBJETO..
                    obj->setEstaAbertoOuFechado(FECHADO);
                    interface->mostraMensagem("Ok.");
                    //DISPARA POSSIVEL EVENTO ASSOCIADO A ESTA ACAO...
                }
            }else{
                msg+= retornaLetraSexo(obj->getSexo()); msg+= " "; msg+= obj->getNome();
                msg+= "' ja esta fechad"; msg+= retornaLetraSexo(obj->getSexo()); msg+= "!"";
                interface->mostraMensagem(msg);
            }
        }else
            interface->mostraMensagem("Hmmm.. Acho que nao estou perto o suficiente...");
    }else{
        msg+= retornaLetraSexo(obj->getSexo()); msg+= " "; msg+= obj->getNome();
        msg+= "' nao pode ser fechad"; msg+= retornaLetraSexo(obj->getSexo()); msg+= "!"";
        interface->mostraMensagem(msg);
    }
}
}

```

```

};

void TJogo::acao7(TResposta *resp){
TObjeto *obj;
string msg;

if ((obj= this->jogador.getObjeto(resp->getNome(0))) == NULL ||
obj->getSexo() != resp->getSexo(0)){
msg= "Hmm... Nao estou carregando "; msg+= retornaLetraSexo(resp->getSexo(0));
msg+= " "; msg+= resp->getNome(0); msg+= " ";
interface->mostraMensagem(msg);
}else{
if ( verificaObjetivos(7, obj->getId(), -1) && !verificaObjetivosProibidos(7, obj->getId(), -1)
){
//executa evento correspondente, se tiver:
executaEvento(7, obj->getId(), -1);
//MOSTRA TEXTO COM A DESCRICAO DO CHEIRO DO OBJETO
interface->mostraMensagem(">> DESCRICAO DO CHEIRO DO OBJETO <<");
}
}
};

void TJogo::acao8(TResposta *resp){
TObjeto *obj;
string msg;

if ((obj= this->jogador.getObjeto(resp->getNome(0))) == NULL ||
obj->getSexo() != resp->getSexo(0)){
msg= "Hmm... Nao estou carregando "; msg+= retornaLetraSexo(resp->getSexo(0));
msg+= " "; msg+= resp->getNome(0); msg+= " ";
interface->mostraMensagem(msg);
}else
if (obj->getPodeSerComido()){
if ( verificaObjetivos(8, obj->getId(), -1) && !verificaObjetivosProibidos(8, obj->getId(), -1)
){
//executa evento correspondente, se tiver:
executaEvento(8, obj->getId(), -1);
interface->mostraMensagem("Ok.");
}
}else
interface->mostraMensagem("Epa! Nao posso comer isto!!");
};

void TJogo::acao9(TResposta *resp){
TObjeto *obj;
string msg;

if ((obj= this->jogador.getObjeto(resp->getNome(0))) == NULL ||
obj->getSexo() != resp->getSexo(0)){
msg= "Hmm... Nao estou carregando "; msg+= retornaLetraSexo(resp->getSexo(0));
msg+= " "; msg+= resp->getNome(0); msg+= " ";
interface->mostraMensagem(msg);
}else
if (obj->getPodeSerUsado()){
if ( verificaObjetivos(9, obj->getId(), -1) && !verificaObjetivosProibidos(9, obj->getId(), -1)
){
//executa evento correspondente, se tiver:
executaEvento(9, obj->getId(), -1);
interface->mostraMensagem("Ok.");
}
}else
interface->mostraMensagem("Epa! Nao sei como usar isto!!");
};

void TJogo::acao10(TResposta *resp){
TObjeto *obj;
string msg;

if ((obj= this->telas[iTelaAtual]->getObjeto(resp->getNome(0))) == NULL ||
obj->getSexo() != resp->getSexo(0)){
msg= "Hmm... Nao estou vendo "; msg+= retornaLetraSexo(resp->getSexo(0));
msg+= " "; msg+= resp->getNome(0); msg+= " por aqui...";
interface->mostraMensagem(msg);
}else

```

```

    if (obj->getPodeSerChutado()){
        if (this->personagemProximoSuficiente(obj)){
            if ( verificaObjetivos(10, obj->getId(), -1) && !verificaObjetivosProibidos(10, obj->getId(), -1) ){
                //executa evento correspondente, se tiver:
                executaEvento(10, obj->getId(), -1);
                //PERSONAGEM CHUTA O OBJETO!!
            }
        }else
            interface->mostraMensagem("Hmmm.. Acho que nao estou perto o suficiente...");
    }else
        interface->mostraMensagem("Ei! E qual o sentido disto?!?!");
};

void TJogo::aca011(TResposta *resp){
string fras, msg;

    if ( (fras= resp->getFrase()).length() > 50)
        interface->mostraMensagem("Putz... Voce vai fazer um discurso?!?!");
    else
        if (fras.length() == 0)
            interface->mostraMensagem("Sim, sim. Mas o que voce quer falar?");
        else
            if ( verificaObjetivos(11, -1, -1) && !verificaObjetivosProibidos(11, -1, -1) ){
                //executa evento correspondente, se tiver:
                executaEvento(11, -1, -1);
                interface->mostraMensagem("Voce falou: '" + fras + "'");
            }
};

void TJogo::aca012(TResposta *resp){
//essa acao n faz nada, foi retirada da gramatica, mas para n precisar
//re-estruturar todo codigo referente a gramatica, a acao permanece aqui, mas
//sem efeito pratico nenhum.... :(
};

void TJogo::aca013(TResposta *resp){
TObjeto *obj, *persn;
string msg;

    if ((obj= this->jogador.getObjeto(resp->getNome(0))) == NULL ||
obj->getSexo() != resp->getSexo(0)){
        msg= "Hmm... Nao estou carregando "; msg+= retornaLetraSexo(resp->getSexo(0));
        msg+= " '"; msg+= resp->getNome(0); msg+= " '.";
        interface->mostraMensagem(msg);
    }else{
        if ((persn= this->telas[iTelaAtual]->getPersn(resp->getNome(1))) == NULL ||
persn->getSexo() != resp->getSexo(1)){
            msg= "Hmm... Nao estou vendo "; msg+= retornaLetraSexo(resp->getSexo(1));
            msg+= " '"; msg+= resp->getNome(1); msg+= " ' por aqui...";
            interface->mostraMensagem(msg);
        }else
            if (persn->getTipoObjeto() != OBJETO_PERSONAGEM){//se obj n for personagem..
                msg= "Ei! Nao posso entregar um objeto para "; msg+= retornaLetraSexo(persn->getSexo());
                msg+= " '"; msg+= persn->getNome(); msg+= " '.";
                interface->mostraMensagem(msg);
            }else
                if (this->personagemProximoSuficiente(persn))
                    if (persn->getPodeReceberObjetos()){ //entrega obj para o personagem
                        if ( verificaObjetivos(13, obj->getId(), persn->getId()) &&
!verificaObjetivosProibidos(13, obj->getId(), persn->getId()) ){
                            //executa evento correspondente, se tiver:
                            executaEvento(13, obj->getId(), persn->getId());
                            persn->setPossuiObjeto(this->jogador.getRetiraObjeto(obj->getNome()));
                            interface->mostraMensagem("Ok.");
                        }
                    }else{ //este personagem nao pode receber objetos...
                        msg= "Esta me parecendo que el"; msg+= retornaLetraSexoAE(persn->getSexo());
                        msg+= " nao pode ou nao quer receber "; msg+= retornaLetraSexo(obj->getSexo());
                        msg+= " '"; msg+= obj->getNome(); msg+= " '.";
                        interface->mostraMensagem(msg);
                    }
            }else{
                msg= "Estou longe demais del"; msg+= retornaLetraSexoAE(persn->getSexo());
            }
};

```

```

        msg+= ".";
        interface->mostraMensagem(msg);
    }
};

void TJogo::acao14(TResposta *resp){
TObjeto *obj, *objLugar;
string msg;

if ( (obj= this->jogador.getObjeto(resp->getNome(0))) == NULL ||
obj->getSexo() != resp->getSexo(0)){
    msg= "Hmm... Nao estou carregando "; msg+= retornaLetraSexo(resp->getSexo(0));
    msg+= " "; msg+= resp->getNome(0); msg+= ".";
    interface->mostraMensagem(msg);
}else
//se n esta carregando o container:
if ( (objLugar= this->jogador.getObjeto(resp->getNome(1))) == NULL ||
objLugar->getSexo() != resp->getSexo(1) )
//se obj_lugar nao esta na tela atual:
if ( (objLugar= this->telas[iTelaAtual]->getObjeto(resp->getNome(1))) == NULL ||
objLugar->getSexo() != resp->getSexo(1)){
    msg= "Hmm... Nao estou vendo "; msg+= retornaLetraSexo(resp->getSexo(1));
    msg+= " "; msg+= resp->getNome(1); msg+= " por aqui...";
    interface->mostraMensagem(msg);
}else{ //se esta na tela:
    if ( this->personagemProximoSuficiente(objLugar)){
        if (objLugar->getEstaAbertoOuFechado() == ABERTO){
            if ( verificaObjetivos(14, obj->getId(), objLugar->getId()) &&
!verificaObjetivosProibidos(14, obj->getId(), objLugar->getId()) ){
                //executa evento correspondente, se tiver:
                executaEvento(14, obj->getId(), objLugar->getId());
                objLugar->setPossuiObjeto(this->jogador.getRetiraObjeto(obj->getNome()));
                interface->mostraMensagem("Ok.");
            }
        }else{ //estah fechado:
            msg= retornaLetraSexo(objLugar->getSexo()); msg+= " ";
            msg+= objLugar->getNome(); msg+= " esta fechad";
            msg+= retornaLetraSexo(objLugar->getSexo()); msg+= "!";
            interface->mostraMensagem(msg);
        }
    }else{ //nao estah prox. o suficiente:
        msg= "Estou longe demais del"; msg+= retornaLetraSexoAE(objLugar->getSexo());
        msg+= ".\n";
        interface->mostraMensagem(msg);
    }
}
}
else{
    if (objLugar->getEstaAbertoOuFechado() == ABERTO){
        if ( verificaObjetivos(14, obj->getId(), objLugar->getId()) &&
!verificaObjetivosProibidos(14, obj->getId(), objLugar->getId()) ){
            //executa evento correspondente, se tiver:
            executaEvento(14, obj->getId(), objLugar->getId());
            objLugar->setPossuiObjeto(this->jogador.getRetiraObjeto(obj->getNome()));
            interface->mostraMensagem("Ok.");
        }
    }else{ //estah fechado:
        msg= retornaLetraSexo(objLugar->getSexo()); msg+= " ";
        msg+= objLugar->getNome(); msg+= " esta fechad";
        msg+= retornaLetraSexo(objLugar->getSexo()); msg+= "!";
        interface->mostraMensagem(msg);
    }
}
};

void TJogo::acao15(TResposta *resp){
};

void TJogo::acao16(TResposta *resp){
};

void TJogo::acao17(TResposta *resp){
};

```

```

};

void TJogo::acao18(TResposta *resp){
string fras;

if ( (fras= resp->getFrase()).length() > 50)
interface->mostraMensagem("Putz... Voce vai fazer um discurso?!?!");
else
if (fras.length() == 0)
interface->mostraMensagem("Sim, sim. Mas o que voce quer gritar??");
else
if ( verificaObjetivos(18, -1, -1) && !verificaObjetivosProibidos(18, -1, -1) ){
//executa evento correspondente, se tiver:
executaEvento(18, -1, -1);
interface->mostraMensagem("Voce gritou: " + fras + "");
interface->mostraMensagem("Interessante, nao?");
}
}

void TJogo::acao24(TResposta *resp){

};

void TJogo::acao25(TResposta *resp){
string msg;
if ( verificaObjetivos(25, -1, -1) && !verificaObjetivosProibidos(25, -1, -1) ){
//executa evento correspondente, se tiver:
executaEvento(25, -1, -1);
msg= "Bem, você deve me dizer o que fazer.\n";
msg+= "Os comandos que eu entendo sao os seguintes:\n\n";
msg+= "olhe | olhe [a,o] <obj> | veja | veja [a,o] <obj> | examine [a,o] <obj>";
msg+= "descreva [a,o] <obj> | pegue [a,o] <obj> | pegue [a,o] <obj> [da,do] <container>";
msg+= "largue [a,o] <obj> | abra [a,o] <obj> | feche [a,o] <obj> | de [a,o] <obj> para [a,o]
<personagem>";
msg+= "\n\nE alguns outros, mas estou com preguica de ficar aqui falando. Descubra sozinho!";
}
interface->mostraMensagem(msg);
msg= "Exemplo:\n\nPara pegar uma pedra digite: 'pegue a pedra'.\n\nAh, e para sair do jogo
digite: 'sair'. ";
interface->mostraMensagem(msg);
};
///// FIM - ACOES SEMANTICAS

////////SETTERS:
void TJogo::setAdicionaTela(TTela *novaTela){
//acha a la posicao nula e coloca o endereco do novo objeto lah
for (int i=0; i<MAX_TELAS; i++)
if (this->telas[i] == NULL){
this->telas[i]= novaTela; //manda apontar para o objeto novaTela
return;
}
}

void TJogo::setAdicionaObj(int idTela, TObjeto *obj){
//acha a tela com id == idTela:
for (int i=0; i<MAX_TELAS; i++)
if (this->telas[i]->getIdImagem() == idTela){
this->telas[i]->setPossuiObjeto(obj);
return;
}
}

void TJogo::setAdicionaCenario(int idTela, TObjeto *objCenario){
//acha a tela com id == idTela:
for (int i=0; i<MAX_TELAS; i++)
if (this->telas[i]->getIdImagem() == idTela){
this->telas[i]->setPossuiCenario(objCenario);
return;
}
}

void TJogo::setAdicionaPersn(int idTela, TObjeto *objPersn){

```

```

//acha a tela com id == idTela:
for (int i=0; i<MAX_TELAS; i++)
    if (this->telas[i]->getIdImagem() == idTela){
        this->telas[i]->setPossuiPersn(objPersn);
        return;
    }
};

void TJogo::setAdicionaObjetivo(TObjetivo *objetivo, int iTipo){
    //acha a 1a posicao nula e coloca o endereco do novo objeto lah
    if (iTipo == OBJETIVO_NORMAL){
        this->iScoreTotal+= objetivo->getPtos();
        for (int i=0; i<MAX_OBJETIVOS; i++){
            if (this->objetivos[i] == NULL){
                this->objetivos[i]= objetivo;
                return;
            }
        }
    }else{ //objetivo proibido:
        for (int i=0; i<MAX_OBJETIVOS; i++){
            if (this->objetivosProibidos[i] == NULL){
                this->objetivosProibidos[i]= objetivo;
                return;
            }
        }
    }
};

void TJogo::setAvisaDependenciaCumprida(int iDep){
    //percorre os objetivos, para q eles atualizem suas dependencias:
    for (int i=0; i<MAX_OBJETIVOS; i++){
        if (this->objetivos[i] != NULL)
            this->objetivos[i]->setDependenciaCumprida(iDep);
        else return; //sai qdo encontra a 1a ocorrencia de NULL
    }
};

void TJogo::setIncScoreAtual(int inc){
    this->iScoreAtual+= inc;
};

void TJogo::setAdicionaEvento(TEvento *evento){
    //percorre os objetivos, para q eles atualizem suas dependencias:
    for (int i=0; i<MAX_EVENTOS; i++){
        if (this->eventos[i] == NULL){
            this->eventos[i]= evento;
            return;
        }
    }
};

void TJogo::setMudouDeTela(bool bSimOuNao){
    this->bMudouDeTela= bSimOuNao;
};

//GETTERS:
bool TJogo::getTodosObjetivosCumpridos(){
    for (int i=0; i<MAX_OBJETIVOS; i++){
        if (this->objetivos[i] != NULL)
            if (this->objetivos[i]->getEhNecessario() && !this->objetivos[i]->getCumprido())
                return false;
    }
    return true;
};

int TJogo::getScore(int iTotalAtual){
    if (iTotalAtual == SCORE_ATUAL)
        return this->iScoreAtual;
    else
        return this->iScoreTotal;
};

/////OUTROS:
void TJogo::executaComando(string comando){
    TResposta *pResposta;

    pResposta= this->interpretador.AnaliseSintatica(comando);
};

```



```

    if ( pResposta->getStatus() == ERRO ){
        interface->mostraMensagem("Desculpe, nao entendi...");
    }else //executa a acao apropriada:
        (*this.*ACOES[pResposta->getAcao()-1])(pResposta);

    delete(pResposta); //libera memoria alocada por TInterpretador
};

void TJogo::movePersonagem(int &destX, int &destY){
    static int destXant=-1, destYant=-1;
    int iDir;

    //para ver se a matriz de ptos jah visitados permanece a mesma ou nao:
    if ( destX != destXant || destY != destYant ){
        destXant= destX;
        destYant= destY;
        for (int y=0; y<800; y++) //reseta matriz de visitados
            for (int x=0; x<600; x++)
                this->ptosVisitados[y][x]= false;
    }

    iDir= achaDirecao(destX, destY);
    if ( iDir > -1 && iDir < 8 ){
        if ( !tentaAndar(iDir, destX, destY) )
            tentaAndar(iDir= achaNovaDirecao(destX, destY), destX, destY);

        this->jogador.setDirecao(iDir);
    }
    //rest(6);
};

bool TJogo::personagemProximoSuficiente(TObjeto *obj){
    if ( abs(this->jogador.getPosicaoX() - obj->getPosicaoXJogo() ) < PROXIMO_X &&
        abs(this->jogador.getPosicaoY() - obj->getPosicaoYJogo() ) < PROXIMO_Y )
        return true;
    return false;
};

//verifica se este objetivo ja pode ser realizado ou nao.
//a fcao retorna TRUE a nao ser q o objetivo ainda NAO possa ser realizado (isto eh,
//possua alguma dependencia).
bool TJogo::verificaObjetivos(int idAcao, int idObj, int idObjLugar){
    int ptos;
    for (int i=0; i<MAX_OBJETIVOS; i++){ //verifica se esta acao cumpriu algum objetivo
        //se cumprir objetivo c/ essa acao, marca:
        if (this->objetivos[i] != NULL &&
            this->objetivos[i]->getCumpreObjetivoCom(idAcao, idObj, idObjLugar, ptos)){
            if (!this->objetivos[i]->getCumprido()){
                if (!this->objetivos[i]->getEstaDependente()){
                    this->objetivos[i]->setCumprido();
                    this->setAvisaDependenciaCumprida(this->objetivos[i]->getId());
                    this->setIncScoreAtual(ptos);

                    //se todos os objetivos necessarios foram cumpridos, seta flag:
                    if (getTodosObjetivosCumpridos())
                        this->bFimDeJogo= true;
                    return true;
                }else{
                    interface->mostraMensagem("Nao posso fazer isto ainda...");
                    return false;
                }
            }else{ //!getEstaDependente()
            }else{ //!getCumprido()
                //se ja foi cumprido, apenas retorna true, sem contar pontos, etc...
                if (this->objetivos[i]->getPodeRepetir())
                    return true;
                else{
                    interface->mostraMensagem("Nao posso fazer isto novamente!!");
                    return false;
                }
            }
        }
    }
}
}
}
return true;
};
};

```

```

bool TJogo::verificaObjetivosProibidos(int idAcao, int idObj, int idObjLugar){
int ptos;
//se existir algum objetivo_proibido q feche c/ os parametros passados, entao
//nao pode realizar a acao em questao!
for (int i=0; i<MAX_OBJETIVOS; i++)
    if (this->objetivosProibidos[i] != NULL &&
        this->objetivosProibidos[i]->getCumpreObjetivoCom(idAcao, idObj, idObjLugar, ptos)){
        interface->mostraMensagem("Acho que isto nao seria a melhor coisa a se fazer...");
        return true;
    }
return false;
};

//percorre a lista de eventos, e se algum bater, executa-o
void TJogo::executaEvento(int idAcao, int idObj, int idObjLugar){
for (int i=0; i<MAX_EVENTOS; i++)
    if (this->eventos[i] != NULL &&
        this->eventos[i]->getDisparaEvento(idAcao, idObj, idObjLugar))
        if (!this->eventos[i]->getJaDisparou() || this->eventos[i]->getPodeRepetir()){
            this->eventos[i]->setJaDisparou();
            if (this->eventos[i]->getTipo() == EVENTO_SOM){
                //toca o som .wav :
                play_sample((SAMPLE *) this->resSoundFX[score].dat, 255, 122, 1000, 0);
            }
        }
};

void TJogo::carregaRecursos(){
if ( ( this->resSoundFX= load_datafile("./resources/soundFX.dat")) == NULL ){
    allegro_message("Erro fatal ao carregar arquivo de recurso - soundFX!!");
    exit(1);
}
this->interface->avancaMedidor();
rest(200);

if ( ( this->resSoundTrack= load_datafile("./resources/soundTrack.dat")) == NULL ){
    allegro_message("Erro fatal ao carregar arquivo de recurso - soundTrack!!");
    exit(1);
}
this->interface->avancaMedidor();
rest(200);

if ( ( this->resVirtualScreens= load_datafile("./resources/virtualScreens.dat")) == NULL ){
    allegro_message("Erro fatal ao carregar arquivo de recurso - Virtual screens!!");
    exit(1);
}
this->interface->avancaMedidor();
rest(200);

if ( ( this->resStringDescTela= load_datafile("./resources/stringDescTela.dat")) == NULL ){
    allegro_message("Erro fatal ao carregar arquivo de recurso - Strings Desc. Tela!!");
    exit(1);
}
this->interface->avancaMedidor();
rest(200);

if ( ( this->resStringDescObjTela= load_datafile("./resources/stringDescObjTela.dat")) == NULL ){
    allegro_message("Erro fatal ao carregar arquivo de recurso - Strings Desc. Objetos Tela!!");
    exit(1);
}
this->interface->avancaMedidor();
rest(200);

if ( ( this->resStringDescObj= load_datafile("./resources/stringDescObj.dat")) == NULL ){
    allegro_message("Erro fatal ao carregar arquivo de recurso - Strings Desc. Objetos!!");
    exit(1);
}
this->interface->avancaMedidor();
rest(200);

if ( ( this->resStringDescObjPersn= load_datafile("./resources/stringDescObjPersn.dat")) == NULL ){
    allegro_message("Erro fatal ao carregar arquivo de recurso - Strings Desc. Personagens!!");
    exit(1);
}
}

```

```

}
this->interface->avancaMedidor();
rest(200);

if ( (this->resMenu= load_datafile("./resources/menu.dat")) == NULL ){
    allegro_message("Erro fatal ao carregar arquivo de recurso - menu!!!");
    exit(1);
}
this->interface->avancaMedidor();
rest(200);

if ( (this->resCreditos= load_datafile("./resources/creditos.dat")) == NULL ){
    allegro_message("Erro fatal ao carregar arquivo de recurso - creditos!!!");
    exit(1);
}
this->interface->avancaMedidor();
rest(200);

//libera memoria e reinicia sistema de cores translucidas:
this->interface->destroiMedidor();
};

//verifica se existe algum obstaculo a frente do personagem e se existir, retorna o tipo
colisao TJogo::getExisteObstaculo(int pX, int pY){
    colisao res;
    int cor= getpixel((BITMAP *)this->resVirtualScreens[iTelaAtual].dat, pX, pY);

    if (cor == makecol(0, 0, 0)) //preto
        res.tipo= OBSTACULO_PARADA;
    else
        if (cor == makecol(0, 255, 0)) //verde
            res.tipo= OBSTACULO_DESVIO;
        else
            if (cor >= makecol(100, 0, 0) && cor <= makecol(255, 0, 0)){ //tons de vermelho
                res.tipo= OBSTACULO_TRANSICAO;
                res.cor= cor; //retorna a cor, para saber exatamente qual o tom de vermelho...
            }else
                res.tipo= OBSTACULO_LIVRE; //nao colidiu com nenhuma linha virtual..

    return res;
};

string TJogo::getResString(int tipo, int id){
    switch(tipo){
        case STR_DESC_OBJ_TELA:
            return (char *)resStringDescObjTela[id].dat;
            break;

        case STR_DESC_OBJ:
            return (char *)resStringDescObj[id].dat;
            break;

        case STR_DESC_TELA:
            return (char *)resStringDescTela[id].dat;
            break;

        case STR_DESC_OBJ_PERSN:
            return (char *)resStringDescObjPersn[id].dat;
            break;
    };
};

int TJogo::achaDirecao(int destx, int desty){
    int pX= this->jogador.getPosicaoX();
    int pY= this->jogador.getPosicaoY();

    try{

        if ( pX == destx )
            if ( pY > desty ){
                return DIRECAO_N;
            }else
                if ( pY < desty ){
                    return DIRECAO_S;
                }
    }
};

```

```

    }else ;
else
    if ( pX > destx )
        if ( pY == desty ){
            return DIRECAO_O;
        }else
            if ( pY > desty ){
                return DIRECAO_NO;
            }else
                if ( pY < desty ){
                    return DIRECAO_SO;
                }else ;
    else
        if ( pX < destx )
            if ( pY == desty ){
                return DIRECAO_L;
            }
            else
                if ( pY > desty ){
                    return DIRECAO_NE;
                }else
                    if ( pY < desty ){
                        return DIRECAO_SE;
                    }
}
}catch (...){ } //para evitar o GPF..!
};

int TJogo::achaNovaDirecao(int destx, int desty){
    int pX= this->jogador.getPosicaoX();
    int pY= this->jogador.getPosicaoY();
    int ptoY, ptoX, direcao, x;
    float ptos[8], menor;

//N:
if ( pY > 0 )
    if ( getExisteObstaculo(pX, pY-1).tipo == OBSTACULO_LIVRE &&
        !this->ptosVisitados[pX][pY-1] ){
        ptoX= pX;
        ptoY= pY-1;
        ptos[DIRECAO_N]= sqrt(pow(desty-ptoy, 2) + pow(destx-pX, 2));
        this->ptosVisitados[ptox][ptoy]= true;
    }else
        ptos[DIRECAO_N]= -1;
else
    ptos[DIRECAO_N]= -1;

//NE:
if ( pY > 0 )
    if ( getExisteObstaculo(pX+1, pY-1).tipo == OBSTACULO_LIVRE &&
        !this->ptosVisitados[pX+1][pY-1] ){
        ptoX= pX+1;
        ptoY= pY-1;
        ptos[DIRECAO_NE]= sqrt(pow(desty-ptoy, 2) + pow(destx-pX, 2));
        this->ptosVisitados[ptox][ptoy]= true;
    }else
        ptos[DIRECAO_NE]= -1;
else
    ptos[DIRECAO_NE]= -1;

//L:
if ( pX > 0 )
    if ( getExisteObstaculo(pX+1, pY).tipo == OBSTACULO_LIVRE &&
        !this->ptosVisitados[pX+1][pY] ){
        ptoX= pX+1;
        ptoY= pY;
        ptos[DIRECAO_L]= sqrt(pow(desty-ptoy, 2) + pow(destx-pX, 2));
        this->ptosVisitados[ptox][ptoy]= true;
    }else
        ptos[DIRECAO_L]= -1;
else
    ptos[DIRECAO_L]= -1;

//SE:

```

```

if ( pX < 800 )
    if ( getExisteObstaculo(pX+1, pY+1).tipo == OBSTACULO_LIVRE &&
        !this->ptosVisitados[pX+1][pY+1] ){
        ptoX= pX+1;
        ptoY= pY+1;
        ptos[DIRECAO_SE]= sqrt(pow(desty-ptoy, 2) + pow(destx-pX, 2));
        this->ptosVisitados[ptoX][ptoY]= true;
    }else
        ptos[DIRECAO_SE]= -1;
else
    ptos[DIRECAO_SE]= -1;

//S:
if ( pY > 600-ALTURA_INVENT )
    if ( getExisteObstaculo(pX, pY+1).tipo == OBSTACULO_LIVRE &&
        !this->ptosVisitados[pX][pY+1] ){
        ptoX= pX;
        ptoY= pY+1;
        ptos[DIRECAO_S]= sqrt(pow(desty-ptoy, 2) + pow(destx-pX, 2));
        this->ptosVisitados[ptoX][ptoY]= true;
    }else
        ptos[DIRECAO_S]= -1;
else
    ptos[DIRECAO_S]= -1;

//SO:
if ( pY > 600-ALTURA_INVENT )
    if ( getExisteObstaculo(pX-1, pY+1).tipo == OBSTACULO_LIVRE &&
        !this->ptosVisitados[pX-1][pY+1] ){
        ptoX= pX-1;
        ptoY= pY+1;
        ptos[DIRECAO_SO]= sqrt(pow(desty-ptoy, 2) + pow(destx-pX, 2));
        this->ptosVisitados[ptoX][ptoY]= true;
    }else
        ptos[DIRECAO_SO]= -1;
else
    ptos[DIRECAO_SO]= -1;

//O:
if ( pX > 0 )
    if ( getExisteObstaculo(pX-1, pY).tipo == OBSTACULO_LIVRE &&
        !this->ptosVisitados[pX-1][pY] ){
        ptoX= pX-1;
        ptoY= pY;
        ptos[DIRECAO_O]= sqrt(pow(desty-ptoy, 2) + pow(destx-pX, 2));
        this->ptosVisitados[ptoX][ptoY]= true;
    }else
        ptos[DIRECAO_O]= -1;
else
    ptos[DIRECAO_O]= -1;

//NO:
if ( pX > 0 )
    if ( getExisteObstaculo(pX-1, pY-1).tipo == OBSTACULO_LIVRE &&
        !this->ptosVisitados[pX-1][pY-1] ){
        ptoX= pX-1;
        ptoY= pY-1;
        ptos[DIRECAO_NO]= sqrt(pow(desty-ptoy, 2) + pow(destx-pX, 2));
        this->ptosVisitados[ptoX][ptoY]= true;
    }else
        ptos[DIRECAO_NO]= -1;
else
    ptos[DIRECAO_NO]= -1;

//percorre o array procurando pela menor distancia:
menor= 9999;
for ( x=0; x<7; x++)
    if ( ptos[x] > -1 && ptos[x] < menor ){
        menor= ptos[x];
        direcao= x;
    }

return direcao;
};

```

```

bool TJogo::tentaAndar(int direcao, int &destX, int &destY){
    int pX= this->jogador.getPosicaoX();
    int pY= this->jogador.getPosicaoY();
    colisao obstaculo;

    switch(direcao){
    case DIRECAO_N:
        if ( (obstaculo= getExisteObstaculo(pX, pY-1)).tipo == OBSTACULO_DESVIO )
            return false;
        else if ( obstaculo.tipo == OBSTACULO_PARADA ){
            destX= pX;
            destY= ++pY;
            return true;
        }else if ( obstaculo.tipo == OBSTACULO_TRANSICAO ){
            transicoesTela t= this->telas[this->iTelaAtual]->getTransicao(obstaculo.cor);
            this->telas[this->iTelaAtual]->setPosicaoInicialPersn(t.posX, t.posY, t.nivel);
            this->mudaTela(t.idTelaDestino);
        }else
            pY--;
        break;

    case DIRECAO_S:
        if ( (obstaculo= getExisteObstaculo(pX, pY+1)).tipo == OBSTACULO_DESVIO )
            return false;
        else if ( obstaculo.tipo == OBSTACULO_PARADA ){
            destX= pX;
            destY= --pY;
            return true;
        }else if ( obstaculo.tipo == OBSTACULO_TRANSICAO ){
            transicoesTela t= this->telas[this->iTelaAtual]->getTransicao(obstaculo.cor);
            this->telas[this->iTelaAtual]->setPosicaoInicialPersn(t.posX, t.posY, t.nivel);
            this->mudaTela(t.idTelaDestino);
        }else
            pY++;
        break;

    case DIRECAO_L:
        if ( (obstaculo= getExisteObstaculo(pX+1, pY)).tipo == OBSTACULO_DESVIO )
            return false;
        else if ( obstaculo.tipo == OBSTACULO_PARADA ){
            destX= --pX;
            destY= pY;
            return true;
        }else if ( obstaculo.tipo == OBSTACULO_TRANSICAO ){
            transicoesTela t= this->telas[this->iTelaAtual]->getTransicao(obstaculo.cor);
            this->telas[this->iTelaAtual]->setPosicaoInicialPersn(t.posX, t.posY, t.nivel);
            this->mudaTela(t.idTelaDestino);
        }else
            pX++;
        break;

    case DIRECAO_O:
        if ( (obstaculo= getExisteObstaculo(pX-1, pY)).tipo == OBSTACULO_DESVIO )
            return false;
        else if ( obstaculo.tipo == OBSTACULO_PARADA ){
            destX= ++pX;
            destY= pY;
            return true;
        }else if ( obstaculo.tipo == OBSTACULO_TRANSICAO ){
            transicoesTela t= this->telas[this->iTelaAtual]->getTransicao(obstaculo.cor);
            this->telas[this->iTelaAtual]->setPosicaoInicialPersn(t.posX, t.posY, t.nivel);
            this->mudaTela(t.idTelaDestino);
        }else
            pX--;
        break;

    case DIRECAO_NE:
        if ( (obstaculo= getExisteObstaculo(pX+1, pY-1)).tipo == OBSTACULO_DESVIO )
            return false;
        else if ( obstaculo.tipo == OBSTACULO_PARADA ){
            destX= --pX;
            destY= ++pY;
            return true;
        }
    }
}

```

```

    }else if ( obstaculo.tipo == OBSTACULO_TRANSICAO ){
        transicoesTela t= this->telas[this->iTelaAtual]->getTransicao(obstaculo.cor);
        this->telas[this->iTelaAtual]->setPosicaoInicialPersn(t.posX, t.posY, t.nivel);
        this->mudaTela(t.idTelaDestino);
    }else{
        pX++;
        pY--;
    }
}
break;

case DIRECAO_SE:
    if ( (obstaculo= getExisteObstaculo(pX+1, pY+1)).tipo == OBSTACULO_DESVIO )
        return false;
    else if ( obstaculo.tipo == OBSTACULO_PARADA ){
        destX= --pX;
        destY= --pY;
        return true;
    }else if ( obstaculo.tipo == OBSTACULO_TRANSICAO ){
        transicoesTela t= this->telas[this->iTelaAtual]->getTransicao(obstaculo.cor);
        this->telas[this->iTelaAtual]->setPosicaoInicialPersn(t.posX, t.posY, t.nivel);
        this->mudaTela(t.idTelaDestino);
    }else{
        pX++;
        pY++;
    }
}
break;

case DIRECAO_SO:
    if ( (obstaculo= getExisteObstaculo(pX-1, pY+1)).tipo == OBSTACULO_DESVIO )
        return false;
    else if ( obstaculo.tipo == OBSTACULO_PARADA ){
        destX= ++pX;
        destY= --pY;
        return true;
    }else if ( obstaculo.tipo == OBSTACULO_TRANSICAO ){
        transicoesTela t= this->telas[this->iTelaAtual]->getTransicao(obstaculo.cor);
        this->telas[this->iTelaAtual]->setPosicaoInicialPersn(t.posX, t.posY, t.nivel);
        this->mudaTela(t.idTelaDestino);
    }else{
        pX--;
        pY++;
    }
}
break;

case DIRECAO_NO:
    if ( (obstaculo= getExisteObstaculo(pX-1, pY-1)).tipo == OBSTACULO_DESVIO )
        return false;
    else if ( obstaculo.tipo == OBSTACULO_PARADA ){
        destX= ++pX;
        destY= ++pY;
        return true;
    }else if ( obstaculo.tipo == OBSTACULO_TRANSICAO ){
        transicoesTela t= this->telas[this->iTelaAtual]->getTransicao(obstaculo.cor);
        this->telas[this->iTelaAtual]->setPosicaoInicialPersn(t.posX, t.posY, t.nivel);
        this->mudaTela(t.idTelaDestino);
    }else{
        pX--;
        pY--;
    }
}
break;
};

this->ptosVisitados[pX][pY]= true;
this->jogador.setPosicao(pX, pY, niveisProfundidade[pY]);
return true;
};

void TJogo::mudaTela(int idTela){
    this->interface->mudaCursor(CURSOR_WAIT);
    interface->montaBackScreen(getTela(), getPersonagem(), getLinhaInformacoes());
    this->interface->mostraTela(false);

    rest(2000);
    this->iTelaAnt= this->iTelaAtual;
}

```

```

    this->iTelaAtual= idTela;
    this->interface->mudaCursor(CURSOR_SETA);
    this->bMudouDeTela= true;
};

void TJogo::iniciaJogo(){
    this->iTelaAtual= 0;

    interface->montaBackScreen(getTela(), getPersonagem(), getLinhaInformacoes());
    this->interface->mostraTela(true);

    play_sample((SAMPLE *)this->resSoundTrack[loop_jogo].dat, 50, 125, 1000, 1);

    interface->mostraMensagem("Voce esta aprisionado em seu proprio sonho!\n\n
    Para acordar e sair deste mundo, voce devera entregar um presente para a coruja!\n
    Mas o que seria este presente?? Hmm... Ok, darei uma dica:\n\n
    Voce devera procurar o presente pelos cenarios, mais dicas estarao disponiveis durante o jogo.\n
    Qualquer coisa, peca 'ajuda!'");
};

void TJogo::setMudaPosPersonagemTela(int &x, int &y){
    this->jogador.setPosicao(this->telas[iTelaAnt]->getPosicaoInicialPersn());

    x= this->jogador.getPosicaoX();
    y= this->jogador.getPosicaoY();
};

int TJogo::doMenuPrincipal(){
    BITMAP *tmp= create_bitmap(800, 600);
    bool dentro= false;
    bool bNovo= true;
    bool bCreditos= true;
    bool bSair= true;

    //para o caso de estar tocando a musica de fundo:
    stop_sample((SAMPLE *)this->resSoundTrack[loop_jogo].dat);

    clear_keybuf();

    show_mouse(screen);
    scare_mouse();
    blit((BITMAP *)this->resMenu[fundo].dat, tmp, 0, 0, 0, 0, 800, 600);
    highcolor_fade_in(tmp, 30);
    unscare_mouse();

    play_sample((SAMPLE *)this->resSoundTrack[loop_menu].dat, 255, 125, 1000, 1);

    while (true){
        //verifica GROSSEIRAMENTE qual opcao foi escolhida:
        if ( mouse_x > 165 && mouse_x < 255 && mouse_y > 225 && mouse_y < 252 ){
            if (!dentro){
                play_sample((SAMPLE *)this->resMenu[mouse_over].dat, 255, 125, 1000, 0);
                scare_mouse();
                rectfill(screen, 166, 226, 256, 253, makecol(255,255,255));
                draw_sprite(screen, (BITMAP *)this->resMenu[novo_sel].dat, 169, 227);
                unscare_mouse();
                bNovo= true;
            }
            dentro= true;
            if ( mouse_b & 1 ){
                play_sample((SAMPLE *)this->resMenu[mouse_click].dat, 255, 125, 1000, 0);
                rest(400);
                scare_mouse();
                highcolor_fade_out(30, (SAMPLE *)this->resSoundTrack[loop_menu].dat);
                stop_sample((SAMPLE *)this->resSoundTrack[loop_menu].dat);
                rest(500);
                return STATE_PLAY;
            }
        }else
        /*
            if ( mouse_x > 165 && mouse_x < 246 && mouse_y > 272 && mouse_y < 299 && mouse_b & 1)
                cout << "\nContinuar!";
        else
        */
            if ( mouse_x > 165 && mouse_x < 232 && mouse_y > 321 && mouse_y < 346 ){

```



```

    if (!dentro){
        play_sample((SAMPLE *)this->resMenu[mouse_over].dat, 255, 125, 1000, 0);
        scare_mouse();
        rectfill(screen, 166, 322, 233, 347, makecol(255,255,255));
        draw_sprite(screen, (BITMAP *)this->resMenu[creditos_sel].dat, 169, 322);
        unscare_mouse();
        bCreditos= true;
    }
    dentro= true;
    if ( mouse_b & 1 ){
        play_sample((SAMPLE *)this->resMenu[mouse_click].dat, 255, 125, 1000, 0);
        rest(400);
        scare_mouse();
        highcolor_fade_out(30, (SAMPLE *)this->resSoundTrack[loop_menu].dat);
        stop_sample((SAMPLE *)this->resSoundTrack[loop_menu].dat);
        rest(500);
        return STATE_CREDITOS;
    }
}
}else

if ( mouse_x > 165 && mouse_x < 208 && mouse_y > 365 && mouse_y < 390 ){
    if (!dentro){
        play_sample((SAMPLE *)this->resMenu[mouse_over].dat, 255, 125, 1000, 0);
        scare_mouse();
        rectfill(screen, 166, 366, 209, 391, makecol(255,255,255));
        draw_sprite(screen, (BITMAP *)this->resMenu[sair_sel].dat, 169, 366);
        unscare_mouse();
        bSair= true;
    }
    dentro= true;
    if ( mouse_b & 1 ){
        play_sample((SAMPLE *)this->resMenu[mouse_click].dat, 255, 125, 1000, 0);
        rest(400);
        scare_mouse();
        highcolor_fade_out(30, (SAMPLE *)this->resSoundTrack[loop_menu].dat);
        stop_sample((SAMPLE *)this->resSoundTrack[loop_menu].dat);
        rest(500);
        return STATE_QUIT;
    }
}else{
    dentro= false;
    if (bNovo){
        scare_mouse();
        draw_sprite(screen, (BITMAP *)this->resMenu[novo].dat, 169, 227);
        unscare_mouse();
        bNovo= false;
    }
    if (bCreditos){
        scare_mouse();
        draw_sprite(screen, (BITMAP *)this->resMenu[creditos].dat, 169, 322);
        unscare_mouse();
        bCreditos= false;
    }
    if (bSair){
        scare_mouse();
        draw_sprite(screen, (BITMAP *)this->resMenu[sair].dat, 169, 366);
        unscare_mouse();
        bSair= false;
    }
}
destroy_bitmap(tmp);
}
};

void TJogo::doCreditos(){
    play_sample((SAMPLE *)this->resSoundTrack[creditos].dat, 255, 125, 1000, 1);
    blit((BITMAP *)this->resCreditos[creditos1].dat, screen, 0, 0, 0, 0, 800, 600);
    rest(6500);
    blit((BITMAP *)this->resCreditos[creditos2].dat, screen, 0, 0, 0, 0, 800, 600);
    rest(6500);

    highcolor_fade_out(30, NULL);
    rest(500);
    highcolor_fade_in((BITMAP *)this->resCreditos[talento_nacional].dat, 30);
}

```

```

    rest(6000);
    highcolor_fade_out(30, (SAMPLE *)this->resSoundTrack[creditos].dat);
    rest(500);
    stop_sample((SAMPLE *)this->resSoundTrack[creditos].dat);
};

void TJogo::doGanhou(){
    show_mouse(NULL);
    interface->apagaBackScreen();

    interface->mostraMensagem("Parabens!!\n\nVoce conseguiu cumprir todos os objetivos, e agora
voltara para a realidade!!!");
    stop_sample((SAMPLE *)this->resSoundTrack[loop_jogo].dat);
    rest(1500);

    doCreditos();
};

```

## Arquivo: uLexico.hpp

```

#ifndef __TLEXICO_HPP__
#define __TLEXICO_HPP__

#include <string>
#include "uConstantesLexico.h"
#include "uToken.hpp"

class TLexico{
protected:
    int iPosicao;
    string sFrase;
    string palavrasReservadas[NUM_RESERVADAS];

public:
    TLexico();
    ~TLexico();

    TToken getProxToken();
    void setFraseAtual(string novaFrase);
    bool getPalavraEhReservada(string qualPalavra, int &id);
};

#endif

```

## Arquivo: uLexico.cpp

```

#include<stdio.h>
#include "includes/uLexico.hpp"

TLexico::TLexico(){
    this->iPosicao= 0;

    this->palavrasReservadas[0]= "olhe";
    this->palavrasReservadas[1]= "veja";
    this->palavrasReservadas[2]= "examine";
    this->palavrasReservadas[3]= "descreva";
    this->palavrasReservadas[4]= "pegue";
    this->palavrasReservadas[5]= "largue";
    this->palavrasReservadas[6]= "abra";
    this->palavrasReservadas[7]= "feche";
    this->palavrasReservadas[8]= "cheire";
    this->palavrasReservadas[9]= "coma";
    this->palavrasReservadas[10]= "use";
    this->palavrasReservadas[11]= "chute";
    this->palavrasReservadas[12]= "fale";
    this->palavrasReservadas[13]= "de";
    this->palavrasReservadas[14]= "entregue";
    this->palavrasReservadas[15]= "coloque";
}

```

```

this->palavrasReservadas[16]= "converse";
this->palavrasReservadas[17]= "ria";
this->palavrasReservadas[18]= "descanse";
this->palavrasReservadas[19]= "durma";
this->palavrasReservadas[20]= "grite";
this->palavrasReservadas[21]= "a";
this->palavrasReservadas[22]= "o";
this->palavrasReservadas[23]= "ao";
this->palavrasReservadas[24]= "para";
this->palavrasReservadas[25]= "na";
this->palavrasReservadas[26]= "no";
this->palavrasReservadas[27]= "da";
this->palavrasReservadas[28]= "do";
this->palavrasReservadas[29]= "com";
this->palavrasReservadas[30]= "ajuda";
};

TLexico::~~TLexico(){
};

TToken TLexico::getProxToken(){
    int inicio, x, estado, id;
    TToken tok;//= new TToken();

    estado= 0;
    inicio= x= this->iPosicao;
    while(true){
        switch(estado){
            case 0:
                if ( x == this->sFrase.length() )
                    estado= 4;
                else if ( this->sFrase[x] == ' ' ) //se for um espaco(separador), manda pro estado 1
                    estado= 1;
                else if ( this->sFrase[x] == '\\' )
                    estado= 2;
                break;

            case 1: //terminou uma palavra
                tok.setPalavra(this->sFrase.substr(inicio, x-inicio-1));

                this->iPosicao= x;//vai recommear daqui
                if ( this->getPalavraEhReservada(tok.getPalavra(), id) )
                    tok.setId(id);
                else
                    tok.setId(TOK_IDENTIFICADOR);
                return tok;
                break;

            case 2:
                if ( this->sFrase[x] == '\\' )
                    estado= 3;
                else if ( this->sFrase[x] == '\\0' )
                    estado= ESTADO_ERRO;
                break;

            case 3:
                tok.setPalavra(this->sFrase.substr(inicio+1, x-inicio-2));
                this->iPosicao= x+1;//vai recommear daqui
                tok.setId(TOK_LITERAL);

                return tok;
                break;

            case 4:
                tok.setId(TOK_EOL);
                return tok;
                break;

            case ESTADO_ERRO:
                tok.setId(TOK_ERRO);
                return tok;
                break;
        }//switch
        x++;
    }
};

```

```

    }//while
};

void TLexico::setFraseAtual(string novaFrase){
    this->sFrase= novaFrase + " ";
    //aproveita e reinicia a posicao inicial para analise:
    this->iPosicao= 0;
};

//retorna verdade se for reservada, e coloca o id correspondente ao token no 2o parametro
bool TLexico::getPalavraEhReservada(string qualPalavra, int &id){
    int x;

    for (x=0; x<NUM_RESERVADAS; x++)
        if ( qualPalavra == this->palavrasReservadas[x] ){
            id= x+1;
            return true;
        }

    return false;
};

```

## Arquivo: uMedidor.hpp

```

#ifndef __TMEDIDOR_HPP__
#define __TMEDIDOR_HPP__

#include <allegro.h>

class TMedidor{
protected:
    int iPixelsPorParte;
    int iPos;
    int iMaxPos;

public:
    TMedidor(int max);
    ~TMedidor();

    void avanca();
};

#endif;

```

## Arquivo: uMedidor.cpp

```

#include "../includes/uMedidor.hpp"

TMedidor::TMedidor(int max){
    this->iPos= 1;
    this->iMaxPos= max;
    //calcula qtos pixels cada parte deve ter:
    this->iPixelsPorParte= 250/max;

    //pinta o retangulo vazio na tela atual:
    rect(screen, 260, 565, 510, 572, makecol(255, 255, 255));

    set_trans_blender(0, 0, 0, 100);
    drawing_mode(DRAW_MODE_TRANS, screen, 260, 565);
};

TMedidor::~TMedidor(){
    drawing_mode(DRAW_MODE_SOLID, screen, 0, 0);
};

void TMedidor::avanca(){
    static int ultimaPos= 260;
    //se for chamado demais, apenas sai da funcao:
    if ( this->iPos < this->iMaxPos+1 ){
        rectfill(screen, ultimaPos+1, 566, 260+this->iPixelsPorParte*this->iPos++, 571, makecol(255,

```

```

255, 255));
    ultimaPos+= this->iPixelsPorParte;
}
};

```

## Arquivo: uObjeto.hpp

```

#ifndef __TOBJETIVO_HPP__
#define __TOBJETIVO_HPP__

#include "../uEvento.hpp"
#include "../uDefinicoesTipos.h"
#include "../uConstantesMax.h"

class TObjetivo{
protected:
    int id;
    bool cumprido;
    int dependencia[MAX_OBJETIVOS_DEPEND]; //quais objetivos precisam ser cumpridos antes deste

    int iAcao; //acao q cumpre esse objetivo
    int iObj; //objeto q deve estar envolvido na acao para q o objetivo se cumpra
    int iObjLugar; //objeto container q deve estar envolvido na acao para q o objetivo se cumpra
    int iPontos; //pontos a serem ganhos ao se realizar este objetivo
    bool bNecessario; //objetivo necessario para se vencer o jogo?
    bool bPodeRepetir;

public:
    TObjetivo(int id, int idAcao, int idObj, int idObjLugar, int iPtos, bool necess,
              bool repetir);
    ~TObjetivo();

    //SETTERS:
    void setDependencia(int idDepend);
    //manda o objetivo verificar se o id fornecido eh de alguma dependencia sua:
    void setDependenciaCumprida(int idDependencia);
    void setCumprido() { this->cumprido= true; }

    //GETTERS:
    bool getCumprido() { return this->cumprido; }
    bool getEstaDependente();
    bool getCumpreObjetivoCom(int idAcao, int idObj, int idObjLugar, int &ptos);
    bool getEhNecessario() { return this->bNecessario; }
    bool getPodeRepetir() { return this->bPodeRepetir; }
    int getId() { return this->id; }
    int getPtos() { return this->iPontos; }
};

#endif

```

## Arquivo: uObjeto.cpp

```

#include "../includes/uObjeto.hpp"

TObjeto::TObjeto(int iTipo, int IID){
    for (int i=0; i<MAX_OBJETOS_CONTAINER; i++)
        this->objetos[i]= NULL;

    this->id= IID;

    switch(iTipo){
        case OBJETO_OBJETO:
            this->bPodeSerPego= true;
            this->bPodeSerLargado= true;
            this->bPodeSerAbertoFechado= false;
            this->bPodeReceberObjetos= false;
            break;

```

```

    case OBJETO_CONTAINER:
        this->bPodeSerPego= true;
        this->bPodeSerLargado= true;
        this->bPodeSerAbertoFechado= true;
        this->bPodeReceberObjetos= true;
        break;

    case OBJETO_PERSONAGEM:
        this->bPodeSerPego= false;
        this->bPodeSerLargado= false;
        this->bPodeSerAbertoFechado= false;
        this->bPodeReceberObjetos= true;
        break;

    case OBJETO_CENARIO:
        this->bPodeSerPego= false;
        this->bPodeSerLargado= false;
        this->bPodeSerAbertoFechado= false;
        this->bPodeReceberObjetos= false;
        break;
};

    this->iTipoObjeto= iTipo;
};

TObjeto::~TObjeto() {
};

//////////SETTERS:
void TObjeto::setNivelProfundidade(int iNivel) {
    this->pos.nivelProfundidade= iNivel;
};

void TObjeto::setNome(string sNovoNome) {
    this->sNome= sNovoNome;
};

void TObjeto::setSexo(int iNovoSex) {
    this->iSexo= iNovoSex;
};

void TObjeto::setPosicao(int iX, int iY, int iNivel) {
    this->pos.posX= iX;
    this->pos.posY= iY;
    this->pos.nivelProfundidade= iNivel;
};

void TObjeto::setPosicaoJogo(int iX, int iY, int iNivel=0) {
    this->posJogo.posX= iX;
    this->posJogo.posY= iY;
    this->posJogo.nivelProfundidade= iNivel;
};

void TObjeto::setPodeSerPego(bool bSimOuNao) {
    this->bPodeSerPego= bSimOuNao;
};

void TObjeto::setPodeSerLargado(bool bSimOuNao) {
    this->bPodeSerLargado= bSimOuNao;
};

void TObjeto::setPodeSerAbertoFechado(bool bSimOuNao) {
    this->bPodeSerAbertoFechado= bSimOuNao;
};

void TObjeto::setEstaAbertoOuFechado(int iEstado) {
    this->iEstaAbertoOuFechado= iEstado;
};

void TObjeto::setPodeSerComido(bool bSimOuNao) {
    this->bPodeSerComido= bSimOuNao;
};

void TObjeto::setPodeSerUsado(bool bSimOuNao) {
};

```

```

    this->bPodeSerUsado= bSimOuNao;
};

void TObjeto::setPodeSerChutado(bool bSimOuNao){
    this->bPodeSerChutado= bSimOuNao;
};

void TObjeto::setPossuiObjeto(TObjeto *pObj){
    //acha a la posicao nula e coloca o endereco lah
    for (int i=0; i<MAX_OBJETOS_CONTAINER; i++)
        if (this->objetos[i] == NULL){
            this->objetos[i]= pObj;
            return;
        }
};

//////////GETTERS:
/*
int TObjeto::getNivelProfundidade(){
    return this->pos.nivelProfundidade;
};

string TObjeto::getNome(){
    return this->sNome;
};

int TObjeto::getId(){
    return this->id;
};

int TObjeto::getSexo(){
    return this->iSexo;
};

int TObjeto::getTipoObjeto(){
    return this->iTipoObjeto;
};

bool TObjeto::getPodeSerPego(){
    return this->bPodeSerPego;
};

bool TObjeto::getPodeSerLargado(){
    return this->bPodeSerLargado;
};

bool TObjeto::getPodeSerAbertoFechado(){
    return this->bPodeSerAbertoFechado;
};

int TObjeto::getEstaAbertoOuFechado(){
    return this->iEstaAbertoOuFechado;
};

int TObjeto::getPosicaoX(){
    return this->pos.posX;
};

int TObjeto::getPosicaoY(){
    return this->pos.posY;
};
*/
bool TObjeto::getPodeSerComido(){
    return this->bPodeSerComido;
};

bool TObjeto::getPodeSerUsado(){
    return this->bPodeSerUsado;
};

bool TObjeto::getPodeSerChutado(){
    return this->bPodeSerChutado;
};

```

```

bool TObjeto::getPodeReceberObjetos(){
    return this->bPodeReceberObjetos;
};

TObjeto* TObjeto::getObjeto(string nome){
    for (int i=0; i<MAX_OBJETOS_CONTAINER; i++)
        if (this->objetos[i] != NULL && this->objetos[i]->getNome() == nome)
            return this->objetos[i];
    return NULL;
};

TObjeto* TObjeto::getRetiraObjeto(string nome){
    TObjeto *tmp;

    for (int i=0; i<MAX_OBJETOS_CONTAINER; i++)
        if (this->objetos[i] != NULL && this->objetos[i]->getNome() == nome){
            tmp= this->objetos[i];
            this->objetos[i]= NULL;
            return tmp;
        }
    return NULL;
};

```

## Arquivo: uPersonagem.hpp

```

#ifndef __TPERSONAGEM_HPP__
#define __TPERSONAGEM_HPP__

#include "../uObjeto.hpp"
#include "../uDefinicoesTipos.h"
#include "../uConstantesMax.h"

class TPersonagem{
protected:
    bool parado;
    int iVelocidade;
    int iDirecao; //direcao em q o personagem esta andando (N, S, L, O)
    int iFrameNS; //frames da animacao do personagem
    int iFrameLO;
    posicao pos;
    //array de ponteiros para os objetos q o personagem esta carregando:
    TObjeto *objetos[MAX_INVENTARIO];

public:
    TPersonagem();
    ~TPersonagem();

    //SETTERS:
    void setPossuiObjeto(TObjeto *novoObj);
    void setRemoveObjeto(string nome);
    void setPosicao(int iX, int iY, int iNivel);
    void setPosicao(posicao pos) { this->pos= pos; }
    void setDirecao(int direcao);
    void setNivelProfundidade(int iQualNivel) { this->pos.nivelProfundidade= iQualNivel; }

    //GETTERS:
    //retorna NULL se nao tiver carregando o objeto em questao:
    TObjeto *getObjeto(string nome);
    TObjeto *getObjeto(int iPos) { return this->objetos[iPos]; }
    int getPosicaoX() { return this->pos.posX; }
    int getPosicaoY() { return this->pos.posY; }
    int getDirecao() { return this->iDirecao; }
    int getFrame(int direcao);
    int getNivelProfundidade() { return this->pos.nivelProfundidade; }
    bool getPossuiEspacoNoInventario();

    //retorna ponteiro para objeto e retira-o do inventario do personagem
    TObjeto* getRetiraObjeto(string nome);
};

#endif

```



## Arquivo: uPersonagem.cpp

```
#include "../includes/uPersonagem.hpp"

TPersonagem::TPersonagem(){
    for (int x=0; x<MAX_INVENTARIO; x++)
        this->objetos[x]= NULL;

    this->iFrameNS= 0;
    this->iFrameLO= 0;

    //temp:
    this->iDirecao= DIRECAO_L;
    this->iVelocidade= 5;
};

TPersonagem::~TPersonagem(){
};

void TPersonagem::setPossuiObjeto(TObjeto *novoObj){
    //acha a la posicao nula e coloca o endereco lah
    for (int i=0; i<MAX_INVENTARIO; i++)
        if (this->objetos[i] == NULL){
            this->objetos[i]= novoObj;
            return;
        }
};

//SETTERS:
void TPersonagem::setDirecao(int direcao){
    //este metodo seta os sprites adequados:
    static int tmp= 0;

    if ( tmp < this->iVelocidade ){
        tmp++;
        return;
    }else
        tmp= 0;

    if ( direcao == DIRECAO_NE || direcao == DIRECAO_SE || direcao == DIRECAO_L ){
        direcao= DIRECAO_L;
        if ( this->iFrameLO > 6 )
            this->iFrameLO= 0;
        else
            this->iFrameLO++;
    }else
        if ( direcao == DIRECAO_NO || direcao == DIRECAO_SO || direcao == DIRECAO_O ){
            direcao= DIRECAO_O;
            if ( this->iFrameLO > 6 )
                this->iFrameLO= 0;
            else
                this->iFrameLO++;
        }else
            if ( direcao == DIRECAO_N || direcao == DIRECAO_S )
                if ( this->iFrameNS > 3 )
                    this->iFrameNS= 0;
                else
                    this->iFrameNS++;

    this->iDirecao= direcao;
};

//usado qdo o personagem dah um drop no objeto:
void TPersonagem::setRemoveObjeto(string nome){
    for (int i=0; i<MAX_INVENTARIO; i++)
        if (this->objetos[i]->getNome() == nome){
            this->objetos[i]= NULL;
        }
};

void TPersonagem::setPosicao(int iX, int iY, int iNivel){
    this->pos.posX= iX;
```

```

    this->pos.posY= iY;
    this->pos.nivelProfundidade= iNivel;
};

/*void TPersonagem::setPosicao(posicao pos){
    this->pos= pos;
};

void TPersonagem::setNivelProfundidade(int iQualNivel){
    this->pos.nivelProfundidade= iQualNivel;
};
*/
//GETTERS:

TObjeto* TPersonagem::getObjeto(string nome){
    for (int i=0; i<MAX_INVENTARIO; i++)
        if (this->objetos[i] != NULL && this->objetos[i]->getNome() == nome)
            return this->objetos[i];
    return NULL;
};
/*
TObjeto* TPersonagem::getObjeto(int iPos){
    return this->objetos[iPos];
};

int TPersonagem::getPosicaoX(){
    return this->pos.posX;
};

int TPersonagem::getPosicaoY(){
    return this->pos.posY;
};
*/
bool TPersonagem::getPossuiEspacoNoInventario(){
    //percorre a lista de objetos, se nao tiver nenhuma posicao vazia, entao
    //eh porque n tem mais espaco no inventario
    for (int i=0; i<MAX_INVENTARIO; i++)
        if (this->objetos[i] == NULL)
            return true;
    return false;
};

TObjeto* TPersonagem::getRetiraObjeto(string nome){
    TObjeto *tmp;

    for (int i=0; i<MAX_INVENTARIO; i++)
        if (this->objetos[i] != NULL && this->objetos[i]->getNome() == nome){
            tmp= this->objetos[i];
            this->objetos[i]= NULL;
            return tmp;
        }
    return NULL;
};
/*
int TPersonagem::getNivelProfundidade(){
    return this->pos.nivelProfundidade;
};

int TPersonagem::getDirecao(){
    return this->iDirecao;
};
*/
int TPersonagem::getFrame(int direcao){
    //se mudou de direcao, zera o frame:
    if ( direcao == DIRECAO_L || direcao == DIRECAO_O )
        if ( direcao != this->iDirecao )
            return 0;
    else
        return this->iFrameLO;
    else
        if ( direcao == DIRECAO_S || direcao == DIRECAO_N )
            if ( direcao != this->iDirecao )
                return 0;

```

```

        else
            return this->iFrameNS;
};

```

## Arquivo: uPrompt.hpp

```

#ifndef __TPROMPT_HPP__
#define __TPROMPT_HPP__

#include <allegro.h>
#include <string>
#include "../uConstantesInterface.h"

class TPrompt{
protected:
    BITMAP * tela;
    string comandoNoPrompt;

    int posCursor; // Posição do cursor, em caracteres
    int posMinCursor; // Posição do cursor mínima, sem que invada o "> " ou similar

    string historico[TAMANHO_HISTORICO]; // Guarda os 10 últimos comandos dados
    int posNoHistorico;

    void guardaNoHistorico(string & comando);
    void navegaNoHistorico(string & texto, int posicoes);

public:
    TPrompt();
    ~TPrompt();

    void setaTela(BITMAP * novaTela);

    bool atualiza(string & comando);

    void desenhaNaTela();
};

#endif;

```

## Arquivo: uPrompt.cpp

```

#include "../includes/uPrompt.hpp"

TPrompt::TPrompt(){
    tela = 0;

    posCursor = 0;
    posMinCursor = string(LINHA_PROMPT).length() + 1;

    for (posNoHistorico = 0; posNoHistorico < TAMANHO_HISTORICO; posNoHistorico++) {
        historico[posNoHistorico] = "";
    }

    posNoHistorico = -1;
};

TPrompt::~TPrompt(){

};

void TPrompt::setaTela(BITMAP * novaTela){
    tela = novaTela;
};

bool TPrompt::atualiza(string & comando){

    comandoNoPrompt = comando;
};

```

```

    bool resultado = false;
    int tecla, teclaScan;
    string novaTecla;

    // Lê todas as teclas e as coloca em comando...
while (keypressed()) {
    tecla = readkey();
    teclaScan = tecla >> 8;
    tecla = tecla & 0xff;
    novaTecla = tecla;

switch (teclaScan) {

    case KEY_LEFT : if (posCursor > 0) posCursor--;
                    break;

    case KEY_RIGHT : if (posCursor < comando.length()) posCursor++;
                    break;

    case KEY_UP : navegaNoHistorico(comando, 1);
                 break;

    case KEY_DOWN : navegaNoHistorico(comando, -1);
                  break;

    case KEY_HOME : posCursor = 0;
                   break;

    case KEY_END : posCursor = comando.length();
                  break;

    case KEY_ENTER :
    case KEY_ENTER_PAD : resultado = true;
                        posCursor = 0;
                        guardaNoHistorico(comando);
                        break;

    case KEY_BACKSPACE : if (comando.length() > 0) {
                        if (posCursor > 0) {
                            comando = comando.erase( posCursor - 1, 1);
                            posCursor--;
                        }
                        } break;

    case KEY_DEL : if (comando.length() > 0) {
                  if (posCursor < comando.length()) {
                      comando = comando.erase( posCursor, 1);
                  }
                  } break;

    default : if ((tecla >= 32) && (tecla <= 126)) {
              if (posCursor == 0) {
                  comando = novaTecla + comando;
              } else {
                  if (posCursor >= comando.length()) {
                      comando += novaTecla;
                  } else {
                      comando = comando.substr(0, posCursor) + novaTecla +
                                  comando.substr(posCursor, comando.length());
                  }
              }
              posCursor++;
          }
    }

return resultado;
};

void TPrompt::desenhaNaTela() {

    // Desenha o prompt....

```

```

        string textoSaida, caracterAtual;
int indice, posX;
        int posYCursor, posXCursorFim, posCursorComPrompt;

//rectfill(tela, PROMPT_X_INICIO, PROMPT_Y_INICIO, PROMPT_X_FINAL, PROMPT_Y_FINAL, GUI_COR_BG);

posX          = PROMPT_X_INICIO + 3;
textoSaida    = LINHA_PROMPT + comandoNoPrompt;
posYCursor    = PROMPT_Y_INICIO + text_height(font) + 3;
posCursorComPrompt = posCursor + posMinCursor;

for (indice = 0; indice < textoSaida.length(); indice++) {
    caracterAtual = textoSaida.substr(indice,1);

    int velhoModo = text_mode(-1);

    textout(tela, font, caracterAtual.c_str(), posX, PROMPT_Y_INICIO, GUI_COR_TEXTO);

    text_mode(velhoModo);

    if (indice == posCursorComPrompt - 1) {
        // O cursor se encontra aqui... desenha...
        posXCursorFim    = posX + gui_strlen( caracterAtual.c_str() );

        line(tela, posX, posYCursor, posXCursorFim, posYCursor, GUI_COR_TEXTO);
    }

    posX += gui_strlen(caracterAtual.c_str());
}

// Se o cursor ainda não foi desenhado, desenha...
if (textoSaida.length() < posCursorComPrompt) {
    line(tela, posX, posYCursor, posX + 6, posYCursor, GUI_COR_TEXTO);
}
};

void TPrompt::guardaNoHistorico(string & comando) {

    int indice;

    if (comando != "") {

        for (indice = TAMANHO_HISTORICO - 1; indice > 0 ; indice--) {
            historico[indice] = historico[indice - 1];
        }

        historico[0] = comando;

        posNoHistorico = -1;
    }
};

void TPrompt::navegaNoHistorico(string & texto, int posicoes) {

    int novaPosicao;

    novaPosicao = posNoHistorico + posicoes;

    if ((novaPosicao >= 0) && (novaPosicao < TAMANHO_HISTORICO)) {

        if (historico[novaPosicao] != "") {

            posNoHistorico = novaPosicao;

            texto = historico[posNoHistorico];

            posCursor = texto.length();

        }
    }
}

```

```

    if (novaPosicao == -1) {
        posNoHistorico = novaPosicao;
        texto = "";
        posCursor = 0;
    }
};

```

## Arquivo: uResposta.hpp

```

#ifndef __TRESPOSTA_HPP__
#define __TRESPOSTA_HPP__

//objeto que sera devolvido como resposta a chamada ao parser
#include<string>

class TResposta{
protected:
    int iStatus; //ok ou erro
    int iSexo[2]; //sexo do id em questao
    int iAcao; //acao a ser executada pela classe TJogo
    string sNomeObj[2];
    string sFrase; //usada para armazenar o valor "literal"

public:
    //CONSTRUTORES:
    TResposta();
    ~TResposta();

    //SETTERS:
    void setStatus(int novoStatus);
    void setSexo(int iIndice, int novoSexo);
    void setAcao(int novaAcao);
    void setNome(int iIndice, string novoNome);
    void setFrase(string novaFrase);

    //GETTERS:
    int getStatus();
    int getSexo(int iIndice);
    int getAcao();
    string getNome(int iIndice);
    string getFrase();
};

#endif

```

## Arquivo: uResposta.cpp

```

#include "../includes/uResposta.hpp"

TResposta::TResposta(){
    this->sNomeObj[0]= "";
    this->sNomeObj[1]= "";
    this->sFrase= "";
    this->iSexo[0]= -1;
    this->iSexo[1]= -1;
}

TResposta::~~TResposta(){
}

//SETTERS:

void TResposta::setStatus(int novoStatus){
    this->iStatus= novoStatus;
}

```

```

}

void TResposta::setSexo(int iIndice, int novoSexo){
    this->iSexo[iIndice]= novoSexo;
}

void TResposta::setAcao(int novaAcao){
    this->iAcao= novaAcao;
}

void TResposta::setNome(int iIndice, string novoNome){
    this->sNomeObj[iIndice]= novoNome;
}

void TResposta::setFrase(string novaFrase){
    this->sFrase= novaFrase;
};

//GETTERS:

int TResposta::getStatus(){
    return this->iStatus;
}

int TResposta::getSexo(int iIndice){
    return this->iSexo[iIndice];
}

int TResposta::getAcao(){
    return this->iAcao;
}

string TResposta::getNome(int iIndice){
    return this->sNomeObj[iIndice];
}

string TResposta::getFrase(){
    return this->sFrase;
};

```

## **6.4 Artigo sobre o trabalho**

É apresentado a seguir um artigo elaborado como pré-requisito da disciplina de Projeto em Ciências da Computação II.

Este artigo aborda o trabalho de conclusão de curso apresentado neste documento.



# Projeto e Implementação de um Game estilo Adventure

Victor Simon Lee

Ciências da Computação, 2003

Departamento de Informática e Estatística - INE

Universidade Federal de Santa Catarina - UFSC, Brasil, 88040-900

victor@inf.ufsc.br

## Resumo

*Este trabalho de conclusão do curso de Ciências da Computação, trata do projeto e implementação de um software de entretenimento eletrônico, mais especificamente, um game que segue o estilo "Adventure".*

*O principal objetivo deste trabalho consiste na aplicação dos conhecimentos adquiridos durante o curso de graduação, em uma aplicação real de tamanho e complexidades consideráveis.*

**Palavras-chave:** Programação de Jogos, GDD, Allegro, Aventura.

## Abstract

*This work of conclusion of the Computer Science course, deals with the project and implementation of a electronic entertainment software, more specifically, a game, which follows the adventure style.*

*The main objective of this work consists on the application of the knowledge acquired during the graduation course, in a real application with considerable size and complexity.*

**Key-words:** Game Programming, GDD, Allegro, Adventure.

## INTRODUÇÃO

A história dos games eletrônicos

divide-se em vários períodos. Períodos estes caracterizados pela complexidade

dos games e que eram delimitadas, principalmente, pelo que os computadores podiam fazer e pela criatividade dos seus criadores.

Bem antes de placas aceleradoras 3D, bem antes de games multi-jogadores, numa época em que gráficos 2D com 256 cores eram considerados a última tecnologia e a maioria dos jogadores ainda sacrificava seus olhos nas quatro cores de um monitor CGA, os games "Adventure" dominavam uma grande parcela do mercado.

Não mais em modo texto como nos primórdios, onde o jogador lia uma descrição do cenário em que se encontrava e digitava que ações tomar, mas um pouco mais avançados, com grande parte da tela mostrando o cenário e o personagem que o jogador incorporava. Os movimentos do personagem eram controlados pelo teclado (mais tarde, por cliques do mouse) e suas ações continuavam sendo digitadas (isso também foi modificado com o passar dos anos, dispensando completamente o teclado).

Numa época em que os games não tinham enredos bem definidos, os

games "Adventure" criavam enredos mais complexos, que se dividiam em seqüências. Séries inteiras como os títulos King's Quest, Space Quest, Police Quest, Leisure Suit Larry, da produtora Sierra, ou The Secret Of Monkey Island da Lucasfilm Games fizeram história.

### **Proposta de Trabalho e Objetivos**

Esse trabalho de conclusão do curso de Ciências da Computação é uma volta aqueles tempos. Uma tentativa de recriar aqueles games antigos em estilo Adventure que marcaram época. Os objetivos principais deste trabalho são o de projetar e implementar um game estilo adventure similar ao game "King's Quest" da empresa Sierra On-Line, utilizando para isso os conhecimentos adquiridos durante o curso de Ciências da Computação assim como a utilização de técnicas e ferramentas disponíveis atualmente.

### **Implementação do Game**

Para a implementação do game proposto, foi utilizado a seguinte metodologia:

- 1 foi realizado um estudo sobre a

- biblioteca gráfica Allegro (<http://www.allegro.cc>);
- 2 foi realizado um estudo e a posterior elaboração de um "Game Design Document";
  - 3 Foi elaborado o projeto do game (em termos de implementação);
  - 4 Finalmente foram realizados a implementação e os testes do game;

A biblioteca gráfica Allegro fora utilizada para a manipulação de todas as imagens do game.

A finalidade do Game Design Document é a de documentar todos os aspectos do game a ser desenvolvido. Esta é uma técnica muito utilizada na área de desenvolvimento de games.

Procurou-se no projeto do game, utilizar o paradigma de orientação a objetos. Obtendo-se assim um bom nível de modularidade. Muitos autores nesta área não recomendam a utilização deste paradigma alegando uma possível perda de performance. Contudo, o game foi desenvolvido utilizando-se a linguagem C++ (orientada a objetos), mas tomou-se o cuidado de não utilizar muitas sub-classes ou sobrecargas de métodos e

funções, aspectos estes que poderiam degradar um pouco a performance geral do game. Desta maneira pode-se tirar proveito da organização e modularidade de uma linguagem orientada a objetos sem prejudicar a performance do sistema.

### **Interpretador de Comandos**

Como o game proposto trata-se de um "remake" de um game adventure, tornou-se necessário a implementação de um interpretador de comandos. Para tal aplicou-se as técnicas utilizadas para a construção de compiladores, tais como: analisador léxico, analisador sintático e analisador semântico. Tendo-se atenção especial ao analisador semântico, responsável pelas ações que o personagem do game deve efetuar. Assim, quando o jogador instruir o personagem a realizar alguma ação, será o analisador semântico o responsável por "comandar" o personagem (se o comando estiver léxica e sintaticamente correto).

### **Busca do Caminho**

Outro aspecto muito importante neste projeto é o algoritmo para busca do caminho (Path Find).

O jogador comanda as ações que o personagem deve realizar através de texto, mas o movimento do personagem deve ser comandado através de cliques do mouse. Sendo assim, tornou-se necessário a implementação de um algoritmo para que o personagem pudesse encontrar um caminho livre de obstáculos ao se deslocar. Assim, quando o jogador clica em uma área da tela, o personagem move-se até este ponto, desviando de possíveis obstáculos presentes em seu caminho.

O algoritmo implementado não foi o melhor existente para esta finalidade (o algoritmo não foi capaz de achar o caminho correto em algumas situações), mas foi o suficiente para realizar uma demonstração do game.

## **Discussão e Conclusões**

O game implementado conseguiu reproduzir muitos dos aspectos presentes nos games da empresa Sierra On-Line, especialmente nos títulos: King's Quest e Leisure Suit Larry. Apesar de algumas características tais como animações do cenário de fundo, inteligência artificial para os personagens não controlados

pelo jogador não terem sido implementadas, as mesmas não apresentam grandes dificuldades para serem incorporadas ao produto final. Estas implementações foram deixadas de lado devido ao tempo reduzido que a equipe dispunha para finalizar o projeto.

Finalmente, pode ser constatado o sucesso no cumprimento dos objetivos propostos no projeto, deixando em aberto ainda possibilidades para futuros aperfeiçoamentos no game.