CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA

ESCOLA TÉCNICA ESTADUAL ZONA LESTE

CHEM QUARK  
JOGO DE QUÍMICA

SÃO PAULO  
2013

CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA

ESCOLA TÉCNICA ESTADUAL ZONA LESTE

CHEM QUARK  
JOGO DE QUÍMICA

DANILO SANTOS ARAÚJO

DOUGLAS HENRIQUE MARINHO

LUCAS MARQUES GASPARINO

Projeto que será apresentado na Feira Tecnológica do Centro Paula Souza, sob a orientação do professor Jeferson Roberto de Lima.

SÃO PAULO

2013

**Resumo**

Este projeto foi realizado com o objetivo de facilitar o aprendizado de Química para alunos do Ensino Médio. O sistema utiliza conceitos de química para interagir com o conteúdo lecionado pelo professor. Os simuladores do sistema têm a função de testar o conhecimento dos alunos.

A pesquisa e o desenvolvimento do projeto foram realizados com o acompanhamento do professor de química da Unidade Escolar Etec Zona Leste, Erick Figueroa, que auxiliou o grupo com os cálculos, fórmulas de química e com indicações das áreas da química em que os alunos mais tinham problemas em estudar.

A pontuação nos jogos é uma maneira de medir o aprendizado em certa matéria, e o ranking de pontuação marca o recorde dos melhores jogadores, trazendo o objetivo de superar essa marca para os outros jogadores e fazendo com que eles obtenham o conhecimento para conseguirem deixar sua marca. Os simuladores utilizam o espírito competitivo dos usuários para fazer com que eles gostem e queiram aprender.

**Palavras-chave:** química, simulador, ensino médio, estudo.

**Abstract**

This project was performed in order to facilitate the learning of chemistry to high school students. The system takes advantage of ilustrations in chemical concepts to interact with lessons presented by the teacher. Games in the system make chemistry easier and help teachers to measure the knowledge acquired by students.

The research and the development of this project were realized with the accompaniment of the chemistry teacher from the unit ETEC ZONA LESTE, Erick Figueroa, that helped the group with the concepts, chemistry formulas and by indicating areas in chemistry where students had more problems to study.

Points in games develop rankings that, with avaliation tools, will make students want to overcome colleagues’ points. Considering that, knowledge becomes a prior importance for users. The simulators utilize the competitive spirit of the users to make them like it and to make them want to learn more.

**Key words**: Chemistry, simulator, high school, study.

Sumário

[1 Introdução 8](#_Toc372756213)

[2 Objetivo geral 9](#_Toc372756214)

[2.1 Objetivos específicos 9](#_Toc372756215)

[3 Metodologia 9](#_Toc372756216)

[4 Hipóteses 10](#_Toc372756217)

[5 Introdução ao assunto 11](#_Toc372756218)

[5.1 Informática 11](#_Toc372756219)

[5.2 Software 11](#_Toc372756220)

[6 Instituição 12](#_Toc372756221)

[6.1 Descrição 12](#_Toc372756222)

[6.2 Estrutura 13](#_Toc372756223)

[6.3 Instrutor da empresa 14](#_Toc372756224)

[7 MARKUP 15](#_Toc372756225)

[8 Diagrama de Caso de Uso 17](#_Toc372756226)

[8.1 Descrição dos Atores 17](#_Toc372756227)

[8.2 Descrição dos Casos de Uso 18](#_Toc372756228)

[8.2.1 Caso de uso – Jogar: 18](#_Toc372756229)

[8.2.2 Caso de uso - Validar Pontuação: 19](#_Toc372756230)

[8.2.3 Caso de uso – Acessar Conta: 20](#_Toc372756231)

[8.2.4 Caso de uso – Alterar Descrições: 21](#_Toc372756232)

[8.4.5 Caso de uso – Visualizar Pontuações: 22](#_Toc372756233)

[9 Diagrama de Atividades 23](#_Toc372756234)

[10 MER - Modelo Entidade-Relacionamento 25](#_Toc372756235)

[11 DER - Diagrama de Entidade e Relacionamento 25](#_Toc372756236)

[12 Base 26](#_Toc372756237)

[12.1 Tabela Periódica 27](#_Toc372756238)

[12.2 Simulador de Distribuição Eletrônica 29](#_Toc372756239)

[12.2.1 Resumo sobre Distribuição Eletrônica 29](#_Toc372756240)

[12.2.2 Telas Simulador de Distribuição Eletrônica: 32](#_Toc372756241)

[12.2.3 Descrição 33](#_Toc372756242)

[12.2.4 Funcionalidade 33](#_Toc372756243)

[12.3 Simulador de Cálculos Estequiométricos 35](#_Toc372756244)

[12.3.1 Resumo sobre Cálculo Estequiométrico 35](#_Toc372756245)

[12.4 Simulador de Ligação Iônica 37](#_Toc372756246)

[12.4.1 Descrição 37](#_Toc372756247)

[12.5 Simulador de Solubilidade 39](#_Toc372756248)

[12.5.1 Resumo sobre Solubilidade 39](#_Toc372756249)

[Referências Bibliográficas 42](#_Toc372756250)

[Referências de Imagens 43](#_Toc372756251)

[Anexo A – Código-fonte 44](#_Toc372756252)

**Índice de Figuras**

[Figura 1 - Diagrama de Caso de Uso 15](#_Toc360374046)

[Figura 2 - Diagrama de Atividades Tabela Periódica 21](#_Toc360374047)

[Figura 3 - Diagrama de Atividades Distribuição Eletrônica 22](#_Toc360374048)

[Figura 4 - MER: Modelo Entidade-Relacionamento 23](file:///C:\Users\GUSTAVO\Desktop\Monografia%202.0%20final%20alterada%20para%20impressão.docx#_Toc360374049)

[Figura 5 - DER: Diagrama de Entidade e Relacionamento 23](#_Toc360374050)

[Figura 6 - Tabela Periódica 25](#_Toc360374051)

[Figura 7 - Tabela Periódica - Detalhes do Elemento 26](#_Toc360374052)

[Figura 8 - Ilustração de um átomo dividido em camadas 27](file:///C:\Users\GUSTAVO\Desktop\Monografia%202.0%20final%20alterada%20para%20impressão.docx#_Toc360374053)

[Figura 9 - Diagrama de Linus Pauling 29](file:///C:\Users\GUSTAVO\Desktop\Monografia%202.0%20final%20alterada%20para%20impressão.docx#_Toc360374054)

[Figura 10 - Tela de Distribuição Eletrônica 30](#_Toc360374055)

[Figura 11 - Tela de Distribuição Eletrônica com elemento selecionado 30](#_Toc360374056)

[Figura 12 - Tela Cálculos Estequiométricos 35](#_Toc360374057)

[Figura 13 - Tela Ligação Iônica 36](#_Toc360374058)

[Figura 14 - Tela Solubilidade 38](#_Toc360374059)

# 1 Introdução

Pesquisas realizadas pelo grupo apontam um aproveitamento baixo dos alunos na disciplina de química, cujo o rendimento dos alunos nessa disciplina é indispensável para um bom aproveitamento em provas de vestibulares ou até mesmo em avaliações da instituição.

Devido à esse problema, o projeto foi realizado com o intuito de ser uma ferramenta a mais para o professor. O software busca facilitar os estudos dos alunos e o trabalho do corpo docente, com aulas mais dinâmicas e interativas, trazendo melhores resultados no aproveitamento das aulas e nas menções dos alunos.

# 2 Objetivo geral

Facilitar o estudo e o lecionamento da matéria de química.

## 2.1 Objetivos específicos

* Desenvolver sistema;
* Implantar sistema na instituição;
* Testar sistema;
* Apresentar;
* Avaliar o aprendizado dos alunos.

# 3 Metodologia

Com uma metodologia ágil e de pesquisa interna, o desenvolvimento do projeto fez-se de modo flexível, em que as tarefas foram divididas em etapas. O raciocínio elaborado para a produção do dinamismo foi feito a partir de testes, análises e opiniões diante de professores e alunos de química. Através do recolhimento de dados do conselho realizado dia 24 de abril de 2013 detalhando as menções parciais dos alunos do 1º ano da ETEC da Zona Leste, foi observado que os alunos do ensino médio não possuem um bom rendimento quanto a matéria de química. As informações obtidas foram as seguintes:

|  |
| --- |
|  |
|  | **1º MA** | **1º MB** | **1º IIM** | **1º IAM** |
|  | *40 Alunos (2 desistentes)* | *40 Alunos* | *38 Alunos* | *40 Alunos* |
| **MB** | 1 | 0 | 10 | 1 |
| **B** | 20 | 18 | 14 | 18 |
| **R** | 15 | 21 | 14 | 21 |
| **I** | 2 | 0 | 0 | 0 |

Percebe-se que, de um total de 158 alunos, temos apenas 12 com menção MB, que atingiram o nível de aprendizado muito bom na matéria lecionada nesse período; 70 menções B (Bom), que atingiram um bom nível de aprendizado. Já num nível fora do esperado pelo professor, temos 71 menções R (Regular) e apenas 2 menções I (Insatisfatório). À partir disso observa-se que mesmo com o número baixo de menções I, o número de menções R é muito grande, ou seja, a dificuldade que os alunos apresentam é muito acentuada.

Gerou-se, portanto, uma certeza de que o mais adequado às típicas necessidades de uma classe de ensino médio estava sendo produzido, o software que traria, na prática, essa matéria que possui conceitos tão complexos e um tanto abstratos.

Essa produção fez-se a partir, principalmente, do compartilhamento de distintas soluções para um mesmo problema, em que os padrões químicos como camadas de eletrização, regras do octeto e curva de solubilidade foram observados, analisados e explorados por diferentes perspectivas para que, por fim, utilizando materiais simples, como computadores pessoais e livros didáticos (citados nas referências), todas as ideias e aspectos antes discutidos fossem transformados em programação e interface virtual.

# 4 Hipóteses

ChemQuark, por ser um aplicativo inovador, possui muitas características que não podem ser encontradas facilmente em outros aplicativos. No entanto, geram-se hipóteses à favor e contra a implantação do software na ETEC da Zona Leste, em relação ao aproveitamento de estudos pelos alunos.

Implantando o sistema ChemQuark na Etec Zona Leste, haverá melhor aproveitamento nos estudos pelos alunos da Unidade?

Implantando o sistema ChemQuark na Etec Zona Leste, não haverá melhor aproveitamento nos estudos pelos alunos da Unidade?

# 5 Introdução ao assunto

## 5.1 Informática

Informática é o estudo da informação. Nesse grupo de informação são aplicados os assuntos da ciência da computação, a teoria da informação, o processo de cálculo, a análise numérica e os métodos teóricos da representação dos conhecimentos e de modelagem dos problemas.

## 5.2 Software

Software é toda a parte lógica de um sistema. É uma sequência de instruções escritas para serem interpretadas por um computador com o objetivo de executar tarefas específicas.

Os softwares incluem aplicações informáticas como, por exemplo, um processador de textos, que permite ao usuário da aplicação realizar tarefas relacionadas com a edição de textos.

Existem três tipos de classificação:

**Software de Sistema**: é o conjunto de informações processadas pelo sistema interno de um computador que permite a interação entre usuário e os periféricos do computador através de uma interface gráfica. Engloba o sistema operativo e os controladores de dispositivos (memória, impressora, teclado e outros).

Software de Programação: é o conjunto de ferramentas que permitem ao programador desenvolver sistemas informáticos, geralmente usando linguagens de programação e um ambiente visual de desenvolvimento integrado.

Software de Aplicação: são programas de computadores que permitem ao usuário executar uma série de tarefas específicas em diversas áreas de atividade como arquitetura, contabilidade, educação, medicina e outras áreas comerciais. São ainda os videojogos, as bases de dados, os sistemas de automação industrial, etc.

# 6 Instituição

Escola Técnica da Zona Leste

**Localização e Contato**

Av. Águia de Haia, 2633 - Cidade A.E. Carvalho

CEP 03685-000 - São Paulo/SP

Telefone: (11) 2045-4016 / (11) 2045-4002 / (11) 2045-4003

E-mail: dir.etezonaleste@centropaulasouza.sp.gov.br

e.zonaleste.adm@centropaulasouza.sp.gov.br

## 6.1 Descrição

A Escola Técnica da Zona Leste é uma instituição pública de ensino médio e técnico, localizada na cidade de São Paulo, no bairro Cidade Antônio Estevão de Carvalho (Cidade A.E. Carvalho) e mantida pelo Centro Estadual de Educação Tecnológica Paula Souza (CEETEPS), que é vinculado à Secretaria do Desenvolvimento do Governo do Estado de São Paulo.

O Centro Tecnológico da Zona Leste, constituído pela ETEC da Zona Leste e FATEC da Zona leste, foi criado em fevereiro de 2002 e extinto em agosto de 2009, passando, a partir desta data, a existirem separadamente uma da outra. O parceiro do presente software, que será a escola técnica, oferece vários cursos que se dividem no turno da manhã, tarde e da noite.

A presença de um professor em sala de aula é indispensável, entretanto, algumas vezes, os alunos, por não entenderem o conceito inicial com a ajuda do mesmo, deixam tal matéria de lado e acabam não aprendendo. O que fará uma grande falta quando estes forem prestar o vestibular.

Portanto, uma ferramenta que ajude o professor a ensinar, e simultaneamente aos alunos a aprenderem é de grande importância.

## 6.2 Estrutura

Por ser uma escola técnica que inclui cursos que utilizam a tecnologia, existem diversos lugares para que o sistema possa ser implantado:

- 11 Laboratórios com cerca de 20 computadores em cada.

- Biblioteca (Computadores que todos os alunos da escola podem acessá-los no horário de funcionamento da biblioteca).

Inclusive, a maioria dos computadores possuem hardware excelente; com sistemas operacionais atuais (Windows 7 / Windows XP), pode-se notar isto, ao ver os atuais softwares que estes suportam, como: NetBeans, SQL Server, Microsoft Office e Embarcadero.

Os tipos de hardwares são completamente semelhantes entre si; seja pela capacidade de processamento quanto armazenamento.

Todos estes computadores, mesmo possuindo algumas diferenças entre si, estão ligados a uma rede em um servidor, que permite acesso à internet.

Por ter esse acesso a internet, há duas possíveis ligações de banco de dados, sendo ambas bastante difíceis de serem implantadas. São elas: Virtual e Local.

Na virtual, o banco de dados de cadastro dos alunos com suas devidas pontuações e todas suas ações no aplicativo estariam em um servidor externo, que seria acessado via internet. Quanto à local, o banco de dados estaria alocado no servidor da própria ETEC, o que seria mais compreensível, já que o acesso seria mais instantâneo do que pela internet.

Vendo este problema, um estudo mais detalhado foi feito, e observou-se que fazer a instalação em todas as máquinas é completamente desnecessário. Considerando que o ensino médio possui apenas um professor de Química, um ou dois laboratórios em específico é o suficiente para atender à necessidade dos alunos e do professor. Principalmente pelo uso dos laboratórios durante o período da tarde não ser muito constante, que ocasiona uma maior disponibilidade dos laboratórios de química, que para serem usados, devem ser reservados anteriormente. Ou seja:

- Se um professor de química reservar um laboratório, será disponibilizado aquele que possui o software.

- Caso um professor que não é de química reserve um laboratório, será disponibilizado aquele que não possui o software de química.

## 6.3 Instrutor da empresa

A pessoa que ajudou na orientação e na produção do software para um funcionamento correto quanto às regras da química foi o professor da disciplina já citada do ensino médio, o professor Erick Figueroa, bacharelado em química com conhecimentos profundos em diversos aspectos da mesma, sejam eles teóricos ou práticos. O mesmo, por conviver diariamente com o grupo, ajudou sempre que houve necessidade do grupo contatá-lo, seja para cálculos, regras químicas ou para ideias e opiniões que poderiam ser incluídas no software, tornando o aplicativo realmente educativo.

**Conclusão**

Por ser uma escola com estrutura técnica, a implantação do software educativo de química é facilitado ao extremo, exigindo somente modificações internas e não externas aos computadores. A única dificuldade realmente encontrada é a que cerca o banco de dados, afetado pelo aplicativo "DeepFreeze"; pela internet com pouca velocidade e pelas limitações de alterações no servidor da ETEC.

# 7 MARKUP

OMarkup  é um termo usado em Economia para indicar quanto do preço do produto está acima do seu custo de produção e distribuição. Significa diferença entre o custo de um bem ou serviço e seu preço de venda. O valor representa a quantia efetivamente cobrada sobre o produto a fim de obter o preço de venda.

**CONCEITO**

**Custo de Produto Vendido 🡪**Valor Pago ao Comprador + Impostos + Frete + Encargos Financeiros, (não incluso descontos e bonificações).

**Margem de Lucro 🡪** Razão entre o acréscimo sobre o custo do produto e o preço de venda desse produto.

**Fórmula:**

**Margem de Lucro = (Preço de Venda - Preço de Custo) / Preço de Venda.**

**Markup  🡪 Índice de Marcação de Preços:**é um conceito parecido com a Margem de Lucro, porém a sua aplicação é um pouco diferente, pois o objetivo é estabelecer o preço de venda a partir do seu custo de aquisição.

**Fórmula:**

**MarkUp = (Preço de Venda - Preço de Custo) / Preço de Custo.**

**Preço de Venda - Preço de Custo x (1 + MarkUp).**

**Diferenças**:

Margem de Lucro: Cálculo com base no Preço de Venda e Custo de Compra.

MarkUp: Usado para chegar ao Preço de Venda, baseado no Custo de Aquisição.

**CÁLCULO DO SOFTWARE CHEMQUARK**

Delphi e Access = **R$ 0,00** (Versões gratuitas para estudantes)

Condução = 10 pessoas x **R$ 6,00** X 22 dias = **R$ 1.320,00**

Alimentação = 10 pessoas x **R$ 5,00** X 22 dias = **R$ 1.100,00**

Programadores Delphi x 22 dias = **R$ 5.940,00**

2 Marketing = **R$ 2.000,00**

1 Analista = **R$ 2.200,00**

1 DBA (Administrador de Banco de Dados) = **R$ 1.200,00**

1 Designer = **R$ 1.200,00**

Gastos Diversos = **R$ 250,00**

Total de Custo = **R$ 15.210**

Total de Escolas em São Paulo = **3500**

**Margem de Lucro:** R$ 0,40 a cada R$ 1,00 investido

**Lucro por cada Software vendido:** R$ 64,00

**Deverá ser vendido para pagar todo investimento inicial:** 96 Cópias

**Preço do Software:** R$ 160,00

10% das Escolas = 350 / 21.378,00 = 15.210

15.210 + 6.084 / 200

Total de Escolas em SP = 200

# 8 Diagrama de Caso de Uso

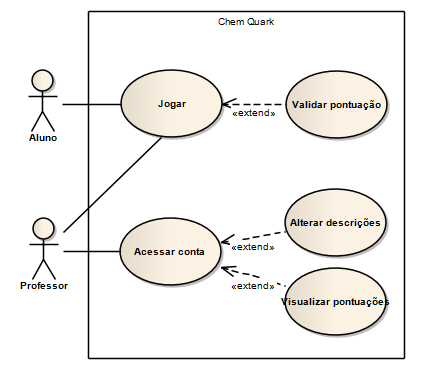


Figura 1 - Diagrama de Caso de Uso

## 

## 8.1 Descrição dos Atores

**Aluno –** Possui apenas a opção de jogar e ser avaliado.

**Professor** – Possui como função alterar dados, inserir novos cálculos estequiométricos, visualizar pontuações e liberar o modo avaliação; para isso deve acessar sua conta administradora.

## 8.2 Descrição dos Casos de Uso

### 8.2.1 Caso de uso – Jogar:

**Descrição Sucinta**

Conjunto de módulos e funções que permite toda a dinâmica do usuário com o jogo, fornecendo-o opções de dificuldade e instruções.

**Atores**

* Aluno
* Professor

**Fluxo Básico:** Modo livre

1. O aluno escolhe o modo de jogo desejado;
2. O aluno segue as instruções para inicialização do jogo;
3. O aluno exerce todas as funções necessárias para completar o jogo.

**Fluxo alternativo:** Modo avaliação

* 1. Para escolher o modo avaliação, o professor deve liberar o acesso.

**Regras de Negócio**

*(RN1) -* O Aluno deve responder todas as questões e passar por todas as etapas do jogo escolhido.

## 

### 8.2.2 Caso de uso - Validar Pontuação:

**Descrição Sucinta**

Conjunto de funções que permite o usuário visualizar, compartilhar e validar sua pontuação.

**Ator**

* Aluno

**Pré-condições**

O modo escolhido deve ser o modo avaliação, caso contrário, não há pontuação válida para avaliação.

**Fluxo Básico**

1. O Aluno visualiza sua pontuação final, tendo uma média de acertos e erros;
2. O Aluno insere seus dados e confirma sua pontuação.

**Regras de Negócio**

* *(RN1)* Os dados inseridos devem ser válidos para que não haja problemas na utilização dos mesmos.
* *(RN2)* O Aluno deve validar sua pontuação, caso contrário, a mesma não será utilizada para sua avaliação.

## 

### 8.2.3 Caso de uso – Acessar Conta:

**Descrição Sucinta**

Fazer login para visualizar todas as pontuações cadastradas e se necessário, alterar informações sobre algum elemento químico ou inserção de novos cálculos estequiométricos.

**Ator**

* Professor

**Fluxo Básico**

1. O Professor insere seu usuário e senha;
2. O Professor valida o usuário e a senha e escolhe qual atividade deseja realizar.

**Fluxo Alternativo:** Login e/ou senha incorretos

*2.1 -* O Professor digita novamente o usuário e senha.

**Regras de Negócio**

*(RN1)* Os dados do professor devem estar cadastrados para que o mesmo efetue o login.

## 

### 8.2.4 Caso de uso – Alterar Descrições:

**Descrição Sucinta**

Um conjunto de opções que permite a alteração dos dados de um ou mais elementos químicos ou inserção de novos cálculos estequiométricos.

**Ator**

* Professor

**Pré-Condições**

O Professor deve fazer o login com seu usuário e senha para ter acesso a esse recurso.

**Fluxo Básico**

1. O Professor acessa aos dados do(s) elemento(s) químicos(s) desejado(s) ou à tela de inserção de novos cálculos estequiométricos;
2. O Professor altera ou insere os dados necessários;
3. O Professor salva as alterações.

### 8.4.5 Caso de uso – Visualizar Pontuações:

**Descrição Sucinta**

Visualização das pontuações cadastradas ao efetuar os jogos.

**Ator**

* Professor

**Pré-Condições**

O Professor deve fazer o login com seu usuário e senha para ter acesso a esse recurso.

**Fluxo Básico**

1. O Professor visualiza as pontuações cadastradas pelos alunos ao executar os jogos.

**Regras de Negócio**

*(RN1)* O Professor deve estar acessando sua conta para acessar esse recurso.

*(RN2)* Os dados obtidos devem ser utilizados apenas para questões de avaliação.

# 9 Diagrama de Atividades

Ambos os diagramas a seguir são processos do caso de uso: ***Jogar***

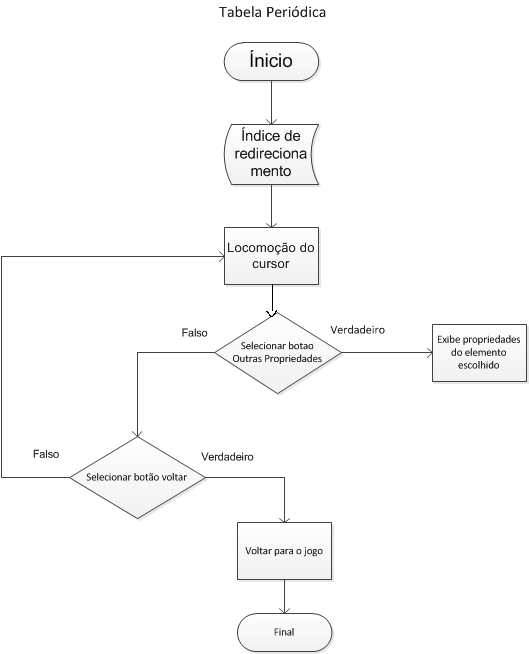


Figura 2 - Diagrama de Atividades Tabela Periódica

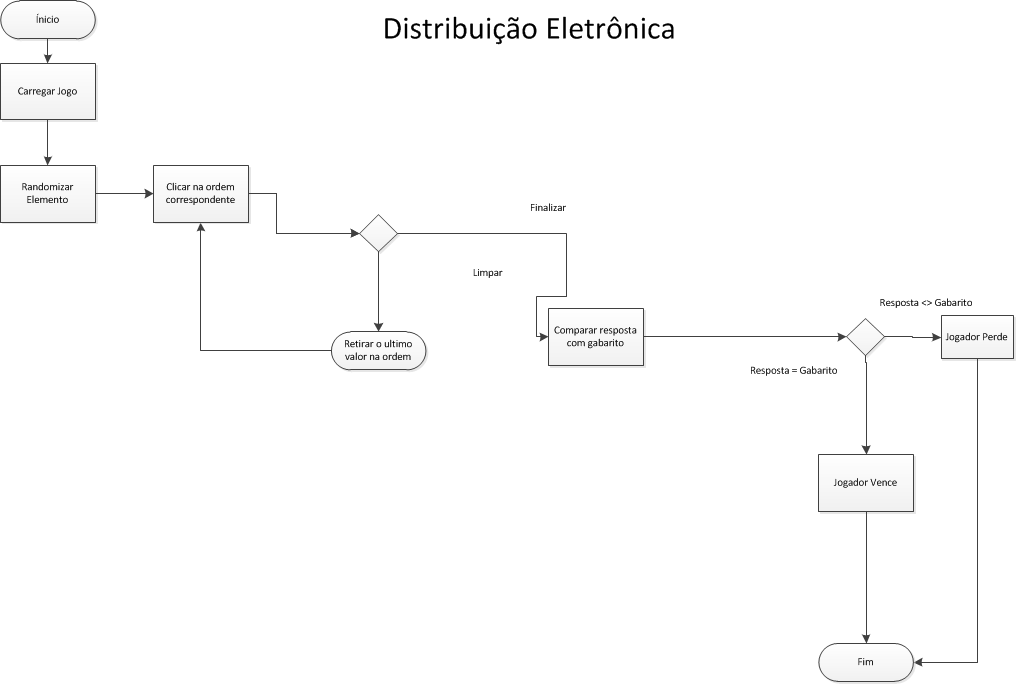


Figura 3 - Diagrama de Atividades Distribuição Eletrônica

# 10 MER - Modelo Entidade-Relacionamento

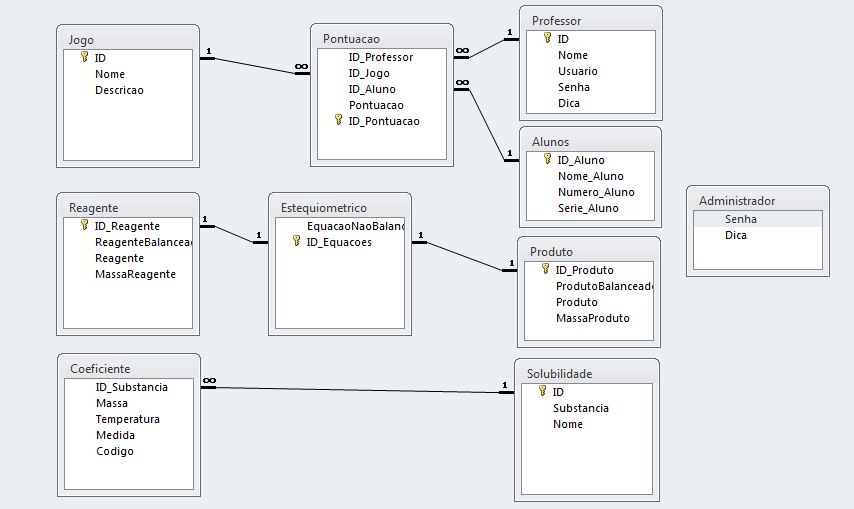


Figura 4 - MER: Modelo Entidade-Relacionamento

# 11 DER - Diagrama de Entidade e Relacionamento



Figura 5 - DER: Diagrama de Entidade e Relacionamento

# 12 Base

A base do software foca-se em simuladores educativos e interativos quanto a alguns assuntos específicos da química, são eles:

* Tabela Periódica;
* Distribuição Eletrônica;
* Ligação Iônica;
* Solubilidade;
* Cálculo Estequiométrico.

Por serem dinâmicos e intuitivos, facilitarão o aprendizado e simultaneamente, haverá um melhor aproveitamento das aulas.

## 12.1 Tabela Periódica

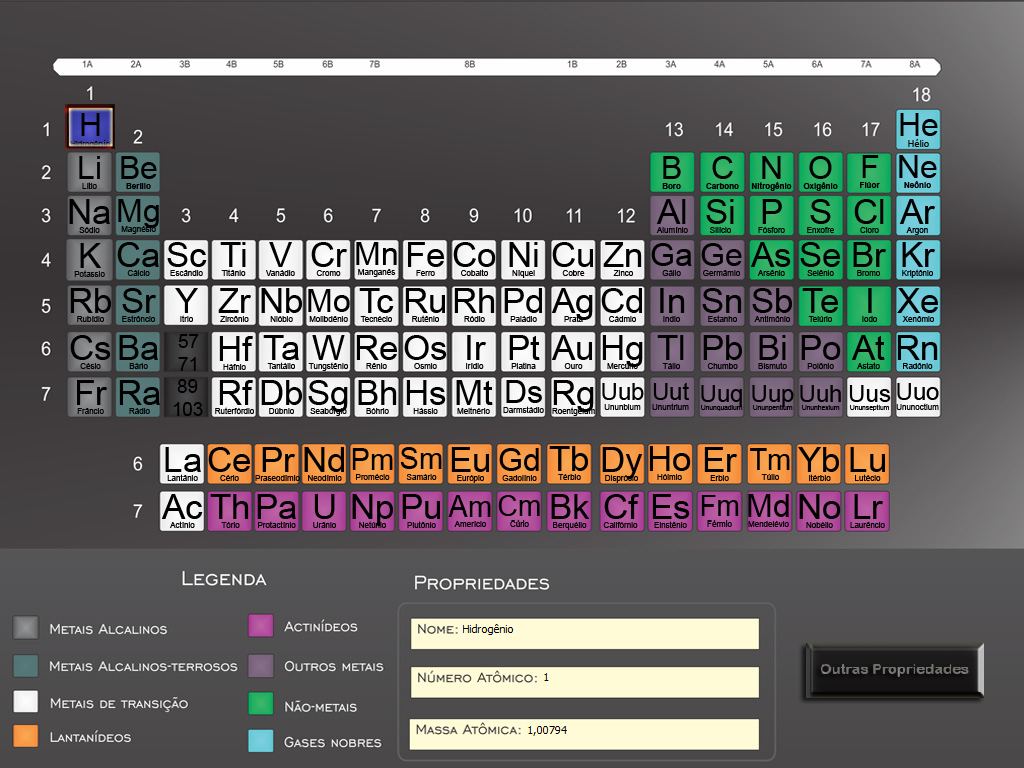


Figura 6 - Tabela Periódica

A tabela periódica pode ser acessada a qualquer momento durante o uso do aplicativo, nela contém como visualização primária as informações básicas que contém em todas as cópias de tabela periódica (Nome, Número Atômico, Massa Atômica e Símbolo). Entretanto ao clicar em um determinado elemento, abre-se uma nova janela onde contem informações mais detalhadas sobre o elemento, como por exemplo: Onde é usado; Onde é encontrado; e Atributos específicos.

A tabela periódica contém somente a exibição por meio de design de um Array (Conjunto de valores do mesmo tipo) de elementos, ou seja, o array de elementos levará 118 índices, onde cada um conterá um tipo diferente de objeto de uma mesma classe.

Esses objetos conterão atributos como:

* Número atômico;
* Massa atômica;
* Nome científico;
* Letra molecular;
* Grupo;
* Período;
* Descrição.

Que são utilizados constantemente em diversos cálculos dos jogos.

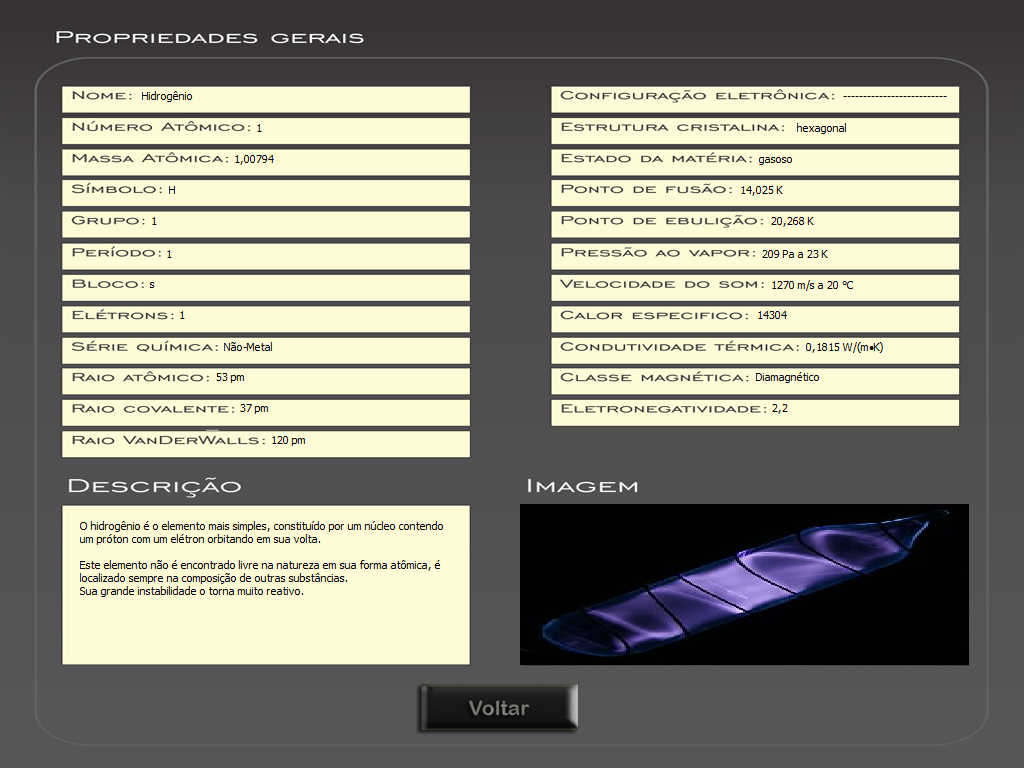
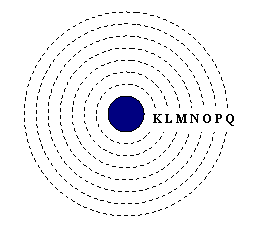


Figura 7 - Tabela Periódica - Detalhes do Elemento

## 12.2 Simulador de Distribuição Eletrônica

### 12.2.1 Resumo sobre Distribuição Eletrônica

A seguir, temos um tutorial que irá ajudá-lo a entender o que é e como fazer a distribuição eletrônica para que você tenha sua base para o término do jogo, e para que ocorra uma aprendizagem da sua parte. Distribuição Eletrônica, o que é?

Os elétrons giram em torno do núcleo em camadas pré-definidas, e utilizamos a Distribuição Eletrônica para que possamos descobrir o número de Elétrons que cada camada possui, e também o número de elétrons que o átomo possui na última camada ou o que chamamos de Camada de Valência.

A partir de onde calculamos?

Um átomo é representado da seguinte forma:

Figura 8 - Ilustração de um átomo dividido em camadas

* Onde A é o Elemento;
* X é o Número de Massa;
* Y é o Número Atômico.

Em um átomo, o número de prótons sempre se iguala ao número de elétrons onde neste caso é representado pelo Y.

Agora vamos analisar um elemento em questão; o Hélio:

He = Hélio;

4 = Número de Massa;

2 = Número Atômico = Prótons = Elétrons.

Então agora sabemos que devemos calcular a partir do valor 2.

#### Linus Carl Pauling

Nascido em 1921 nos Estados Unidos, Linus Pauling, formou-se em Engenharia Química, doutorou-se no Instituto de Tecnologia da Califórnia e estagiou em várias universidades da Europa. Em 1927, tornou-se professor do Instituto de Tecnologia da Califórnia e, em 1968, da Universidade de Stanford.

Seus trabalhos mais importantes versam sobre a estrutura atômica, a natureza das ligações químicas e a estrutura das proteínas. Divulgou o uso da vitamina C no combate ao resfriado e a certos tipos de câncer. Foi um ardoroso pacifista. Recebeu o Prêmio Nobel de Química em 1945 e o Prêmio Nobel da Paz em 1962. Faleceu em 1994 nos Estados Unidos. O cientista Linus Pauling imaginou um diagrama para realizar a distribuição eletrônica dos elétrons. Esse diagrama ficou conhecido como Diagrama de Linus Pauling.

**A Base para o Cálculo**

Cada camada comporta um número máximo de elétrons como segue abaixo

K 2

L 8

M 18

N 32

O 32

P 18

Q 8

Linus Pauling apresentou as subcamadas que também tem o seu número máximo de elétrons como abaixo:

|  |  |  |
| --- | --- | --- |
| Subnível | Num. Máximo de Elétrons | Nomenclatura |
| s | 2 | s2 |
| p | 6 | p6 |
| d | 10 | d10 |
| f | 14 | f14 |

Depois de apresentar estes dois conceitos básicos, Linus os uniu para que pudesse fazer uma forma simples e fácil de descobrir estes elétrons na camada. Esta foi a distribuição eletrônica a partir deste diagrama:

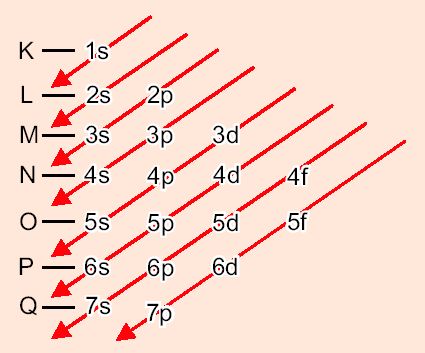


Figura 9 - Diagrama de Linus Pauling

Onde utilizamos a seguinte sequência:

1s2, 2s2, 2p6, 3s2, 3p6, 4s2, 3d10, 4p6, 5s2, 4d10, 5p6, 6s2, 4f14, 5d10, 6p6, 7s2, 5f17, 6d10, 7p6.

### 12.2.2 Telas Simulador de Distribuição Eletrônica:

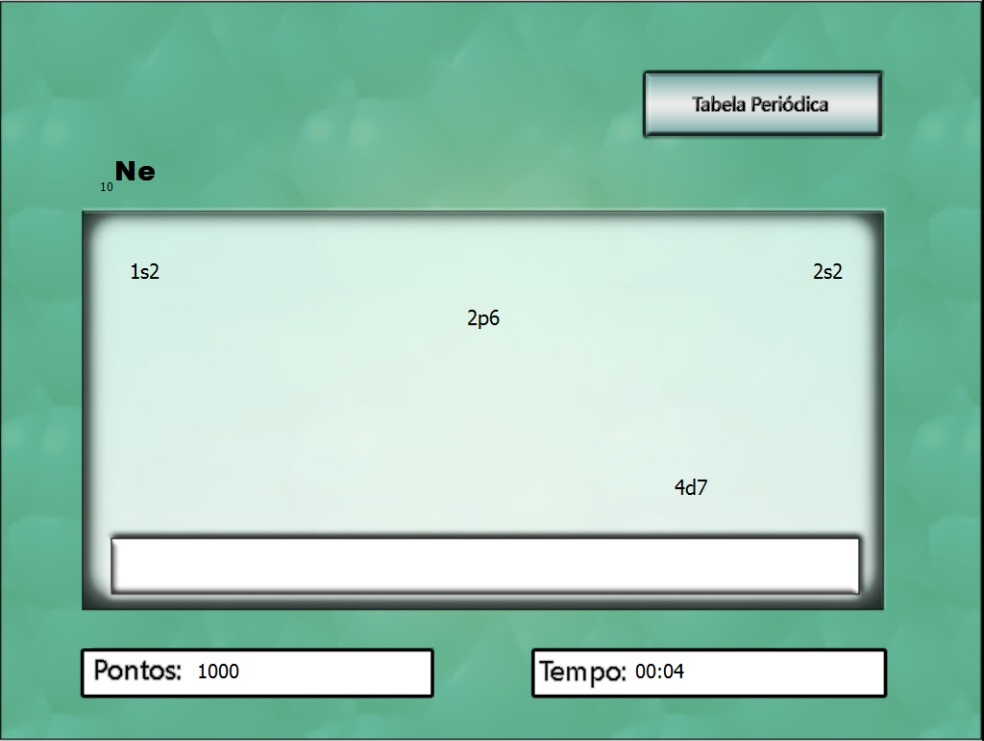


Figura 10 - Tela de Distribuição Eletrônica

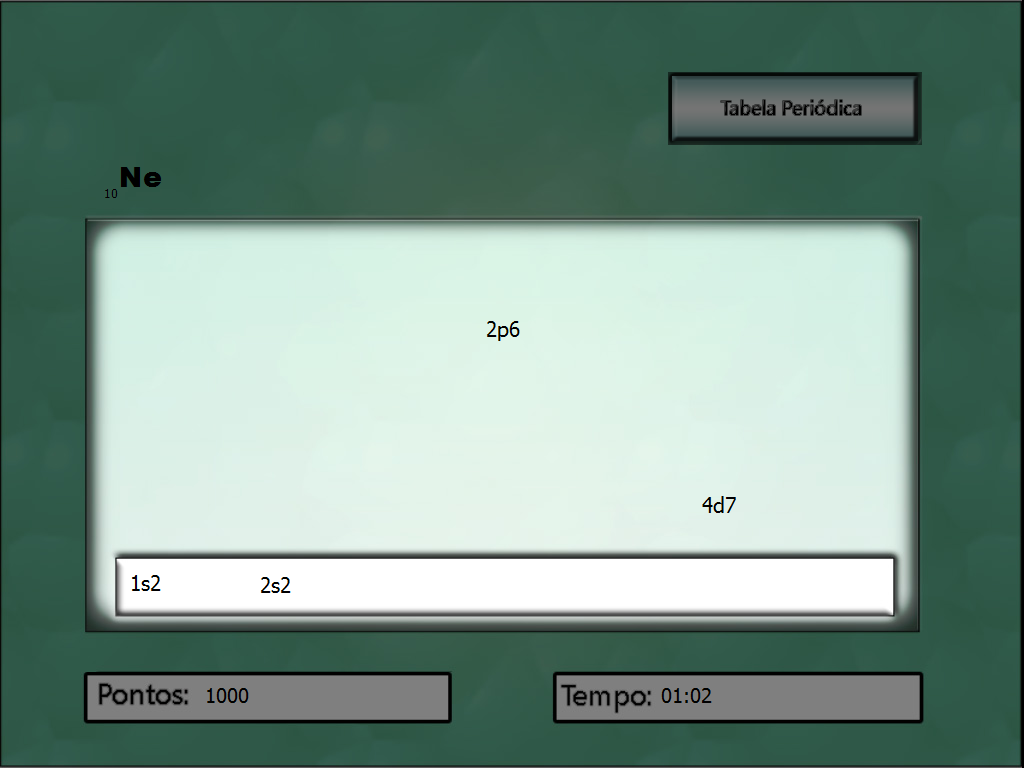


Figura 11 - Tela de Distribuição Eletrônica com elemento selecionado

### 12.2.3 Descrição

Neste jogo, o único e principal objetivo do jogador será fazer a correta distribuição eletrônica de acordo com o **número atômico** do elemento da rodada. A interação do usuário com o aplicativo limitar-se-á somente ao controle dos níveis e subníveis (Exemplo: 1S2; 2S2; 2P6). Ao clicar em cima de tal objeto, a possibilidade de locomoção do objeto pelo mouse é gerada, o objetivo do jogador será colocá-los em uma ordem correta em uma espécie de receptáculo abaixo, a ordem dos objetos ficará de acordo com a ação do usuário, "Primeiro a entrar, último a sair". Caso o usuário queira remover tal objeto, basta movê-lo de volta ao quadro de subníveis.

O software avisará o jogador automaticamente caso ele ganhe, ou seja, quando todos os devidos objetos (níveis e subníveis. Exemplo: "1s1") estiverem em suas posições corretas, o usuário vence. Entretanto, para controle de pontuação, a cada vez que o jogador remover um objeto da tabela de resposta, pontos (dependentes do nível de dificuldade) são subtraídos da pontuação final. O tempo dado ao jogador é feito a partir de uma fórmula:

***Número Atômico \* 10.***

O tempo diminui a cada segundo, a pontuação final é feita de acordo com a fórmula:

***Tempo restante \* Pontos***

Se o tempo acabar antes que o jogador conclua o objetivo, um aviso será dado, a fim de que ele escolha se jogará novamente ou optará por jogar outro simulador.

### 12.2.4 Funcionalidade

As funções e a lógica que farão o jogo funcionar corretamente começam pela determinação do elemento a ser jogado. Como já dito: cada elemento será um objeto com alguns principais atributos, entre eles, o **número atômico.** Portanto, para a definição do elemento, uma função de números aleatórios será chamada, e via uma interação no array que contém todos os elementos (elementos moleculares) do jogo, será verificado à qual elemento o número atômico é igual ao número sorteado (1 a 118).

O elemento que estiver dentro desta condição será o selecionado para aquela rodada. Assim, já tendo o número atômico em posse, verifica-se agora qual a resposta correta da distribuição, para que se possam colocar níveis e subníveis corretos no quadro de seleção; usaremos a lógica a seguir:

**Início**

Se (Número atômico - 2 > 0) então

Número atômico -= 2;

Adiciona o texto "1s²" a um vetor.

Se (Número atômico - 2 > 0) então

Número atômico -= 2;

Adiciona texto "2s²" a um vetor;

Se (Número atômico - 6 > 0) então

Número atômico -= 2;

Adiciona texto "2P6" a um vetor;

Se não

Adiciona texto "2P" + Número atômico a um vetor;

Fim Se;

Se não

Adiciona texto "2S" + Número atômico a um vetor;

Fim Se;

Senão

Adiciona no painel de seleção um rótulo com evento clicável com o texto: "1S" + Número atômico;

Fim Se;

Adiciona todos os valores textos de tal vetor como rótulos com evento clicável no painel de seleção;

**Fim.**

Lembrando que isto é só o início da estrutura, o restante continua uniformemente ao mostrado acima.

Com todas as opções já no painel de seleções e em um vetor, agora a programação da interatividade do jogador com o software funcionará da seguinte maneira:

Quando o jogador clicar no *label (rótulo)*, este *label* muda sua posição para a posição livre mais próxima à esquerda no painel de resposta. O que será controlado via variável inteira, que levará como valor a quantidade de objetos que já estão no painel de resposta. Simultaneamente a isto, as opções que forem escolhidas pelo jogador, serão adicionadas a um vetor, que no fim será verificado se é idêntico ao vetor já mostrado na decisão encadeada anterior, se for igual, o jogo termina.

A cada segundo, uma variável que leva o tempo é subtraída por 1, e atualiza o *label* que exibe o tempo. A pontuação começa equivalente ao número atômico, caso esta pontuação chegue ao 0, pois o jogador retirou a quantidade máxima de vezes uma opção do painel de resposta, o jogo será finalizado e as opções de jogar novamente e escolher outro jogo aparecerão para o jogador.

## 

## 12.3 Simulador de Cálculos Estequiométricos

### 12.3.1 Resumo sobre Cálculo Estequiométrico

A estequiometria estuda as relações entre as quantidades de reagentes e dos produtos de uma reação química. Estas relações podem ser feitas em mols, massas, volumes, número de moléculas, etc. Para realizar estes cálculos deve-se:

1º- Escrever a equação da reação química.

2º- Balancear os coeficientes estequiométricos da equação.

3º - Efetuar uma regra de três entre as grandezas envolvidas, obedecendo aos coeficientes da equação, que indicam a proporção entre o número de mols.

Uma determinada quantidade de produto, ou parte dele, é determinada por uma quantidade específica de reagente:

2 H_2 + O_2 \longrightarrow 2 H_2O

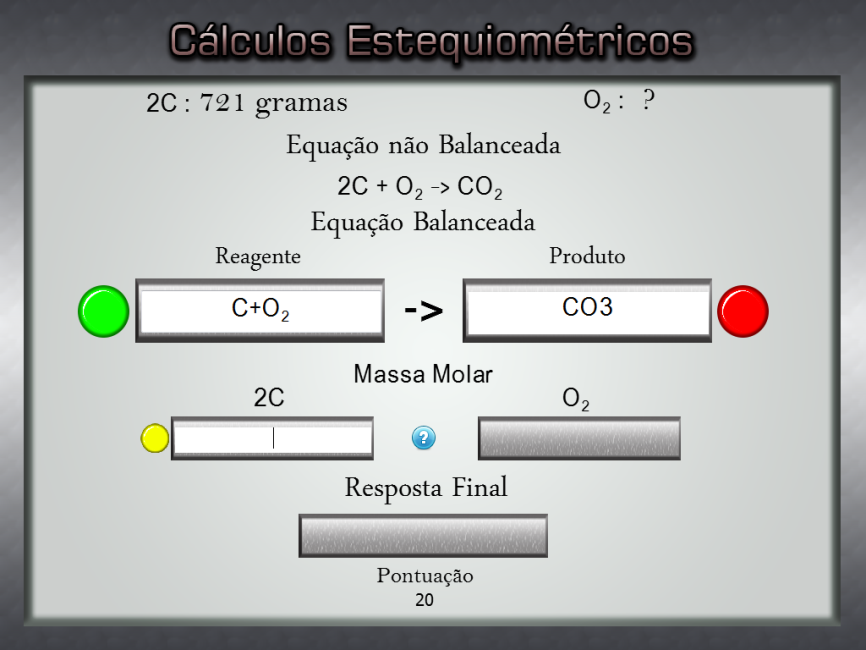


Figura 12 - Tela Cálculos Estequiométricos

No simulador, portanto, o usuário auxiliará o programa a efetuar uma correta solução, digitanto os dados, etapa por etapa:

* Reagente Balanceado;
* Produto Balanceado;
* Massa Molar do Reagente;
* Massa Molar do Produto;
* Resposta final;

O simulador analisa os dados das áreas de texto e verifica a resposta, o resultado é indicado pela cor do circulo ao lado:

* Verde: Resposta correta;
* Amarelo: Aguardando resposta;
* Vermelho: Resposta incorreta.

## 12.4 Simulador de Ligação Iônica

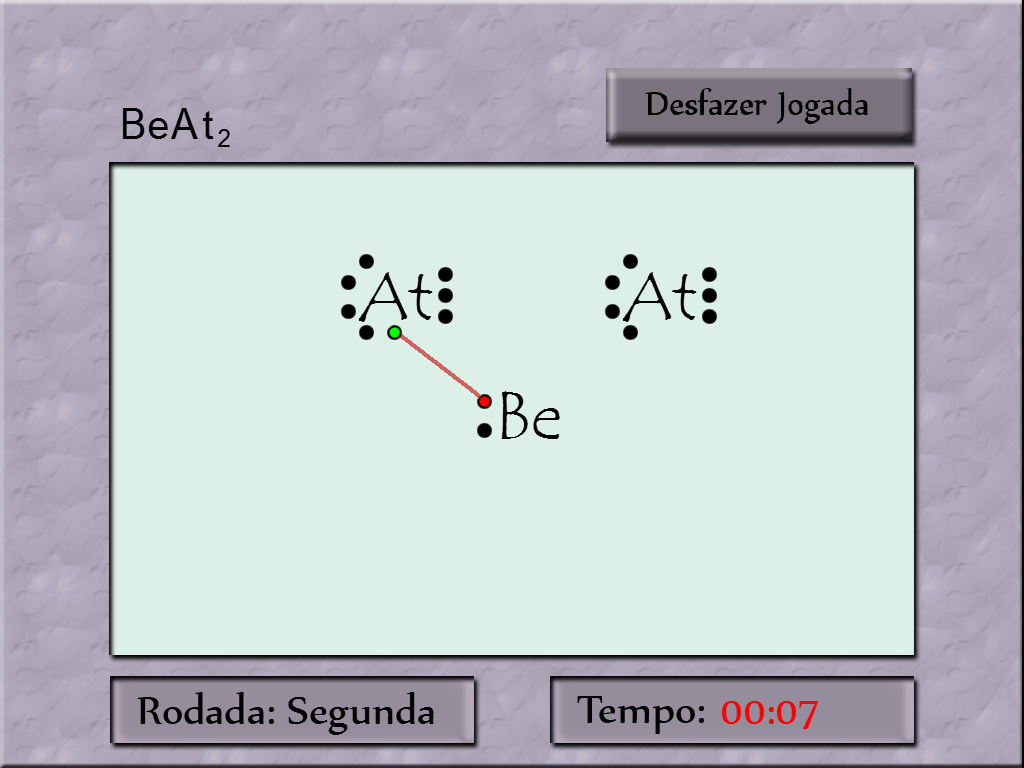


Figura 13 - Tela Ligação Iônica

### 12.4.1 Descrição

Neste jogo, os objetivos são dois, a primeira etapa, é fazer a ligação dos elétrons de cada elemento. Cada elemento tem como padrão, uma quantidade de elétrons que é definida pelo seu grupo na tabela periódica.

Como já dito, a quantidade de elétrons é algo que será inserido no aplicativo previamente para a possibilidade de envolvê-los nos cálculos. Cada elemento, com exceção de alguns, como o Hidrogênio, necessitam de 8 elétrons em sua camada para que possam ser declarados nobres, portanto, as ligações covalentes tem como princípio, a doação de elétrons de um elemento para com outro elemento.

Portanto, para a criação da molécula para a resolução do problema pelo jogador, serão necessários alguns cálculos:

- Fazer uma pré-seleção dos elementos não-metais a partir dos grupos e períodos (atributos da classe Elementos).

- Colocar todos esses elementos numa array e chamar a função de números aleatórios uma quantidade aleatória de vezes. Ao fazer isto, 2 ou mais elementos serão obtidos, entretanto, isto não significa que eles serão realmente os determinados, já que será necessário o cálculo da quantidade de elétrons. Cada elemento tem que ter a quantidade certa há mais ou a menos do que o outro elemento. Por exemplo:

* Um elemento A possui 6 elétrons.
* Um elemento B possui 5 elétrons.

Portanto, A doará 2 elétrons para B; e B doará 2 elétrons para A. Ficando assim:

* A com 8 elétrons
* B com 7 elétrons

Portanto, B necessita de mais um elétron, logo, busca-se na pré-seleção de elementos não-metais, um elemento que necessite de 1 elétron para ser nobre. Como exemplo, o Flúor, que foi nomeado como C.

- C com 7 elétrons

Logo, B doa 1 elétron para C; e C doa 1 elétron para B.

Logo, todos os 3 elementos, A; B e C, possuem 8 elétrons na camada de valência, evidenciando-se como nobres.

Com os elementos já escolhidos, eles são inseridos no painel de resultado, onde haverá a interatividade do jogador com o aplicativo.

Esta interatividade funcionará como a relação de chaves primárias e estrangeiras, ao clicar em 1 elétron de um elemento, aparece um aviso para escolher o elétron com que haverá o relacionamento, se o jogador clicar nas coordenadas que envolvem o bitmap do elétron de outro elemento, a ligação é feita, ao fazer esta ligação, checa-se a quantidade atual de elétrons que o elemento possui, já que estes receberam 1 a mais. No fim, se todos tiverem exatamente 8 elétrons, o jogador vence e passa para a próxima etapa.

## 12.5 Simulador de Solubilidade

****

Figura 14 - Tela Solubilidade

### 12.5.1 Resumo sobre Solubilidade

Primeiramente ao se deparar com o jogo talvez você possa pensar: o que é solubilidade?

A solubilidade é basicamente a quantidade de uma substância que é possível se diluir em um determinado líquido.

A quantidade dessa substância pode variarde acordo com quantidade do seu líquido e também de acordo com sua temperatura.

Essa mistura gera três tipos de soluções, que podem ser:

* Solução Super Saturada;
* Solução Saturada;
* Solução Insaturada.

**Super Saturada:** Onde a quantidade máxima de uma substância foi atingida, e o líquido não pode mais diluir, a quantidade que sobra não foi diluídae fica no fundo do recipiente.

Como por um exemplo: se tivermos um suco, onde, quando colocamos uma quantidade maior de açúcar do que lhe é possível de se diluir, temos uma sobra de açúcar no fundo da jarra, assim, percebe-se que a solução é Super Saturada.

**Saturada:** Onde temos uma quantidade relativamente perfeita onde toda a substância é diluída pelo líquido em que é colocada, fazendo assim com que não haja sobras e nem falta desta substância.

Usando o mesmo exemplo da solução anterior, só que desta vez, colocandoa quantidade de açúcar e água indicada no rótulo do suco, teremos uma solução saturada.

**Insaturada:** Onde a quantidade da substância colocada no líquido é menor do que pode ser diluido, faltandocerta quantidade desta substância.

Com o mesmo exemplo, só que desta vez, colocando menos açúcar do que é recomendado, a solução fica fraca, precisando de mais acúcar, essa solução é Insaturada.

O objetivo do jogo é fazer com que a mistura que será feita seja Saturada. No simulador, os dados necessários para fazer os cálculos serão exibidos na tela, sendo eles a quantidade de termosnas caixas de texto para digitar os valores pedidos, onde uma destas, aleatoriamente, estará preenchida e bloqueada para edição, enquanto a outra estará vazia esperando a digitação de um valor.

O elemento que virá, também será de forma aleatória. Basta você fazer o que é chamado de regra de três, para descobrir o valor que deve ser colocado na caixa de texto, mas tome cuidado, pois se fizer a conta errada poderá perder pontos assim como produzirá uma mistura Super Saturada ou Insaturada.

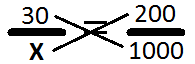
**Regra de três**

A regra de três é basica e fácil, e aqui iremos aprendê-la.

Vamos supor que temos o seguinte 30g/200 ml e queremos descobrir quando 1L então temos a seguinte forma para a conta:



Agora basta multiplicarmos pelo que chamamos de conta em X.



Ficaria então assim 200 x X = 30 x 1000, resolvendo isto:

200x = 30000

Então passaremos o 200 para o outro lado da conta só que se ele estava multiplicando passamos dividindo, assim:

X = 30000/200🡪X = 150

Assim acabamos descobrir que teremos 30g/200 ml = 150g/1000 ml então o valor a ser colocado para que haja uma mistura Saturada são 150.

# Referências Bibliográficas

Daltamir, B. A. (2005). *Universo da Química.* São Paulo: FTD.

Feltre, R. (2004). *QUÍMICA.* São Paulo: Moderna.

(s.d.). Acesso em 8 de Setembro de 2012, disponível em:  
<http://www.home.c2i.net/astandne>

(s.d.). Acesso em 8 de Setembro de 2012, disponível em:  
<www.univ-ab.pt/disciplinas/dcet/qg607/files/Chmcalc3.ZIP>

(s.d.). Acesso em 8 de Outubro de 2012, disponível em:  
<http://www.cdcc.sc.usp.br/elementos/>

(s.d.). Acesso em 8 de Outubro de 2012, disponível em:  
<http://pt.wikipedia.org/wiki/Tabela\_peri%C3%B3dica>

(s.d.). Acesso em 3 de Abril de 2013, disponível em:  
<http://www.fateczl.edu.br/crbst\_5.html>

(s.d.). Acesso em 4 de Abril de 2013, disponível em:  
<http://ireneslopes.wordpress.com/2013/06/13/linus-carl-pauling/>

(s.d.). Acesso em 4 de Abril de 2013, disponível em:

<http://www.significados.com.br/software/>

(s.d.). Acesso em 4 de Abril de 2013, disponível em:

http://pt.wikipedia.org/wiki/Inform%C3%A1tica

(s.d.). Acesso em 3 de Junho de 2013, disponível em:

<http://pt.wikipedia.org/wiki/Markup>

(s.d.). Acesso em 3 de Junho de 2013, disponível em:

<http://www.zrsistemas.com.br/index.php/usando-controller/margem-de-lucro-e-markup>

# Referências de Imagens

Figura 8 – Ilustração de um átomo dividido em camadas:  
Acesso em 8 de Outubro de 2012, disponível em:  
<http://comofas.com/como-fazer-a-distribuicao-eletronica-distribuir-os-eletrons-em-camadas-eletronicas/>

Figura 9 – Diagrama de Linus Pauling:  
Acesso em 9 de Outubro de 2012, disponível em:  
<http://www.profjoaoneto.com/quimicag/distribuicao2005.htm>

# Anexo A – Código-fonte

unit DB\_Integrator;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, ExtCtrls, DBCtrls, Grids, DBGrids, DB, ADODB, StdCtrls, Mask,

DBTables, Elementos, MMSystem;

// Informações dos elementos periódicos

var Elemento : Array[1..118] of TElemento; // Banco de dados

// Resolução do monitor do usuário

var Before\_Resolution : Array[1..2] of Integer; // Antigas configurações

type

TDataModule1 = class(TDataModule)

DataSource1: TDataSource;

ADOTable1: TADOTable;

ADOQuery1: TADOQuery;

ADOQuery2: TADOQuery;

ADOTable2: TADOTable;

DataSource2: TDataSource;

DataSource3: TDataSource;

ADOTable3: TADOTable;

ADOQuery3: TADOQuery;

procedure DataModuleCreate(Sender: TObject);

procedure Start();

procedure DataModuleDestroy(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

procedure PerformForm(Sender: TForm);

function RemovePoints( Text : String ) : Real;

function SpaceDelete(Text : String) : String;

function OrdinalToCardinal( Number : Byte ) : String;

function Crypt( Word : String ) : String;

var

DataModule1: TDataModule1;

implementation

uses Geometry;

{$R \*.dfm}

//------------------------------------------------------------------------------

// \* Converte números ordinais em cardinais ( Somente os 10 primeiros )

// Number : 0..9

//------------------------------------------------------------------------------

function OrdinalToCardinal( Number : Byte ) : String;

begin

// Conversão

case Number of

0: Result := 'Nula';

1: Result := 'Primeira';

2: Result := 'Segunda';

3: Result := 'Terceira';

4: Result := 'Quarta';

5: Result := 'Quinta';

6: Result := 'Sexta';

7: Result := 'Sétima';

8: Result := 'Oitava';

9: Result := 'Nona';

end;

end;

//------------------------------------------------------------------------------

// \* Deleta todos os espaços presentes em um Texto

// Text : Texto qualquer

//------------------------------------------------------------------------------

function SpaceDelete(Text : String) : String;

var I : Integer;

begin

I := 1;

// Para todos os caracteres do texto

while I <= Length(Text) do

begin

// Se encontrar um espaço

if Text[I] = #32 then

// Delete-o

Delete(Text,I,1)

// Caso não encontre um espaço

else

// Passa para o próximo caracter

I := I + 1;

end;

// Retorna o texto sem espaços

Result := Text;

end;

//------------------------------------------------------------------------------

// \* Substitui pontos(.) por vírgulas(,)

// Text : Texto qualquer

//------------------------------------------------------------------------------

function RemovePoints( Text : String ) : Real;

var I : Byte;

begin

I := 1;

// Para todos os caracteres do texto

while I <= Length(Text) do

begin

// Se encontrar um ponto(.)

if Text[I] = '.' then

begin

// Delete-o

Delete(Text,I,1);

// Insira uma vírgula em seu lugar

Insert(',',Text,I);

end;

// Passa para o próximo caracter

I := I + 1;

end;

// Retorna o texto como valor decimal

Result := StrToFloatDef(Text,0);

end;

//------------------------------------------------------------------------------

// \* Adapta o formulário

// Sender : Formulário que se adaptará

//------------------------------------------------------------------------------

procedure PerformForm(Sender: TForm);

begin

// Torna o cursor visível

ShowCursor(True);

// Com o formulário

With Sender do

begin

BorderStyle := BsNone; // Forma sem bordas

Width := Screen.Width; // Largura da forma proporcional à da tela

Height := Screen.Height; // Altura da forma proporcional à da tela

Left := 0; // Coordenada X inicial da forma

Top := 0; // Coordenada Y inicial da forma

DoubleBuffered := true;

AlphaBlend := true;

end;

end;

//------------------------------------------------------------------------------

// \* Criação da Classe

// Sender : Componente responsável pela execução

//------------------------------------------------------------------------------

procedure TDataModule1.DataModuleCreate(Sender: TObject);

begin

// Ativa o banco de dados

AdoTable1.Open;

AdoTable2.Open;

AdoTable3.Open;

// Ativa a conexão com o banco de dados

AdoQuery1.Open;

AdoQuery2.Open;

AdoQuery3.Open;

// Executa tabela professor

DataModule1.ADOQuery2.ExecSQL;

// Cria um novo Cursor

Screen.Cursors[ 5 ] := Windows.LoadCursorFromFile('Graphics/System/Cursor1.cur');

// Altera o cursor para o novo cursor criado

Screen.Cursor := 5;

// Inicializa o programa

Start();

end;

//------------------------------------------------------------------------------

// \* Alteração da Resolução da tela

// Width : Largura da tela

// Height : Altura da tela

//------------------------------------------------------------------------------

procedure Set\_ScreenResolution(Width, Height : Integer);

var Mode : TDevMode; // Instância para modificação de tela

begin

ZeroMemory(@mode, SizeOf( TDevMode )); // Zera as configurações

Mode.DmSize := SizeOf(TDevMode); // Define tipo de configuração

mode.DmPelsWidth := Width; // Nova largura da resolução

mode.DmPelsHeight := Height; // Nova altura da resolução

mode.DmFields := Dm\_PelsWidth or Dm\_PelsHeight; // Especifica modificações

ChangeDisplaySettings(Mode, 0); // Altera definitivamente

end;

//------------------------------------------------------------------------------

// \* Guarda informações de resolução

//------------------------------------------------------------------------------

procedure Keep\_ScreenResolution;

begin

Before\_Resolution[1] := Screen.Width; // Guarda a largura da tela

Before\_Resolution[2] := Screen.Height; // Guarda a altura da tela

end;

//------------------------------------------------------------------------------

// \* Altera a resolução para a resolução 1024x768

//------------------------------------------------------------------------------

procedure GoScreenSettings;

var HTaskBar : HWND; // Instância de componentes do sistema operacional

var R : TRect;

begin

Keep\_ScreenResolution(); // Guarda as configurações de vídeo

Set\_ScreenResolution(1024, 768); // Define novas configurações

SystemParametersInfo(SPI\_GETWORKAREA, 0, @r,0);

end;

//------------------------------------------------------------------------------

// \* Retorna à resolução padrão

//------------------------------------------------------------------------------

procedure ReturnScreenSettings;

var HTaskBar : HWND; // Instância de componentes do sistema operacional

begin

{ Retorna resolução }

Set\_ScreenResolution(Before\_Resolution[1], Before\_Resolution[2]);

HTaskBar := FindWindow('Shell\_TrayWnd', nil); // Instância a Barra de Tarefas

EnableWindow(HTaskBar, True); // Torna a barra acessível

ShowWindow(HTaskbar, Sw\_Show); // Torna a barra inacessível

end;

//------------------------------------------------------------------------------

// \* Destruição do formulário

// Sender : Componente responsável pela execução

//------------------------------------------------------------------------------

procedure TDataModule1.DataModuleDestroy(Sender: TObject);

begin

ReturnScreenSettings();

Application.Terminate();

end;

//------------------------------------------------------------------------------

// \* Inicializa o programa

//------------------------------------------------------------------------------

procedure TDataModule1.Start;

begin

GoScreenSettings(); // Define novas configurações de vídeo

{ Guarda informações em banco de dados interno }

Elemento[1] := TElemento. Create ( 1, 'Hidrogênio', 'H', 1.00794, '1', 1, 's', '1', 'Não-Metal', '53 pm', '37 pm', '120 pm', '1s1 ', 'hexagonal', 'gasoso', '14,025 K', '20,268 K', '209 Pa a 23 K', '1270 m/s a 20 °C', 14304, '0,1815 W/(m•K) ', '1312 kJ/mol', 'Diamagnético', '2,2', 'O hidrogênio é o elemento mais simples, constituído por um núcleo contendo um próton com um elétron orbitando em sua volta.' + ^m + ^m + 'Este elemento não é encontrado livre na natureza em sua forma atômica, é localizado sempre na composição de outras substâncias.' + ^m + 'Sua grande instabilidade o torna muito reativo.', 'Graphics/Properties/Pictures/1' );

Elemento[2] := TElemento. Create ( 2, 'Hélio', 'He', 4.002602, '18', 1, 's', '2', 'Gases Nobres', '21 pm', '32 pm', '140 pm', '1s2 ', 'hexagonal', 'gasoso', '0,95 K', '4,22 K', '100 Pa a 1,23 K', '972 m/s a 20 °C', 5193, '0,152 W/(m•K) ', '2372,3 kJ/mol', '-', '-', ' ' );

Elemento[3] := TElemento. Create ( 3, 'Lítio', 'Li', 6.941, '1', 2, 's', '2, 1', 'Metais Alcalinos', '152 pm', '134 pm', '182 pm', '1s2 2s1 ', 'cúbico de corpo centrado', 'sólido', '453 K', '1615 K', '1,63', '6000 m/s a 20 °C', 3582, '84,7 W/(m•K) ', '520,2 kJ/mol', 'Paramagnético', '0,98', ' ' );

Elemento[4] := TElemento. Create ( 4, 'Berílio', 'Be', 9.012182, '2', 2, 's', '2, 2', 'Metal Alcalinoterroso', '112 pm', '90 pm', '153 pm', '1s2 2s2 ', 'hexagonal', 'sólido', '1560 K', '2744 K', '1 Pa a 1462 K', '13000 m/s a 20 °C', 1825, '201 W/(m•K) ', '899,5 kJ/mol', 'Diamagnético', '1,57', ' ' );

Elemento[5] := TElemento. Create ( 5, 'Boro', 'B', 10.811, '13', 2, 's', '2, 3', 'Semimetal', '87 pm', '82 pm', '192 pm', '1s2 2s2 2p1 ', 'tetragonal', 'sólido', '2348 K', '4273 K', '0,348', '16200 m/s a 20 °C', 1026, '27,4 W/(m•K) ', '800,6 kJ/mol', '-', '2,04', ' ' );

Elemento[6] := TElemento. Create ( 6, 'Carbono', 'C', 12.0107, '14', 2, 'p', '2, 4', 'Não-Metal', '67 pm', '77 pm', '170 pm', '1s2 2s2 2p2 ', 'hexagonal', 'sólido', '-', '-', '-', '18 350 m/s a 20 °C', 710, '129 W/(m•K) ', '1086,5 kJ/mol', '-', '2,55', ' ' );

Elemento[7] := TElemento. Create ( 7, 'Nitrogênio', 'N', 14.0067, '15', 2, 'p', '2, 5', 'Não-Metal', '65 pm', '75 pm', '155 pm', '1s2 2s2 2p3 ', 'hexagonal', 'gasoso', '63,15 K', '75,36 K', '-', '334 m/s a 20 °C', 1040, '0,02598 W/(m•K) ', '1402,3 kJ/mol', 'Diamagnético', '3,04', ' ' );

Elemento[8] := TElemento. Create ( 8, 'Oxigênio', 'O', 15.9994, '16', 2, 'p', '2, 6', 'Não-Metal', '48 pm', '73 pm', '152 pm', '1s2 2s2 2p4 ', 'cúbico', 'gasoso', '50,35 K', '90,18 K', '-', '317,5 m/s a 20 °C', 920, '0,02674 W/(m•K) ', '1313,9 kJ/mol', 'Paramagnético', '3,44', ' ' );

Elemento[9] := TElemento. Create ( 9, 'Flúor', 'F', 18.9984032, '17', 2, 'p', '2, 7', 'Halogênio', '42 pm', '71 pm', '147 pm', '1s2 2s2 2p5 ', 'cúbico', 'gasoso', '53,53 K', '85,03 K', '-', '-', 824, '0,0279 W/(m•K) ', '1681 kJ/mol', '-', '3,98', ' ' );

Elemento[10] := TElemento. Create ( 10, 'Neônio', 'Ne', 20.1797, '18', 2, 'p', '2, 8', 'Gases Nobres', '38 pm', '69 pm', '154 pm', '1s2 2s2 2p6 ', 'cúbico de faces centradas', 'gasoso', '24,56 K', '27,07 K', '1 Pa a 12 K', '435 m/s a 20 °C', 103, '0,0493 W/(m•K) ', '2080,7 kJ/mol', '-', '-', ' ' );

Elemento[11] := TElemento. Create ( 11, 'Sódio', 'Na', 22.98976928, '1', 3, 's', '2, 8, 1', 'Metais Alcalinos', '186 pm', '154 pm', '227 pm', '1s2 2s2 2p6 3s1 ', 'cúbico de corpo centrado', 'sólido', '370,95 K', '1156 K', '1 Pa a 554 K', '3200 m/s a 20 °C', 1230, '141 W/(m•K) ', '495,8 kJ/mol', 'Paramagnético', '0,93', ' ' );

Elemento[12] := TElemento. Create ( 12, 'Magnésio', 'Mg', 24.305, '2', 3, 's', '2, 8, 2', 'Metal Alcalinoterroso', '160 pm', '130 pm', '173 pm', '1s2 2s2 2p6 3s2 ', 'hexagonal', 'sólido', '923 K', '1363 K', '361 Pa a 923 K', '4602 m/s a 20 °C', 1020, '156 W/(m•K) ', '737,7 kJ/mol', 'Paramagnético', '1,31', ' ' );

Elemento[13] := TElemento. Create ( 13, 'Alumínio', 'Al', 26.9815386, '13', 3, 'p', '2, 8, 3', 'Metais Representativos', '143 pm', '121 pm', '184 pm', '1s2 2s2 2p6 3s2 3p1 ', 'cúbico de faces centradas', 'sólido', '933,47 K', '2792 K', '-', '-', 900, '237 W/(m•K) ', '577,5 kJ/mol', 'Paramagnético', '1,61', ' ' );

Elemento[14] := TElemento. Create ( 14, 'Silício', 'Si', 28.0855, '14', 3, 'p', '2, 8, 4', 'Semimetais', '111 pm', '111 pm', '210 pm', '1s2 2s2 2p6 3s2 3p2 ', 'cúbico de faces centradas', 'sólido', '1687 K', '3538 K', '-', '8433 m/s a 20 °C', 700, '148 W/(m•K) ', '786,5 kJ/mol', '-', '1,9', ' ' );

Elemento[15] := TElemento. Create ( 15, 'Fósforo', 'P', 30.973762, '15', 3, 'p', '2, 8, 5', 'Não-Metais', '98 pm', '106 pm', '180 pm', '1s2 2s2 2p6 3s2 3p3 ', 'alótropos com várias estruturas', 'sólido', '317,3 K', '553,6 K', '(B)1 Pa a 279 K ou (V)1 Pa a 455 K', '-', 12.4, '0,236 W/(m•K) ', '1011,8 kJ/mol', '-', '2,19', ' ' );

Elemento[16] := TElemento. Create ( 16, 'Enxofre', 'S', 32.065, '16', 3, 'p', '2, 8, 6', 'Não-Metal', '88 pm', '102 pm', '180 pm', '1s2 2s2 2p6 3s2 3p4 ', 'ortorrômbico', 'sólido', '388,36 K', '717,75 K', '1 Pa a 375 K', '-', 710, '0,269 W/(m•K) ', '999,6 kJ/mol', '-', '2,58', ' ' );

Elemento[17] := TElemento. Create ( 17, 'Cloro', 'Cl', 35.453, '17', 3, 'p', '2, 8, 7', 'Halogênio', '79 pm', '99 pm', '175 pm', '1s2 2s2 2p6 3s2 3p5 ', 'ortorrômbico', 'gasoso', '171,6 K', '239,11 K', '1000 Pa a 170 K', '-', 480, '0,089 W/(m•K) ', '1251,2 kJ/mol', '-', '3,16', ' ' );

Elemento[18] := TElemento. Create ( 18, 'Argônio', 'Ar', 39.948, '18 ', 3, 'p', '2, 8, 8', 'Gases Nobres', '71 pm', '97 pm', '188 pm', '1s2 2s2 2p6 3s2 3p6 ', 'cúbico de faces centradas', 'gasoso', '83,80 K', '87,30 K', '-', '319 m/s a 20 °C', 520, '0,01772 W/(m•K) ', '1520,6 kJ/mol', '-', '-', ' ' );

Elemento[19] := TElemento. Create ( 19, 'Potássio', 'K', 39.0983, '1', 4, 's', '2, 8, 8, 1', 'Metais Alcalinos', '243 pm', '196 pm', '275 pm', '1s2 2s2 2p6 3s2 3p6 4s1 ', 'cúbico de corpo centrado', 'sólido', '336,53 K', '1032 K', '0,000106 Pa a 336,5 K', '2000 m/s a 20 °C', 757, '102,4 W/(m•K) ', '418,8 kJ/mol', 'Paramagnético', '0,82', ' ' );

Elemento[20] := TElemento. Create ( 20, 'Cálcio', 'Ca', 40.078, '2', 4, 's', '2, 8, 8, 2', 'Metal Alcalinoterroso', '194 pm', '174 pm', '-', '1s2 2s2 2p6 3s2 3p6 4s2 ', 'cúbico de faces centradas', 'sólido', '1115 K', '1757 K', '254 Pa a 1112 K', '3810 m/s a 20 °C', 632, '201 W/(m•K) ', '589,8 kJ/mol', 'Paramagnético', '1', ' ' );

Elemento[21] := TElemento. Create ( 21, 'Escândio', 'Sc', 44.955912, '3', 4, 'd', '2, 8, 9, 2', 'Metais de Transição', '184 pm', '144 pm', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d1 ', 'hexagonal', 'sólido', '1814 K', '3109 K', '22,1 Pa a 1112 K', '-', 568, '15,8 W/(m•K) ', '633,1 kJ/mol', '-', '1,36', ' ' );

Elemento[22] := TElemento. Create ( 22, 'Titânio', 'Ti', 47.867, '4', 4, 'd', '2, 8, 10, 2', 'Metais de Transição', '176 pm', '136 pm', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d2 ', 'hexagonal', 'sólido', '1941 K', '3560 K', '0,49 Pa a 1933 K', '4140 m/s a 20 °C', 520, '21,9 W/(m•K) ', '658,8 kJ/mol', '-', '1,54', ' ' );

Elemento[23] := TElemento. Create ( 23, 'Vanádio', 'V', 50.9415, '5', 4, 'd', '2, 8, 11, 2', 'Metais de Transição', '134 pm', '125 pm', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d3 ', 'cúbica centrada no corpo', 'sólido', '2183 K', '3680 K', '1 Pa a 2101 K', '4560 m/s a 20 °C', 489, '30,7 W/(m•K) ', '650,9 kJ/mol', '-', '1,63', ' ' );

Elemento[24] := TElemento. Create ( 24, 'Cromo', 'Cr', 51.9961, '6', 4, 'd', '2, 8, 13, 1', 'Metais de Transição', '140 pm', '127 pm', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d4 ', 'cúbico de corpo centrado', 'sólido', '2180 K', '2944 K', '1 Pa a 1656 K', '5940 m/s a 20 °C', 448, '93,7 W/(m•K) ', '652,9 kJ/mol', 'Paramagnético', '1,66', ' ' );

Elemento[25] := TElemento. Create ( 25, 'Manganês', 'Mn', 54.938045, '7', 4, 'd', '2, 8, 13, 2', 'Metais de Transição', '127 pm', '139 pm', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d5 ', 'cúbico de corpo centrado', 'sólido', '1519 K', '2334 K', '1 Pa a 1228 K', '5150 m/s a 20 °C', 479, '7,82 W/(m•K) ', '717,3 kJ/mol', '-', '1,55', ' ' );

Elemento[26] := TElemento. Create ( 26, 'Ferro', 'Fe', 55.845, '8', 4, 'd', '2, 8, 14, 2', 'Metais de Transição', '156 pm', '125 pm', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d6 ', 'cúbico de corpo centrado', 'sólido', '1811 K', '3134 K', '7,05 Pa a 1808 K', '4910 m/s a 20 °C', 440, '80,2 W/(m•K) ', '762,5 kJ/mol', 'Ferromagnético', '1,83', ' ' );

Elemento[27] := TElemento. Create ( 27, 'Cobalto', 'Co', 58.933195, '9', 4, 'd', '2, 8, 15, 2', 'Metais de Transição', '126 pm', '152 pm', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d7 ', 'hexagonal', 'sólido', '1768 K', '3200 K', '1 Pa a 1790 K', '4720 m/s a 20 °C', 420, '100 W/(m•K) ', '760,4 kJ/mol', 'Ferromagnético', '1,88', ' ' );

Elemento[28] := TElemento. Create ( 28, 'Níquel', 'Ni', 58.6934, '10', 4, 'd', '2, 8, 17, 1', 'Metais de Transição', '124 pm', '124±4 pm', '163 pm', '1s2 2s2 2p6 3s2 3p6 4s2 3d8 ', 'cúbico de faces centradas', 'sólido', '1728 K', '3186 K', '1 Pa a 1783 K', '4970 m/s a 20 °C', 444, '90,7536 W/(m•K) ', '737,1 kJ/mol', 'Ferromagnético', '1,91', ' ' );

Elemento[29] := TElemento. Create ( 29, 'Cobre', 'Cu', 63.546, '11', 4, 'd', '2, 8, 18, 1', 'Metais de Transição', '128 pm', '132±4 pm', '140 pm', '1s2 2s2 2p6 3s2 3p6 4s2 3d9 ', 'cúbico de faces centradas', 'sólido', '1357,77 K', '2835 K', '1 Pa a 1509 K', '3570 m/s a 20 °C', 385, '401 W/(m•K) ', '745,5 kJ/mol', 'Diamagnético', '1,9', ' ' );

Elemento[30] := TElemento. Create ( 30, 'Zinco', 'Zn', 65.38, '12', 4, 'd', '2, 8, 18, 2', 'Metais de Transição', '134 pm', '122±4 pm', '139 pm', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 ', 'hexagonal', 'sólido', '692,68 K', '1180 K', '1 Pa a 610 K', '3700 m/s a 20 °C', 444, '116 W/(m•K) ', '906,4 kJ/mol', '-', '1.65', ' ' );

Elemento[31] := TElemento. Create ( 31, 'Gálio', 'Ga', 69.723, '13', 4, 'p', '2, 8, 18, 3', 'Metais representativos', '135 pm', '122±3 pm', '187 pm', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p1 ', 'Ortorrômbico', 'Sólido', '302,9146 K', '2477 K', '1Pa a 1310 K', '2740 m/s a 20ºC', 0, '40,6 W/(m-K) ', '578,8 kJ/mol', '', '1,81', ' ' );

Elemento[32] := TElemento. Create ( 32, 'Germânio', 'Ge', 72.64, '14', 4, 'p', '2, 8, 18, 4', 'Semimetal', '122 pm', '122 pm', '211 pm', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p2 ', 'Cúbico de faces centradas', 'Sólido', '1211,40 K', '3106 K', '1 Pa a 1644 K', '5400 m/s a 20ºC', 320, '59,9 W/(m-K) ', '762 kJ/mol', '', '2,01', ' ' );

Elemento[33] := TElemento. Create ( 33, 'Arsênio', 'As', 74.9216, '15', 4, 'p', '2, 8, 18, 5', 'Semimetal', '119 pm', '119±4 pm', '185 pm', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p3 ', 'Romboédrico', 'Sólido', '1090 K', '887 K', '1 Pa a 553 K', '', 330, '50 W/(m-K) ', '947,0 kJ/mol', '', '2,18', ' ' );

Elemento[34] := TElemento. Create ( 34, 'Selênio', 'Se', 78.96, '16', 4, 'p', '2, 8, 18, 6', 'Não metal', '120 pm', '120±4 pm', '190 pm', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p4 ', 'Hexagonal', 'Sólido', '494 K', '958 K', '1 Pa a 500 K', '3350 m/s a 20ºC', 320, '2,04 W/(m-K) ', '941 kJ/mol', '', '2,55', ' ' );

Elemento[35] := TElemento. Create ( 35, 'Bromo', 'Br', 79.904, '17', 4, 'p', '2, 8, 18, 7', 'Não metal', '120 pm', '120±3 pm', '185 pm', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p5 ', 'Ortorrômbico', 'Líquido', '265,8 K', '332 K', '1 Pa a 185 K', '206m/s a 20ºC', 480, '0,122 W/(m-K) ', '1139,9 kJ/mol', '', '2,96', ' ' );

Elemento[36] := TElemento. Create ( 36, 'Criptônio', 'Kr', 83.798, '18', 4, 'p', '2, 8, 18, 8', 'Gás nobre', '88 pm', '116±4 pm', '202 pm', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 ', 'Cúbico de faces centradas', 'Gasoso', '115,79 K', '119,93 K', '1 Pa a 59 K', 'Gás: 220 m/s a 23ºC', 248, '0,00949 W/(m•K) ', '1350,8 kJ/mol', '', '3', ' ' );

Elemento[37] := TElemento. Create ( 37, 'Rubídio', 'Rb', 85.4678, '1', 5, 's', '2, 8, 18, 8, 1', 'Metal Alcalino', '248 pm', '211 pm', '2,44 pm', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s1 ', 'Cúbica centrada no corpo', 'Sólido', '312,46 K', '961 K', '434 Pa a 1 K', '1300 m/s a 20 °C', 0, '58,2 W/(m•K) ', '403,0 kJ/mol', 'Paramagnético', '0,82', ' ' );

Elemento[38] := TElemento. Create ( 38, 'Estrôncio', 'Sr', 87.62, '2', 5, 's', '2, 8, 18, 8, 2', 'Metal alcalinoterroso', '219 pm', '192 pm', '249 pm', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 ', 'Cúbico de corpo centrado', 'Sólido', '1050 K', '1655 K', '1 Pa a 796 K', '', 0, '35,3 W/(m•K) ', '549,5 kJ/mol', 'Paramagnético', '0,95', ' ' );

Elemento[39] := TElemento. Create ( 39, 'Ítrio', 'Y', 88.90585, '3', 5, 'd', '2, 8, 18, 9, 2', 'Metal de transição', '180 pm', '190±7 pm', '', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d1 ', 'Hexagonal', 'Sólido', '1799 K', '3609 K', '1 Pa a 1883 K', '3300 m/s a 20 °C', 300, '17,2 W/(m•K) ', '600 kJ/mol', '', '1,22', ' ' );

Elemento[40] := TElemento. Create ( 40, 'Zircônio', 'Zr', 91.224, '4', 5, 'd', '2, 8, 18, 10, 2', 'Metal de transição', '160 pm', '175±7 pm', '', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d2 ', 'Hexagonal', 'Sólido', '2128 K', '4682 K', '1 Pa a 2639 K', '3800 m/s a 20 °C', 0.27, '22,7 W/(m•K) ', '640,1 kJ/mol', '', '1,33', ' ' );

Elemento[41] := TElemento. Create ( 41, 'Nióbio', 'Nb', 92.90638, '5', 5, 'd', '2, 8, 18, 12, 1', 'Metal de transição', '146 pm', '164±6 pm', '', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d3 ', 'Cúbica centrada no corpo', 'Sólido', '2750 K', '5017 K', '1 Pa a 2942 K', '3480 m/s a 20 °C', 265, '53,7 W/(m•K) ', '652,1 kJ/mol', '', '1,6', ' ' );

Elemento[42] := TElemento. Create ( 42, 'Molibdênio', 'Mo', 95.96, '6', 5, 'd', '2, 8, 18, 13, 1', 'Metal de transição', '139 pm', '154±5 pm', '', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d4 ', 'Cúbica centrada no corpo', 'Sólido', '2896 K', '4912 K', '1 Pa a 2742 K', '', 250, '138 W/(m•K) ', '684,3 kJ/mol', '', '2,16', ' ' );

Elemento[43] := TElemento. Create ( 43, 'Tecnécio', 'Tc', 98, '7', 5, 'd', '2, 8, 18, 13, 2', 'Metal de transição', '136 pm', '147±7 pm', '', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d5 ', 'Hexagonal', 'Sólido', '2430 K', '4538 K', '1 Pa a 2727 K', '16200 m/s a 20 °C', 210, '50,6 W/(m•K) ', '702 kJ/mol', 'Paramagnético', '1,9', ' ' );

Elemento[44] := TElemento. Create ( 44, 'Rutênio', 'Ru', 101.07, '8', 5, 'd', '2, 8, 18, 15, 1', 'Metal de transição', '134 pm', '146±7 pm', '', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d6 ', 'Hexagonal', 'Sólido', '2607 K', '4423 K', '1 Pa a 2588 K', '5970 m/s a 20 °C', 238, '117 W/(m•K) ', '710,2 kJ/mol', '', '2,2', ' ' );

Elemento[45] := TElemento. Create ( 45, 'Ródio', 'Rh', 102.9055, '9', 5, 'd', '2, 8, 18, 16, 1', 'Metal de transição', '134 pm', '142±7 pm', '', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d7 ', 'Cúbico de faces centradas', 'Sólido', '2237 K', '3968 K', '1 Pa a 2288 K', '4700 m/s a 20 °C', 242, '150 W/(m•K) ', '719,7 kJ/mol', '', '2,28', ' ' );

Elemento[46] := TElemento. Create ( 46, 'Paládio', 'Pd', 106.42, '10', 5, 'd', '2, 8, 18, 18', 'Metal de transição', '137 pm', '139±6 pm', '163 pm', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d8 ', 'Cúbico de faces centradas', 'Sólido', '1828,05 K', '3236 K', '1 Pa a 1721 K', '3070 m/s a 20 °C', 244, '71,8 W/(m•K) ', '804,4 kJ/mol', '', '2,2', ' ' );

Elemento[47] := TElemento. Create ( 47, 'Prata', 'Ag', 107.8682, '11', 5, 'd', '2, 8, 18, 18, 1', 'Metal de transição', '144 pm', '145±5 pm', '172 pm', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d9 ', 'Cúbico de faces centradas', 'Sólido', '1234,93 K', '2435 K', '1 Pa a 1283 K', '2680 m/s a 20 °C', 0, '429 W/(m•K) ', '731 kJ/mol', 'Diamagnético', '1,93', ' ' );

Elemento[48] := TElemento. Create ( 48, 'Cádmio', 'Cd', 112.411, '12', 5, 'd', '2, 8, 18, 18, 2', 'Metal de transição', '151 pm', '144±9 pm', '158 pm', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 ', 'Hexagonal', 'Sólido', '594,22 K', '1040 K', '1 Pa a 530 K', '2310 m/s a 20 °C', 26.02, '96,8 W/(m•K) ', '867,8 kJ/mol', '', '1,69', ' ' );

Elemento[49] := TElemento. Create ( 49, 'Índio', 'In', 114.818, '13', 5, 'p', '2, 8, 18, 18, 3', 'Metais representativos', '167 pm', '142±5 pm', '193 pm', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p1 ', 'Tetragonal', 'Sólido', '429,7485 K', '2345 K', '1 Pa a 1196 K', '1215 m/s a 20 °C', 26.74, '81,6 W/(m•K) ', '558,3 kJ/mol', '', '1,78', ' ' );

Elemento[50] := TElemento. Create ( 50, 'Estanho', 'Sn', 118.71, '14', 5, 'p', '2, 8, 18, 18, 4', 'Metais representativos', '140 pm', '139±4 pm', '217 pm', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p2 ', 'Tetragonal', 'Sólido', '505,08 K', '2875 K', '1 Pa a 1497 K', '2500 m/s a 20 °C', 228, '66,6 W/(m•K) ', '708,6 kJ/mol', '', '1,96', ' ' );

Elemento[51] := TElemento. Create ( 51, 'Antimônio', 'Sb', 121.76, '15 ', 5, 'p', '2, 8, 18, 18, 5', 'Semimetal', '140 pm', '139±5 pm', '206 pm', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p3 ', 'Romboédrico', 'Sólido', '903,78 K', '1860 K', '1 Pa a 807 K', '', 210, '24,3 W/(m•K) ', '834 kJ/mol', '', '2,05', ' ' );

Elemento[52] := TElemento. Create ( 52, 'Telúrio', 'Te', 127.6, '16', 5, 'p', '2, 8, 18, 18, 6', 'Semimetal', '140 pm', '138±4 pm', '206 pm', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p4 ', 'Hexagonal', 'Sólido', '722,66 K', '1261 K', '100 Pa a 1266 K', '', 202, '2,35 W/(m•K) ', '869,3 kJ/mol', '', '2,1', ' ' );

Elemento[53] := TElemento. Create ( 53, 'Iodo', 'I', 126.90447, '17', 5, 'p', '2, 8, 18, 18, 7', 'Halogênios', '140 pm', '139±3 pm', '198 pm', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p5 ', 'Ortorrômbico', 'Sólido', '386,85 K', '457,6 K', '100 Pa a 309 K', '', 145, '8 x 10-8 S/m ', '1008,4 kJ/mol', '', '2,66', ' ' );

Elemento[54] := TElemento. Create ( 54, 'Xenônio', 'Xe', 131.293, '18', 5, 'p', '2, 8, 18, 18, 8', 'Gás nobre', '108 pm', '130 pm', '216 pm', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 ', 'Cúbico de faces centradas', 'Gasoso', '161,36 K', '165,03 K', '100 Pa a 103 K', '1090 m/s a 20 °C', 158, '0,00569 W/(m•K) ', '1170,4 kJ/mol', '', '2,6', ' ' );

Elemento[55] := TElemento. Create ( 55, 'Césio', 'Cs', 132.9054519, '1', 6, 's', '2, 8, 18, 18, 8, 1', 'Metal Alcalino', '298 pm', '225 pm', '', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s1 ', 'Cúbica centrada no corpo', 'Sólido', '301,6 K', '944 K', '1 Pa a 418 K', '', 0, '35,9 W/(m•K) ', '375,7 kJ/mol', 'Paramagnético', '0,79', ' ' );

Elemento[56] := TElemento. Create ( 56, 'Bário', 'Ba', 137.327, '2', 6, 's','2, 8, 18, 18, 8, 2','Metal Alcalinoterroso','222 pm','215±11 pm','268 pm', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 ', 'Cúbica centrada no corpo', 'Sólido', '1000 K', '2170 K', '100 Pa a 1185 K', '1620 m/s a 20 °C', 0, '18,4 W/(m•K) ', '502,9 kJ/mol', 'Paramagnético', '0,89', ' ' );

Elemento[57] := TElemento. Create ( 57, 'Lantânio', 'La', 138.90547, '-', 6, 'f', '2, 8, 18, 18, 9, 2', 'Lantanídios', '195 pm', '169 pm', '', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f1 ', 'Hexagonal', 'Sólido', '1191 K', '3737 K', '100 Pa a 2458 K', '2475 m/s a 20 °C', 0, '13,5 W/(m•K) ', '538,1 kJ/mol', '', '1,1', ' ' );

Elemento[58] := TElemento. Create ( 58, 'Cério', 'Ce', 140.116, '-', 6, 'f', '2, 8, 18, 19, 9, 2', 'Lantanídios', '185 pm', '', '', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f2 ', 'Cúbico de faces centradas', 'Sólido', '1071 K', '3699 K', '100 Pa a 2442 K', '2100 m/s a 20 °C', 0, '11,4 W/(m•K) ', '534,4 kJ/mol', '', '1,12', ' ' );

Elemento[59] := TElemento. Create ( 59, 'Praseodímio', 'Pr', 140.93765, '-', 6, 'f', '2, 8, 18, 21, 8, 2', 'Lantanídios', '185 pm', '165 pm', '', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f3 ', 'Hexagonal', 'Sólido', '1204 K', '3793 K', '100 Pa a 2227 K', '2280 m/s a 20 °C', 0, '12,5 W/(m•K) ', '527 kJ/mol', '', '1,13', ' ' );

Elemento[60] := TElemento. Create ( 60, 'Neodímio', 'Nd', 144.242, '-', 6, 'f', '2, 8, 18, 22, 8, 2', 'Lantanídios', '185 pm', '', '', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f4 ', 'Hexagonal', 'sólido', '1294 K', '3347', '100 Pa a 1998 K', '2330 m/s a 20 °C', 190, '16,5 W/(m°C) ', '533,1 kJ/mol', 'diamagnético', '1,14', ' ' );

Elemento[61] := TElemento. Create ( 61, 'Promécio', 'Pm', 144.9127, '-', 2, 's', '2,8,18,23,8,2', 'Lantanídeos', '!', '1,63 A', '!', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f5 ', 'Hexagonal', 'Sólido', '!', '!', '120 Pa a 1450 K', '2340 m/s a 20 °C', 1975, '1,6 E6/ohms/m ', '5,55', 'Paramagnético', '1,13', ' ' );

Elemento[62] := TElemento. Create ( 62, 'Samário', 'Sm', 150.36, '-', 2, 's', '2,8,18,24,8,2', 'Lantanídeos', '1,81 A', '1,62 A', '!', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f6 ', 'Trigonal', 'Sólido', '1072', '1900', '1 Pa a 1001 K', '2400 m/s a 20 °C', 986, '2 E6/ohms/m ', '5,63', 'Diamagnético', '1,17', ' ' );

Elemento[63] := TElemento. Create ( 63, 'Európio', 'Eu', 151.964, '-', 2, 's', '2,8,18,25,8,2', 'Lantanídeos', '1,99 A', '1,85 A', '!', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f7 ', 'Cúbica de Corpo Centrado', 'Sólido', '822', '1597', '1 Pa a 863 K', '1000 m/s a 20 °C', 1345, '1,1 E6/ohms/m ', '5,67', 'Diamagnético', '1,2', ' ' );

Elemento[64] := TElemento. Create ( 64, 'Gadolínio', 'Gd', 157.25, '-', 2, 's', '2,8,18,25,9,2', 'Lantanídeos', '1,8 A', '1,61 A', '!', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f8 ', 'Hexagonal Compactada', 'Sólido', '1311', '3233', '10 Pa a 1979 K', '1300 m/s a 20 °C', 1670, '1,1 E6/ohms/m ', '6,15', 'Paramagnético', '1,2', ' ' );

Elemento[65] := TElemento. Create ( 65, 'Térbio', 'Tb', 158.92535, '-', 2, 's', '2,8,18,27,8,2', 'Lantanídeos', '1,8 A', '1,59 A', '!', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f9 ', 'Hexagonal Compactada', 'Sólido', '1360', '3041', '10 Pa a 1979 K', '2629 m/s a 20°C', 1815, '0,8 E6/ohms/m ', '5,86', 'Paramagnético', '1,2', ' ' );

Elemento[66] := TElemento. Create ( 66, 'Disprósio', 'Dy', 162.5, '-', 2, 's', '2,8,18,28,8,2', 'Lantanídeos', '1,8 A', '1,59 A', '!', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f10 ', 'Hexagonal Compactada', 'Sólido', '1412', '2562', '10 Pa a 1523 K', '2710 m/s a 20°C', 170, '0,9 E6/ohms/m ', '5,93', 'Paramagnético', '1,22', ' ' );

Elemento[67] := TElemento. Create ( 67, 'Hólmio', 'Ho', 164.93032, '-', 2, 's', '2,8,18,29,8,2', 'Lantanídeos', '1,79 A', '1,58 A', '!', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f11 ', 'Hexagonal Compactada', 'Sólido', '1470', '2720', '10 Pa a 1584 K', '2710 m/s a 20°C', 160, '1,1 E6/ohms/m ', '6,02', 'Paramagnético', '1,23', ' ' );

Elemento[68] := TElemento. Create ( 68, 'Érbio', 'Er', 167.259, '-', 2, 's', '2,8,18,30,8,2', 'Lantanídeos', '1,78 A', '1,57 A', '!', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f12 ', 'Hexagonal Compactada', 'Sólido', '1522', '2510', '1 Pa a 1504 K', '2310 m/s a 20°C', 170, '1,1 E6/ohms/m ', '6,101', 'Paramagnético', '1,24', ' ' );

Elemento[69] := TElemento. Create ( 69, 'Túlio', 'Tm', 168.93421, '-', 2, 's', '2,8,18,31,8,2', 'Lantanídeos', '1,77 A', '1,56 A', '!', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f13 ', 'Hexagonal Compactada', 'Sólido', '1545', '1727', '10 Pa a 1235 K', '2239 m/s a 20°C', 160, '1,2 E6/ohms/m ', '6,184', 'Diamagnético', '1,25', ' ' );

Elemento[70] := TElemento. Create ( 70, 'Itérbio', 'Yb', 173.04, '-', 2, 's', '2,8,18,32,8,2', 'Lantanídeos', '1,94 A', '1,74 A', '!', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 ', 'Cúbica de Corpo Centrado', 'Sólido', '824', '1466', '1 Pa a 736 K', '1590 m/s a 20 °C', 150, '1,3 E6/ohms/m ', '6,254', 'Paramagnético', '1,1', ' ' );

Elemento[71] := TElemento. Create ( 71, 'Lutécio', 'Lu', 174.967, '3', 2, 's', '2,8,18,32,9,2', 'Lantanídeos', '1,75 A', '1,56 A', '!', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d1 ', 'Hexagonal Compactada', 'Sólido', '1656', '3315', '100 Pa a 2346 K', '2259 m/s a 20°C', 150, '3,7 E6/ohms/m ', '5,43', 'Paramagnético', '1,27', ' ' );

Elemento[72] := TElemento. Create ( 72, 'Háfnio', 'Hf', 178.49, '4', 2, 's', '2,8,18,32,10,2', 'Metais de Transição', '1,67 A', '1,44 A', '!', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d2 ', 'Hexagonal Compactada', 'Sólido', '2150', '5400', '100 Pa a 3277 K', '3010 m/s a 20°C', 140, '1,5 E6/ohms/m ', '6,65', 'Diamagnético', '1,3', ' ' );

Elemento[73] := TElemento. Create ( 73, 'Tantálio', 'Ta', 180.94788, '5', 2, 's', '2,8,18,32,11,2', 'Metais de Transição', '1,49 A', '1,34 A', '!', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d3 ', 'Cúbica de Corpo Centrado', 'Sólido', '2996', '5425+-100', '0,776 Pa a 3269 K', '3400 m/s a 20°C', 150, '3,6 E6/ohms/m ', '7,89', 'Paramagnético', '1,5', ' ' );

Elemento[74] := TElemento. Create ( 74, 'Tungstênio', 'W', 183.84, '6', 2, 's', '2,8,18,32,12,2', 'Metais de Transição', '1,41 A', '1,30 A', '!', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d4 ', 'Cúbica de Corpo Centrado', 'Sólido', '3410+-20', '5660', '4,27 Pa a 3269 K', '5174 m/s a 20°C', 140, '8,1 E6/ohms/m ', '7,98', 'Paramagnético', '2,36', ' ' );

Elemento[75] := TElemento. Create ( 75, 'Rênio', 'Ra', 186.207, '7', 2, 's', '2,8,18,32,13,2', 'Metais de Transição', '1,37 A', '1,28 A', '!', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d5 ', 'Hexagonal Compactada', 'Sólido', '3180', '5627', '10 Pa a 3614 K', '4700 m/s a 20°C', 137, '18,2 E6/ohms/m ', '7,88', 'Paramagnético', '1,9', ' ' );

Elemento[76] := TElemento. Create ( 76, 'Ósmio', 'Os', 190.23, '8', 2, 's', '2,8,18,32,14,2', 'Metais de Transição', '1,35 A', '1,26 A', '!', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d6 ', 'Hexagonal Compactada', 'Sólido', '3045', '5027', '10 Pa a 3423 K', '4940 m/s a 20°C', 130, '5,8 E6/ohms/m ', '8,7', 'Paramagnético', '2,2', ' ' );

Elemento[77] := TElemento. Create ( 77, 'Irídio', 'Ir', 192.217, '9', 2, 's', '2,8,18,32,15,2', 'Metais de Transição', '1,35 A', '1,27 A', '!', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d7 ', 'Cúbica de Face Centrado', 'Sólido', '2410', '4527+-100', '10 Pa a 2957 K', '4825 m/s a 20°C', 130, '12,3 E6/ohms/m ', '9,1', 'Paramagnético', '2,2', ' ' );

Elemento[78] := TElemento. Create ( 78, 'Platina', 'Pt', 195.084, '10', 2, 's', '2,8,18,32,17,1', 'Outros Metais', '1,36 A', '1,3 A', '!', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d8 ', 'Cúbica de Face Centrado', 'Sólido', '1772', '3827', '100 Pa a 2815 K', '2680 m/s a 20°C', 130, '21,3 E6/ohms/m ', '9', 'Paramagnético', '2,28', ' ' );

Elemento[79] := TElemento. Create ( 79, 'Ouro', 'Au', 196.966569, '11', 2, 's', '2,8,18,32,18,1', 'Outros Metais', '1,39 A', '1,34 A', '166 pm', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d9 ', 'Cúbica de Face Centrado', 'Sólido', '1064,43', '2807', '0,000237 Pa a 1337 K', '1740 m/s a 20°C', 128, '9,4 E6/ohms/m ', '9,225', 'Diamagnético', '2,54', ' ' );

Elemento[80] := TElemento. Create ( 80, 'Mercúrio', 'Hg', 200.59, '12', 2, 's', '2,8,18,32,18,2', 'Outros Metais', '1,46 A', '1,49 A', '155 pm', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 ', 'Romboédrica', 'Liquido', '-38,87', '356,58', '10 Pa a 350 K', '1407 m/s a 20°C', 140, '48,8 E6/ohms/m ', '10,437', 'Diamagnético', '2', ' ' );

Elemento[81] := TElemento. Create ( 81, 'Tálio', 'Ti', 204.3833, '13', 2, 's', '2,8,18,32,18,3', 'Outros Metais', '1,6 A', '1,48 A', '196 pm', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p1 ', 'Hexagonal', 'Sólido', '303,5', '1457+-10', '100 Pa a 1097 K', '818 m/s a 20°C', 129, '1 E6/ohms/m ', '6,108', 'Diamagnético', '2,04', ' ' );

Elemento[82] := TElemento. Create ( 82, 'Chumbo', 'Pb', 207.2, '14', 2, 's', '2,8,18,32,18,4', 'Outros Metais', '1,71 A', '1,47 A', '202 pm', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p2 ', 'Cúbica de Face Centrado', 'Sólido', '327,5', '1740', '4,21x10(a menos 7)Pa a 1097', '1260 m/s a 20°C', 129, '5,6 E6/ohms/m ', '7,416', 'Diamagnético', '2,33', ' ' );

Elemento[83] := TElemento. Create ( 83, 'Bismuto', 'Bi', 208.9804, '15', 2, 's', '2,8,18,32,18,5', 'Outros Metais', '1,75 A', '1,46 A', '207 pm', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p3 ', 'Monocíclica', 'Sólido', '271,3', '1560+-5', '1 Pa a 941 K', '1790 m/s a 20°C', 122, '4,8 E6/ohms/m ', '7,289', '!', '2,02', ' ' );

Elemento[84] := TElemento. Create ( 84, 'Polônio', 'Po', 208.9824, '16', 2, 's', '2,8,18,32,18,6', 'Outros Metais', '1,7 A', '1,46 A', '!', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p4 ', 'Cúbica', 'Sólido', '254', '962', '10 Pa a 1003 K', 's.d m/s a 20°C', 0, '0,9 E6/ohms/m ', '8,42', 'Não Magnético', '2', ' ' );

Elemento[85] := TElemento. Create ( 85, 'Astato', 'At', 208.9871, '17', 2, 's', '2,8,18,32,18,7', 'Metais Halogênios', '1,67 A', '1,45 A', '!', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p5 ', '!', 'Sólido', '302', '337', '10 Pa a 392 K', 's.d m/s a 20°C', 0, '0,7 E6/ohms/m ', '!', 's.d', '2,2', ' ' );

Elemento[86] := TElemento. Create ( 86, 'Radônio', 'Rn', 206.0176, '18', 2, 's', '2,8,18,32,18,8', 'Gases Nobres', '1,45 A', '!', '220 pm', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 ', '!', 'Gasoso', '-71', '-61,8', '1 Pa a 110 K', 's.d m/s a 20°C', 0, '! ', '10,748', 's.d', '!', ' ' );

Elemento[87] := TElemento. Create ( 87, 'Frâncio', 'Fr', 223.0197, '1', 2, 's', '2,8,18,32,18,8,1', 'Metais Alcalinos', '1,34 A', '!', '348 pm', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s1 ', '!', 'Sólido', '27', '677', '1 Pa a 404 K', 's.d m/s a 20°C', 0, '! ', '0', 'Paramagnético', '0,7', ' ' );

Elemento[88] := TElemento. Create ( 88, 'Rádio', 'Ra', 226.0254, '2', 2, 's', '2,8,18,32,18,8,2', 'Metais Alcalinos - Terrosos', '2,7 A', '!', '283 pm', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 ', 'Cúbica de Corpo Centrado', 'Sólido', '700', '1737', '1 Pa a 819 K', 's.d m/s a 20°C', 0, '! ', '5,279', 'Paramagnético', '0,9', ' ' );

Elemento[89] := TElemento. Create ( 89, 'Actínio', 'Ac', 227.0278, '-', 2, 's', '2,8,18,32,18,9,2', 'Actinídeos', '2,33 A', '!', '!', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 5f1 ', 'Cúbica de Face Centrado', 'Sólido', '1050', '3200+-300', '!', '!', 0, '1 E6/ohms/m ', '5,17', '!', '1,1', ' ', 'Graphics/Properties/Pictures/89' );

Elemento[90] := TElemento. Create ( 90, 'Tório', 'Th', 232.03806, '-', 2, 's', '2,8,18,32,18,10,2', 'Actinídeos', '1,88 A', '1,65 A', '!', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 5f2 ', 'Cúbica de Face Centrado', 'Sólido', '1750', '4790', '1 Pa a 2633 K', '2490 m/s a 20°C', 120, '54 E6/ohms/m ', '!', '!', '1,3', ' ' );

Elemento[91] := TElemento. Create ( 91, 'Protactínio', 'Pa', 231.03588, '-', 7, 'f', '2, 8, 18, 32, 20, 9, 2', 'Actinídeo', '163 pm', '200 pm', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 5f3 ', 'tetragonal', 'sólido', '1841K', '4300K', '481kJ/mol', '-', 120, '47W(m-k) ', '568kJ/mol', '-', '1,5', ' ' );

Elemento[92] := TElemento. Create ( 92, 'Urânio', 'U', 238.02891, '-', 7, 'f', '2,8,18,32,21,9,2', 'Actinídeo', '-', '175 pm', '186 pm', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 5f4 ', 'ortorrômbico', 'sólido', '1405,3K', '4404K', '-', '3155m/s a 20ºC', 300, '27,5W/(m-k) ', '-', 'Paramagnético', '1,38', ' ' );

Elemento[93] := TElemento. Create ( 93, 'Netúnio', 'Np', 237, '-', 7, 'f', '2,8,18,32,22,9,2', 'Actinídeo', '155 pm', '190+/-1 pm', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 5f5 ', 'ortorrómbica, tetragonal e cúbica', 'sólido', '910K', '4273K', '1 Pa a 2194K', '-', 0, '6,3W/(m-k) ', '604,5Kj/mol', 'Paramagnético', '1,36', ' ', 'Graphics/Properties/Pictures/93' );

Elemento[94] := TElemento. Create ( 94, 'Plutônio', 'Pu', 244, '-', 7, 'f', '2,8,18,32,24,8,2', 'Actinídeo', '159 pm', '187+/-1pm', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 5f6 ', 'monoclínico', 'sólido', '912,5k', '3505k', '1Pa a 1756k', '2260m/s a 20ºc', 0, '6,74W(m-k) ', '584,7Kj/mol', 'Paramagnético', '1,28', ' ' );

Elemento[95] := TElemento. Create ( 95, 'Amerício', 'Am', 243, '-', 7, 'f', '2,8,18,32,25,8,2', 'Actinídeo', '-', '-', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 5f7 ', 'monoclínico', 'sólido', '1176c', '2607c', '-', '-', 0, '10W(m-k) ', '-', '-', '1,3', ' ' );

Elemento[96] := TElemento. Create ( 96, 'Cúrio', 'Cm', 247, '-', 7, 'f', '2,8,18,32,25,9,2', 'Actinídeo', '-', '-', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 5f8 ', 'hexagonal compacta', 'sólido', '1618k', '3383k', '-', '-', 0, '- ', '581Kj/mol', '-', '1,3', ' ' );

Elemento[97] := TElemento. Create ( 97, 'Barquélio', 'Bk', 247, '-', 7, 'f', '2,8,18,32,27,8,2', 'Actinídeo', '-', '-', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 5f9 ', '-', 'sólido', '1323k', '-', '-', '-', 0, '- ', '601kJmol', '-', '1,3', ' ' );

Elemento[98] := TElemento. Create ( 98, 'Califórnio', 'Cf', 251, '-', 7, 'f', '2,8,18,32,28,8,2', 'Actinídeo', '169 pm', '-', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 5f10 ', '-', 'sólido', '1173k', '1743k', '-', '-', 0, '- ', '608kjmol', '-', '-', ' ' );

Elemento[99] := TElemento. Create ( 99, 'Einstênio', 'Es', 252, '-', 7, 'f', '2,8,18,32,29,8,2', 'Actinídeo', '-', '-', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 5f11 ', '-', 'sólido', '860c', '-', '-', '-', 0, '10W(m-k) ', '619kjmol', '-', '1,3', ' ' );

Elemento[100] := TElemento. Create ( 100, 'Férmio', 'Fm', 257, '-', 7, 'f', '2,8,18,32,30,8,2', 'Actinídeo', '-', '-', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 5f12 ', '-', 'sólido', '1800k', '-', '-', '-', 0, '- ', '-', '-', '-', ' ' );

Elemento[101] := TElemento. Create ( 101, 'Mendelévio', 'Md', 258, '-', 7, 'f', '2,8,18,32,31,8,2', 'Actinídeo', '-', '-', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 5f13 ', '-', 'sólido', '-', '-', '-', '-', 0, '- ', '-', '-', '-', ' ' );

Elemento[102] := TElemento. Create ( 102, 'Nobélio', 'No', 259, '-', 7, 'f', '2,8,18,32,32,8,2', 'Actinídeo', '-', '-', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 5f14 ', '-', 'sólido', '-', '-', '-', '-', 0, '- ', '-', '-', '-', ' ' );

Elemento[103] := TElemento. Create ( 103, 'Laurêncio', 'Lr', 262, '3', 7, 'f', '2,8,18,32,32,8,3', 'Actinídeo', '-', '-', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 5f14 6d1 ', '-', 'sólido', '-', '-', '-', '-', 0, '- ', '-', '-', '-', ' ' );

Elemento[104] := TElemento. Create ( 104, 'Rutherfórdio', 'Rf', 267, '4', 7, 'd', '2,8,18,32,32,10,2', 'Metal em Transição', '-', '-', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 5f14 6d2 ', '-', 'sólido', '-', '-', '-', '-', 0, '- ', '-', '-', '-', ' ' );

Elemento[105] := TElemento. Create ( 105, 'Dúbnio', 'Db', 268, '5', 7, 'd', '2,8,18,32,32,11,2', 'Metal em Transição', '-', '-', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 5f14 6d3 ', '-', 'sólido', '-', '-', '-', '-', 0, '- ', '-', '-', '-', ' ' );

Elemento[106] := TElemento. Create ( 106, 'Seabórgio', 'Sg', 271, '6', 7, 'd', '2,8,18,32,32,12,2', 'Metal em Transição', '-', '-', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 5f14 6d4 ', '-', 'sólido', '-', '-', '-', '-', 0, '- ', '-', '-', '-', ' ' );

Elemento[107] := TElemento. Create ( 107, 'Bóhrio', 'Bh', 272, '7', 7, 'd', '2,8,18,32,32,13,2', 'Metal em Transição', '-', '-', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 5f14 6d5 ', '-', 'sólido', '-', '-', '-', '-', 0, '- ', '-', '-', '-', ' ' );

Elemento[108] := TElemento. Create ( 108, 'Hássio', 'Hs', 270, '8', 7, 'd', '2,8,18,32,32,14,2', 'Metal em Transição', '-', '-', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 5f14 6d6 ', '-', 'sólido', '-', '-', '-', '-', 0, '- ', '-', '-', '-', ' ' );

Elemento[109] := TElemento. Create ( 109, 'Meitnério', 'Mt', 276, '9', 7, 'd', '2,8,18,32,32,15,2', 'Metal em Transição', '-', '-', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 5f14 6d7 ', '-', 'sólido', '-', '-', '-', '-', 0, '- ', '-', '-', '-', ' ' );

Elemento[110] := TElemento. Create ( 110, 'Darmstádio', 'Ds', 281, '10', 7, 'd', '2,8,18,32,32,17,1', 'Metal em Transição', '-', '-', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 5f14 6d8 ', '-', 'sólido', '-', '-', '-', '-', 0, '- ', '-', '-', '-', ' ' );

Elemento[111] := TElemento. Create ( 111, 'Roentgênio', 'Rg', 280, '11', 7, 'd', '2,8,18,32,32,18,1', 'Metal em Transição', '-', '-', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 5f14 6d9 ', '-', 'sólido', '-', '-', '-', '-', 0, '- ', '-', '-', '-', ' ' );

Elemento[112] := TElemento. Create ( 112, 'Ununbium', 'Uub', 285, '12', 7, 'd', '2,8,18,32,32,18,2', 'Metal em Transição', '-', '-', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 5f14 6d10 ', '-', 'liquido', '-', '-', '-', '-', 0, '- ', '-', '-', '-', ' ' );

Elemento[113] := TElemento. Create ( 113, 'Ununtrio', 'Uut', 284, '13', 7, 'p', '2,8,18,32,32,18,3', 'Metal Representativo', '-', '-', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 5f14 6d10 7p1 ', '-', 'sólido', '-', '-', '-', '-', 0, '- ', '-', '-', '-', ' ' );

Elemento[114] := TElemento. Create ( 114, 'Ununquádio', 'Uuq', 289, '14', 7, 'p', '2,8,18,32,32,18,4', 'Metal Representativo', '-', '-', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 5f14 6d10 7p2 ', '-', 'sólido', '-', '-', '-', '-', 0, '- ', '-', '-', '-', ' ' );

Elemento[115] := TElemento. Create ( 115, 'Ununpentio', 'Uup', 288, '15', 7, 'p', '2,8,18,32,32,18,5', 'Metal Representativo', '-', '-', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 5f14 6d10 7p3 ', '-', 'sólido', '-', '-', '-', '-', 0, '- ', '-', '-', '-', ' ' );

Elemento[116] := TElemento. Create ( 116, 'Ununhéxio', 'Uuh', 293, '16', 7, 'p', '2,8,18,32,32,18,6', 'Metal Representativo', '-', '-', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 5f14 6d10 7p4 ', '-', 'sólido', '-', '-', '-', '-', 0, '- ', '-', '-', '-', ' ' );

Elemento[117] := TElemento. Create ( 117, 'Ununséptio', 'Uus', 294, '17', 7, 'p', '2, 8, 18, 32, 32, 18, 7', 'Halogênio', '-', '-', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 5f14 6d10 7p5 ', '-', 'sólido', '-', '-', '-', '-', 0, '- ', '-', '-', '-', ' ' );

Elemento[118] := TElemento. Create ( 118, 'Ununóctio', 'Uuo', 294, '18', 7, 'd', '2, 8, 18, 32, 32, 18, 8', 'Gases Nobres', '152 pm', '230 pm', '-', '1s2 2s2 2p6 3s2 3p6 4s2 3d10 4p6 5s2 4d10 5p6 6s2 4f14 5d10 6p6 7s2 5f14 6d10 7p6 ', '-', 'gasoso', '-', '3550+/-30k', '-', '-', 0, '- ', '-', '-', '-', ' ' );

{ Put end }

end;

// ------------------------

// JB\_Crypt

// Autor: Lucas Gaspar

// ------------------------

function Crypt( Word : String ) : String;

// String já criptografada

var Crypted : String;

// Chave anterior

var Key\_Before : Byte;

// Chave nova

var Key\_After : Byte;

// Índices

var I, J : Byte;

begin

// Define o tamanho do texto a ser criptografado

SetLength( Crypted, Length( Word ) \* Length( Word ) );

// Define a chave temporária

Key\_Before := Ord( Word[1] );

// Se a chave temporária não for divisível por 2

if not ( Key\_Before mod 2 = 0 ) then

// Faça-a divisível

Key\_Before := Key\_Before - 1;

// Divide a chave por 2.

Key\_After := Key\_Before div 2;

// Para todos os caracteres

for I := 0 to Length( Word ) - 1 do

// Para todos os caracteres

for J := 1 to Length( Word ) do

// Se o caracter criptografado não exceder o número de caracteres ASC II

if ( Ord ( Word[I] ) + Key\_After ) < ( 126 - Length( Word ) ) then

// Criptografa

Crypted[J \* Length(Word) - I] := Chr( Key\_After + Ord ( Word[I] ) + J )

// Se o caracter criptografado exceder o número de caracteres ASC II

else

// Criptografa

Crypted[J \* Length(Word) - I] := Chr( Abs( Ord( Word[I] ) - Key\_After + J ) );

// Retorna a nova string

Result := Crypted;

end;

end.

**unit** GameStoichiometric;

**interface**

**uses**

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, jpeg, ExtCtrls, StdCtrls, pngimage, DB\_Integrator, Tabela, Sound;

**type**

TCalculoEstequiometrico = **class**(TForm)

Background: TImage;

ReagenteBalanceado: TEdit;

PictureReagente: TImage;

Reagente: TLabel;

Reagente\_Massa: TLabel;

Produto: TLabel;

Produto\_Massa: TLabel;

FixNaoBalanceada: TLabel;

EquacaoNaoBalanceada: TLabel;

Fonte: TTimer;

FixReagente: TLabel;

FixProduto: TLabel;

Seta: TLabel;

PictureProduto: TImage;

ProdutoBalanceado: TEdit;

FixMassaMolar: TLabel;

MassaReagente: TLabel;

MassaProduto: TLabel;

PictureMassaReagente: TImage;

AnswerMassaReagente: TEdit;

AnswerMassaProduto: TEdit;

PictureMassaProduto: TImage;

FixFinalAnswer: TLabel;

AnswerFinalAnswer: TEdit;

PictureFinalAnswer: TImage;

CheckReagente: TImage;

CheckMassaReagente: TImage;

CheckMassaProduto: TImage;

CheckProduto: TImage;

CheckFinalAnswer: TImage;

Fix\_Pontuacao: TLabel;

AutoPontuacao: TLabel;

CheckFinished: TImage;

FadeOut: TImage;

PicturePoints: TImage;

Rodada\_X: TLabel;

X: TLabel;

Pontuacao: TLabel;

CountIntervalPoint: TTimer;

Rodada: TImage;

Label\_Rodada: TLabel;

Fix\_Rodada: TLabel;

FixBalanceada: TLabel;

Control: TTimer;

ToTable: TImage;

Forfeit: TImage;

**procedure** FonteTimer(Sender: TObject);

**procedure** SolveFont( Edit : TEdit );

**procedure** ChangeFocus( ID : Byte );

**procedure** MakeAllInOne;

**procedure** UpdateChecker(ID : Byte);

**procedure** ChangePictureToMaybe(ID : Byte);

**procedure** ChangePicture(ID : Byte);

**procedure** OnClickAnswer(Sender: TObject);

**procedure** FadeOutAll();

**procedure** UpdateFinishedSelect();

**procedure** FadeInAll();

**procedure** ShowPoint();

**procedure** CountIntervalPointTimer(Sender: TObject);

**procedure** HidePoints();

**procedure** ShowTime();

**procedure** StartNewTime();

**procedure** AnswerFinalAnswerKeyPress(Sender: TObject; **var** Key: Char);

**procedure** ContolTheControllerTimer(Sender: TObject);

**procedure** UpdatePoints( ID : Byte; Kind : Boolean);

**procedure** FormShow(Sender: TObject);

**procedure** FormKeyPress(Sender: TObject; **var** Key: Char);

**procedure** ControlTimer(Sender: TObject);

**procedure** MakeRandomWeight();

**procedure** CallMe();

**procedure** FormCreate(Sender: TObject);

**procedure** FormClose(Sender: TObject; **var** Action: TCloseAction);

**procedure** OnClick(Sender: TObject);

**procedure** OnKeyDown(Sender: TObject; **var** Key: Word; Shift: TShiftState);

**procedure** Change\_Forfeit\_Picture();

**private**

{ Private declarations }

**public**

{ Public declarations }

**var** Times : Byte;

InTest : Boolean;

Test : **Array**[1..7] **of String**;

**end**;

**var**

CalculoEstequiometrico: TCalculoEstequiometrico;

Answers : **Array**[1..5] **of** TEdit;

Checkers : **Array**[1..5] **of** TImage;

WhoFocused : Byte = 1;

CanFinish : Boolean = False;

Cursor : Byte = 1;

CanShowPoints : Boolean = False;

TimePoints : Short = 0;

TotalPoints : Integer = 0;

Count : Byte = 0;

CanShowTime : Boolean = True;

CannotupdateControl : Boolean = False;

Question : Integer = 0;

Reply : **Array**[1..5] **of String**;

Final\_Reply : **Array**[1..3] of Real;

CorrectAnswers : **Array**[1..5] of Byte;

CanStopTheGame : Boolean = false;

WaitingTime : Byte = 5;

AlreadyPlayed : **Array**[1..5] of Integer;

**implementation**

**uses** SendingPoints, Insert\_Stoichiometric, Menu;

{$R \*.dfm}

//------------------------------------------------------------------------------

// \* Altera a imagem de desistência

// Kind : Opção

//------------------------------------------------------------------------------

**procedure** TCalculoEstequiometrico.Change\_Forfeit\_Picture();

**begin**

// Executa efeito sonoro de cursor

Sound.Play\_Cursor(1);

// Altera imagem de desistência

**With** Forfeit.Picture **do**

LoadFromFile('Graphics/System/Forfeit\_' + IntToStr(Cursor) + '.png');

**end**;

**procedure** TCalculoEstequiometrico.MakeRandomWeight();

**var** I, K : Byte;

**begin**

Final\_Reply[1] := 0;

Final\_Reply[2] := 0;

**for** K := 1 **to** 2 **do**

**while** Final\_Reply[K] = 0 **do**

Final\_Reply[K] := Random(10000);

**if** I <= 5 **then**

**begin**

Reagente\_Massa.Caption := FloatToStr(Final\_Reply[1]) + ' gramas';

Produto\_Massa.Caption := '?';

Final\_Reply[3] := StrToFloat(Reply[4]) \* Final\_Reply[1] / StrToFloat(Reply[3]);

**end**

**else**

**begin**

Produto\_Massa.Caption := FloatToStr(Final\_Reply[2]) + ' gramas';

Reagente\_Massa.Caption := '?';

Final\_Reply[3] := StrToFloat(Reply[3]) \* Final\_Reply[2] / StrToFloat(Reply[4]);

**end**;

Reply[5] := FloatToStr(Final\_Reply[3]);

**end**;

**procedure** TCalculoEstequiometrico.UpdatePoints( ID : Byte; Kind : Boolean );

**var** I : Byte;

**begin**

TimePoints := 0;

Count := 0;

**if** Kind **then**

CorrectAnswers[ID] := 20

**else**

CorrectAnswers[ID] := 0;

**for** I := 1 **to** 5 **do**

TimePoints := TimePoints + CorrectAnswers[I];

AutoPontuacao.Caption := IntToStr(TimePoints + TotalPoints);

**end**;

**function** ChangeFormat\_ILoveGaspar( Text : **String** ) : **String**;

**var** I, J, K, Y: Byte;

**var** Can : Boolean;

**var** Sub : Char;

**begin**

Text := SpaceDelete(Text);

Text := Text + #32;

**for** J := 2 **to** Length(Text) **do**

**for** I := 0 **to** 9 do

**if** Text[J] = IntToStr(I) **then**

**begin**

**for** K := 0 **to** 9 **do**

**if** (Text[J + 1] = IntToStr(K)) **then**

Y := 2;

**if** (Text[J + Y] = '+') **or** (Text[J + Y] = #32) **or** (Text[J + Y] = '-') **then**

Can := True;

**if** (Text[J + Y] <> '-') **and** (Text[J - Y] <> '-') **and** (Text[J + Y] <> '+')

**and** (Text[J - Y] <> '+') **and** (Text[J + Y] <> #32)

**and** (Text[J - Y] <> #32) **and** (Text[J - Y] <> '>') **then**

Can := True;

**if** Can **then**

**begin**

**case** I **of**

0: Sub := 'Í';

1: Sub := 'Ì';

2: Sub := '¿';

3: Sub := 'À';

4: Sub := 'Á';

5: Sub := 'Â';

6: Sub := 'Ã';

7: Sub := 'Î';

8: Sub := 'Ï';

9: Sub := 'ß';

**end**;

Delete(Text,J,1);

Insert(Sub, Text, J);

Can := False;

**end**;

Y := 1;

Can := False;

**end**;

I := 1;

**while** I < Length(Text) **do**

**begin**

**if** Text[I] = '+' then

**begin**

Delete(Text, I, 1);

Insert(' + ', Text, I);

I := I + 2;

**end**

**else if** Text[I] = '-' **then**

**begin**

Delete(Text, I, 2);

Insert(' -> ', Text, I);

I := I + 3;

**end**;

I := I + 1;

**end**;

Result := Text;

**end**;

**procedure** TCalculoEstequiometrico.StartNewTime();

**var** I, K, J : Byte;

**begin**

**for** I := 1 **to** 5 **do**

**begin**

Checkers[I].Visible := False;

self.ChangePictureToMaybe(I);

Answers[I].Clear();

**end**;

Checkers[1].Visible := True;

**if** InTest **then**

**begin**

EquacaoNaoBalanceada.Caption := ChangeFormat\_ILoveGaspar(Test[1]);

Reagente.Caption := ChangeFormat\_ILoveGaspar(Test[2]);

Produto.Caption := ChangeFormat\_ILoveGaspar(Test[3]);

Reply[1] := Test[4];

Reply[2] := Test[5];

Reply[3] := Test[6];

Reply[4] := Test[7];

MassaReagente.Caption := ChangeFormat\_ILoveGaspar(Test[2]);

MassaProduto.Caption := ChangeFormat\_ILoveGaspar(Test[3]);

MakeRandomWeight();

Exit();

**end**;

I := 0;

**while** I = 0 **do**

I := Random(10);

DB\_Integrator.DataModule1.ADOQuery1.SQL.Clear;

DB\_Integrator.DataModule1.ADOQuery1.SQL.Add('Select Count(\*) as ID from' +

' Estequiometrico');

DataModule1.Adoquery1.Active := True;

DataModule1.Adoquery1.ExecSQL;

Question := 0;

**while** Question = 0 **do**

**begin**

Question := Random(DataModule1.ADOQuery1.FieldByName('ID').AsInteger);

**for** J := 1 **to** 5 **do**

**if** Question = AlreadyPlayed[J] **then**

Question := 0;

**end**;

AlreadyPlayed[Times] := Question;

// -------------------------------------------------------------------------- //

DB\_Integrator.DataModule1.ADOQuery1.SQL.Clear;

DB\_Integrator.DataModule1.ADOQuery1.SQL.Add('Select EquacaoNaoBalanceada from' +

' Estequiometrico where ID = ' + IntToStr(Question));

DataModule1.Adoquery1.Active := True;

DataModule1.Adoquery1.ExecSQL;

**with** EquacaoNaoBalanceada **do**

Caption := ChangeFormat\_ILoveGaspar(DataModule1.Adoquery1.FieldByName('' +

'EquacaoNaoBalanceada').AsString);

// -------------------------------------------------------------------------- //

DB\_Integrator.DataModule1.ADOQuery1.SQL.Clear;

DB\_Integrator.DataModule1.ADOQuery1.SQL.Add('Select ReagenteBalanceado from' +

' Estequiometrico where ID = ' + IntToStr(Question));

DataModule1.Adoquery1.Active := True;

DataModule1.Adoquery1.ExecSQL;

Reply[1] := DataModule1.ADOQuery1.FieldByName('ReagenteBalanceado').AsString;

// -------------------------------------------------------------------------- //

DB\_Integrator.DataModule1.ADOQuery1.SQL.Clear;

DB\_Integrator.DataModule1.ADOQuery1.SQL.Add('Select MassaReagente from' +

**' Estequiometrico where ID = '** + IntToStr(Question));

DataModule1.Adoquery1.Active := True;

DataModule1.Adoquery1.ExecSQL;

Reply[3] := FloatToStr(Round(RemovePoints(DataModule1.ADOQuery1.FieldByName(''

+ 'MassaReagente').AsString)));

// -------------------------------------------------------------------------- //

DB\_Integrator.DataModule1.ADOQuery1.SQL.Clear;

DB\_Integrator.DataModule1.ADOQuery1.SQL.Add('Select MassaProduto from' +

' Estequiometrico where ID = ' + IntToStr(Question));

DataModule1.Adoquery1.Active := True;

DataModule1.Adoquery1.ExecSQL;

Reply[4] := FloatToStr(Round(RemovePoints(DataModule1.ADOQuery1.FieldByName(''

+ 'MassaProduto').AsString)));

// -------------------------------------------------------------------------- //

DB\_Integrator.DataModule1.ADOQuery1.SQL.Clear;

DB\_Integrator.DataModule1.ADOQuery1.SQL.Add('Select ProdutoBalanceado from' +

' Estequiometrico where ID = ' + IntToStr(Question));

DataModule1.Adoquery1.Active := True;

DataModule1.Adoquery1.ExecSQL;

Reply[2] := DataModule1.ADOQuery1.FieldByName('ProdutoBalanceado').AsString;

// -------------------------------------------------------------------------- //

MakeRandomWeight();

// -------------------------------------------------------------------------- //

DB\_Integrator.DataModule1.ADOQuery1.SQL.Clear;

DB\_Integrator.DataModule1.ADOQuery1.SQL.Add('Select Reagente from' +

' Estequiometrico where ID = ' + IntToStr(Question));

DataModule1.Adoquery1.Active := True;

DataModule1.Adoquery1.ExecSQL;

**with** Reagente **do**

Caption := ChangeFormat\_ILoveGaspar(DataModule1.Adoquery1.FieldByName('' +

'Reagente').AsString) + ': ';

with MassaReagente do

Caption := ChangeFormat\_ILoveGaspar(DataModule1.Adoquery1.FieldByName('' +

'Reagente').AsString);

// -------------------------------------------------------------------------- //

DB\_Integrator.DataModule1.ADOQuery1.SQL.Clear;

DB\_Integrator.DataModule1.ADOQuery1.SQL.Add('Select Produto from' +

' Estequiometrico where ID = ' + IntToStr(Question));

DataModule1.Adoquery1.Active := True;

DataModule1.Adoquery1.ExecSQL;

**with** Produto **do**

Caption := ChangeFormat\_ILoveGaspar(DataModule1.Adoquery1.FieldByName('' +

'Produto').AsString) + ': ';

with MassaProduto **do**

Caption := ChangeFormat\_ILoveGaspar(DataModule1.Adoquery1.FieldByName('' +

'Produto').AsString);

**end**;

**procedure** TCalculoEstequiometrico.ShowTime();

**begin**

**if** Times = 6 **then**

Exit();

Fonte.Enabled := False;

Control.Enabled := False;

CannotUpdateControl := True;

FadeOutAll();

Rodada.Visible := True;

Fix\_Rodada.Visible := True;

**with** Label\_Rodada **do**

**if** InTest **then**

Caption := 'de Teste'

**else**

**if** Times = 5 **then**

**begin**

Caption := 'Final';

Play\_Me('FinalMatch.wav');

**end**

**else**

**begin**

Caption := IntToStr(Times);

Play\_Me('Match.wav');

**end**;

Label\_Rodada.Visible := True;

CanShowTime := False;

CountIntervalPoint.Enabled := True;

WaitingTime := 3;

StartNewTime();

**end**;

**procedure** TCalculoEstequiometrico.ShowPoint();

**var** I : Byte;

**begin**

WaitingTime := 7;

**for** I := 1 **to** 5 **do**

CorrectAnswers[I] := 0;

TotalPoints := TotalPoints + TimePoints;

CannotUpdateControl := True;

CheckFinished.Visible := False;

**if** Times = 5 **then**

**begin**

WaitingTime := 10;

Play\_Me('FinalGame.wav');

**With** X **do**

**if** InTest **then**

Caption := 'Rodada de Teste'

**else**

Caption := 'Rodada Final';

CanStopTheGame := True;

**end**

**else**

**if** InTest **then**

X.Caption := 'Rodada de Teste'

**else**

X.Caption := 'Rodada ' + IntToStr(Times);

Pontuacao.Caption := IntToStr(TotalPoints);

PicturePoints.Visible := True;

X.Visible := True;

Rodada\_X.Visible := True;

Pontuacao.Visible := True;

CountIntervalPoint.Enabled := True;

Times := Times + 1;

CanShowTime := True;

**end**;

**procedure** TCalculoEstequiometrico.HidePoints();

**var** I : Byte;

**begin**

CountIntervalPoint.Enabled := False;

Rodada.Visible := False;

Fix\_Rodada.Visible := False;

Label\_Rodada.Visible := False;

PicturePoints.Visible := False;

X.Visible := False;

Rodada\_X.Visible := False;

Pontuacao.Visible := False;

FadeInAll();

**for** I := 2 **to** 5 **do**

**with** Answers[I] do

**begin**

Visible := False;

Enabled := False;

**end**;

**end**;

**procedure** TCalculoEstequiometrico.FadeInAll();

**var** I : Byte;

**begin**

**for** I := 1 **to** 5 **do**

**with** Answers[I] **do**

**begin**

Visible := True;

Enabled := True;

**end**;

Fonte.Enabled := True;

FadeOut.Visible := False;

CheckFinished.Visible := False;

Answers[1].SetFocus();

CannotUpdateControl := False;

CanFinish := False;

**end**;

**procedure** TCalculoEstequiometrico.UpdateFinishedSelect;

**begin**

**case** Cursor **of**

1: CheckFinished.Picture.LoadFromFile('Graphics/Estequiométrico/Message/PopUp\_1.png');

2: CheckFinished.Picture.LoadFromFile('Graphics/Estequiométrico/Message/PopUp\_2.png');

**end**;

**end**;

**procedure** TCalculoEstequiometrico.FadeOutAll();

**var** I : Byte;

**begin**

Fonte.Enabled := False;

**for** I := 1 **to** 5 **do**

**with** Answers[I] **do**

**begin**

Visible := False;

Enabled := False;

**end**;

FadeOut.Visible := True;

CanFinish := True;

**end**;

**procedure** TCalculoEstequiometrico.ChangePictureToMaybe(ID: Byte);

**begin**

Checkers[ID].Picture.LoadFromFile('Graphics/Estequiométrico/Incerto.png');

**end**;

**procedure** TCalculoEstequiometrico.ContolTheControllerTimer(Sender: TObject);

**begin**

**if** (Answers[1].Focused) **or** (Answers[2].Focused) **then**

Fonte.Enabled := True

**else**

Fonte.Enabled := False;

**end**;

**procedure** TCalculoEstequiometrico.ControlTimer(Sender: TObject);

**begin**

**if** CanFinish **then**

**if** GetAsyncKeyState(VK\_Right) <> 0 **then**

**begin**

Play\_Cursor(1);

**case** Cursor **of**

1 : Cursor := 2;

2 : Cursor := 1;

**end**;

UpdateFinishedSelect();

**end**

**else if** GetAsyncKeyState(VK\_Left) <> 0 **then**

**begin**

Play\_Cursor(1);

**case** Cursor **of**

1 : Cursor := 2;

2 : Cursor := 1;

end;

UpdateFinishedSelect();

**end**;

**end**;

**procedure** TCalculoEstequiometrico.CountIntervalPointTimer(Sender: TObject);

**begin**

Count := Count + 1;

**if** Count = WaitingTime **then**

**begin**

HidePoints();

Control.Enabled := True;

Fonte.Enabled := True;

Count := 0;

**if** CanStopTheGame **then**

**begin**

**if** InTest **then**

**begin**

InsertStoichiometric.Show();

Self.Hide();

Exit();

**end**

**else**

**begin**

*// Deleta a instância e chama o Garbage Collector*

FreeAndNil(Self);

*// Cria uma instância da classe de seleção de jogos*

Inicio := TInicio.Create(Application);

*// Exibe o formulário*

Inicio.Show();

**end**;

**end**;

**end**;

**if** CanShowTime **then**

**begin**

ShowTime();

CanShowTime := False;

**end**;

**end**;

**procedure** TCalculoEstequiometrico.ChangePicture(ID: Byte);

**var** ToWhere : Boolean;

**var** Answer : **Array**[1..3] **of** Real;

**begin**

ToWhere := False;

**if** ID < 3 **then**

**begin**

**if** SpaceDelete(Answers[ID].Text) = SpaceDelete(

ChangeFormat\_ILoveGaspar(Reply[ID])) **then**

ToWhere := True

**else**

ToWhere := False;

**end**

**else**

**begin**

Answer[ID - 2] := RemovePoints(Answers[ID].Text);

**if** Round(Answer[ID - 2]) = Round(RemovePoints( Reply[ID] )) **then**

ToWhere := True

**else**

ToWhere := False;

**end**;

**if** ToWhere **then**

**begin**

Play\_Decision(2);

UpdatePoints(ID, True);

Checkers[ID].Picture.LoadFromFile('Graphics/Estequiométrico/Certo.png')

**end**

**else**

**begin**

Play\_Error();

UpdatePoints(ID, False);

Checkers[ID].Picture.LoadFromFile('Graphics/Estequiométrico/Errado.png');

**end**;

**end**;

**procedure** TCalculoEstequiometrico.UpdateChecker(ID: Byte);

**begin**

ChangePicture(ID - 1);

Checkers[ID].Visible := True;

ChangePictureToMaybe(ID);

**end**;

**procedure** TCalculoEstequiometrico.MakeAllInOne;

**begin**

Answers[1] := ReagenteBalanceado;

Answers[2] := ProdutoBalanceado;

Answers[3] := AnswerMassaReagente;

Answers[4] := AnswerMassaProduto;

Answers[5] := AnswerFinalAnswer;

Checkers[1] := CheckReagente;

Checkers[2] := CheckProduto;

Checkers[3] := CheckMassaReagente;

Checkers[4] := CheckMassaProduto;

Checkers[5] := CheckFinalAnswer;

**end**;

**procedure** TCalculoEstequiometrico.OnClick(Sender: TObject);

**begin**

Tabela.Gambz\_[1] := 0;

Fonte.Enabled := False;

Play\_Decision(1);

Tabela.PeriodicTable.Show();

Tabela.PeriodicTable.Enabled := true;

Tabela.PeriodicTable.Visible := true;

Tabela.PeriodicTable.ActiveForm := true;

Tabela.PeriodicTable.CursorIndex := Tabela.PeriodicTable.SaveCursorIndex;

Tabela.WhereTo := 2;

Self.Hide();

**end**;

**procedure** TCalculoEstequiometrico.OnClickAnswer(Sender: TObject);

**var** I : Byte;

**begin**

**for** I := 1 **to** 5 **do**

**if** Answers[I].Focused **then**

**begin**

**if** I <> WhoFocused **then**

**begin**

ChangePicture(WhoFocused);

ChangePictureToMaybe(I);

WhoFocused := I;

**end**;

**end**;

**end**;

**procedure** TCalculoEstequiometrico.OnKeyDown(Sender: TObject; **var** Key: Word;

Shift: TShiftState);

**var** I : Byte;

**begin**

// Se a tela de desistência estiver visível

**if** Forfeit.Visible **then**

**begin**

// Se a seta direita for pressionada

**if** Key = VK\_Right **then**

**begin**

**case** Cursor **of**

1: Cursor := 2;

2: Cursor := 1;

**end**;

// Altera a imagem

Change\_Forfeit\_Picture();

**end**

// Se a seta esquerda for pressionada

**else if** Key = VK\_Left **then**

**begin**

**case** Cursor **of**

1: Cursor := 2;

2: Cursor := 1;

**end**;

// Altera a imagem

Change\_Forfeit\_Picture();

**end**

// Se a tecla escape for pressionada

**else if** Key = VK\_ESCAPE **then**

**begin**

// Se continuar no jogo foi selecionado

**if** Cursor = 1 **then**

**begin**

**for** I := 2 **to** 5 **do**

**with** Answers[I] **do**

**begin**

**if** Text <> '' then

**begin**

Visible := True;

Enabled := True;

**end**;

**end**;

// Torna o primeiro campo visível, acessível e focado.

Answers[1].Visible := True;

Answers[1].Enabled := True;

Answers[1].SetFocus();

// Invisibiliza a janela de escolhas

Forfeit.Visible := False;

**end**;

**end**

// Se a tecla Enter for pressionada

**else if** Key = VK\_Return **then**

**begin**

// Executa efeito sonoro de confirmação

Sound.Play\_Decision(1);

// Se continuar no jogo foi selecionado

**if** Cursor = 1 **then**

**begin**

**for** I := 2 **to** 5 **do**

**with** Answers[I] **do**

**begin**

**if** Text <> '' **then**

**begin**

Visible := True;

Enabled := True;

**end**;

**end**;

// Torna o primeiro campo visível, acessível e focado.

Answers[1].Visible := True;

Answers[1].Enabled := True;

Answers[1].SetFocus();

// Invisibiliza a janela de escolhas

Forfeit.Visible := False;

**end**

// Se a opção de desistir foi selecionada

**else if** Cursor = 2 **then**

**begin**

**if** InTest **then**

**begin**

InsertStoichiometric.Show();

Self.Hide();

Exit();

**end**

**else**

**begin**

// Deleta a instância e chama o Garbage Collector

FreeAndNil(Self);

// Cria uma instância da classe de seleção de jogos

Inicio := TInicio.Create(Application);

// Exibe o formulário

Inicio.Show();

**end**;

**end**;

**end**;

**end**;

**end**;

**procedure** TCalculoEstequiometrico.CallMe();

**var** I : Byte;

**begin**

ShowCursor(True);

Cursor := 2;

CanStopTheGame := False;

MakeAllInOne();

ShowTime();

WhoFocused := 1;

CanFinish := False;

CanShowPoints := False;

Times := 1;

Count := 0;

CanShowTime := True;

CannotupdateControl := False;

Question := 0;

CanStopTheGame := false;

TimePoints := 0;

TotalPoints := 0;

**for** I := 1 **to** 5 **do**

CorrectAnswers[I] := 0;

**end**;

**procedure** TCalculoEstequiometrico.AnswerFinalAnswerKeyPress(Sender: TObject; **var** Key: Char);

**var** I : Byte;

**var** Check : Boolean;

**begin**

Check := False;

**if** ( Key = #13 ) **then**

**begin**

Key := #0;

**for** I := 1 **to** 5 **do**

**if** Answers[I].Focused **then**

**begin**

ChangeFocus(I + 1);

Exit();

**end**;

**end**;

**for** I := 1 **to** 5 **do**

**if** Answers[I].Focused **then**

Check := True;

**if not** Check **then**

Exit();

**if** ( Key = #27 ) then

**begin**

Key := #0;

**for** I := 1 **to** 5 **do**

**with** Answers[I] **do**

**begin**

Visible := False;

Enabled := False;

**end**;

// Executa efeito sonoro de cancelamento.

Sound.Play\_Cancel();

Forfeit.Visible := True;

Cursor := 1;

**end**;

**end**;

**procedure** TCalculoEstequiometrico.ChangeFocus( ID : Byte );

**begin**

**if** ID = 6 then

**begin**

ChangePicture(ID - 1);

FadeOutAll();

CheckFinished.Visible := True;

Cursor := 2;

Exit();

**end**;

WhoFocused := ID;

Answers[ID].Enabled := True;

Answers[ID].Visible := True;

Answers[ID].SetFocus;

UpdateChecker(ID)

**end**;

**procedure** TCalculoEstequiometrico.SolveFont(Edit: TEdit);

**var** Active : Boolean;

Active\_ : **Array**[0..9] **of** Boolean;

I : Integer;

**begin**

**if** GetAsyncKeyState(17) <> 0 **then**

Active := True

**else**

Active := False;

**for** I := 0 **to** 9 **do**

**begin**

**if** GetAsyncKeyState(48 + I) <> 0 **then**

Active\_[I] := True

**else**

Active\_[I] := False;

**if (**Active) **and** (Active\_[I]) **then**

**begin**

**case** I **of**

0: Edit.Text := Edit.Text + 'Í';

1: Edit.Text := Edit.Text + 'Ì';

2: Edit.Text := Edit.Text + '¿';

3: Edit.Text := Edit.Text + 'À';

4: Edit.Text := Edit.Text + 'Á';

5: Edit.Text := Edit.Text + 'Â';

6: Edit.Text := Edit.Text + 'Ã';

7: Edit.Text := Edit.Text + 'Î';

8: Edit.Text := Edit.Text + 'Ï';

9: Edit.Text := Edit.Text + 'ß';

**end**;

Keybd\_Event(VK\_END,0,0,0);

**end**;

**end**;

**end**;

**procedure** TCalculoEstequiometrico.FormClose(Sender: TObject;

**var** Action: TCloseAction);

**begin**

DataModule1.DataModuleDestroy(Self);

**end**;

**procedure** TCalculoEstequiometrico.FormCreate(Sender: TObject);

**begin**

PerformForm(Self);

**end**;

**procedure** TCalculoEstequiometrico.FormKeyPress(Sender: TObject; **var** Key: Char);

**begin**

if CanFinish then

**if** Key = #13 **then**

begin

**if** ( CannotUpdateControl ) **then**

Exit();

Play\_Decision(1);

CanFinish := False;

**case** Cursor **of**

1: ShowPoint();

2: FadeInAll();

**end**;

**end**;

**end**;

**procedure** TCalculoEstequiometrico.FormShow(Sender: TObject);

**begin**

Fonte.Enabled := True;

**end**;

**procedure** TCalculoEstequiometrico.FonteTimer(Sender: TObject);

**var** I : Byte;

begin

**if** ReagenteBalanceado.Focused **then**

SolveFont(ReagenteBalanceado)

**else if** ProdutoBalanceado.Focused **then**

SolveFont(ProdutoBalanceado);

**end**;

**end**.

unit Geometry;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, ExtCtrls, jpeg, StdCtrls, pngimage, DB\_Integrator, Sound;

type

//==============================================================================

// \*\* TGeometryGame

//------------------------------------------------------------------------------

// Esta classe é responsável por todo o processamento e exibição do

// jogo Geometria Molecular. Utiliza a classe DB\_Integrator como fonte

// de informações

//==============================================================================

TGeometryGame = class(TForm)

BackGround: TImage;

Elemento\_2: TLabel;

Elemento\_3: TLabel;

Elemento\_6: TLabel;

E\_2\_7: TImage;

E\_2\_1: TImage;

E\_2\_2: TImage;

E\_2\_6: TImage;

E\_2\_4: TImage;

E\_2\_3: TImage;

E\_2\_5: TImage;

E\_6\_3: TImage;

E\_6\_5: TImage;

E\_6\_4: TImage;

E\_6\_6: TImage;

E\_6\_2: TImage;

E\_6\_1: TImage;

E\_6\_7: TImage;

E\_3\_3: TImage;

E\_3\_5: TImage;

E\_3\_4: TImage;

E\_3\_6: TImage;

E\_3\_2: TImage;

E\_3\_1: TImage;

E\_3\_7: TImage;

Equacao: TLabel;

E\_1\_6: TImage;

E\_1\_2: TImage;

E\_1\_1: TImage;

E\_1\_7: TImage;

Elemento\_1: TLabel;

E\_1\_4: TImage;

E\_1\_3: TImage;

E\_1\_5: TImage;

E\_4\_6: TImage;

E\_4\_2: TImage;

E\_4\_1: TImage;

E\_4\_7: TImage;

Elemento\_4: TLabel;

E\_4\_4: TImage;

E\_4\_3: TImage;

E\_4\_5: TImage;

E\_7\_7: TImage;

E\_7\_1: TImage;

E\_7\_2: TImage;

E\_7\_6: TImage;

Elemento\_7: TLabel;

E\_7\_4: TImage;

E\_7\_3: TImage;

E\_7\_5: TImage;

E\_5\_7: TImage;

E\_5\_1: TImage;

E\_5\_2: TImage;

E\_5\_6: TImage;

Elemento\_5: TLabel;

E\_5\_4: TImage;

E\_5\_3: TImage;

E\_5\_5: TImage;

E\_2\_8: TImage;

E\_6\_8: TImage;

E\_3\_8: TImage;

E\_1\_8: TImage;

E\_7\_8: TImage;

E\_5\_8: TImage;

E\_4\_8: TImage;

Back: TImage;

Button: TImage;

Rodada: TLabel;

Tempo: TLabel;

TimeCount: TTimer;

FadeOut: TImage;

GameOverPicture: TImage;

PontuacaoPicture: TImage;

Pontuacao: TLabel;

Forfeit: TImage;

procedure FormCreate(Sender: TObject);

procedure StartGame();

procedure DrawText();

procedure MakeArray();

procedure DrawElectrons();

procedure ChangeElectronsPosition(I : Byte; J : Byte);

procedure ElectronClick(Sender: TObject);

procedure CanvasConfig();

procedure Button1Click(Sender: TObject);

procedure FormDestroy(Sender: TObject);

procedure ReturnOneTime();

procedure SaveTime( ID : Byte );

procedure NextMatch();

procedure OnMouseLeave(Sender: TObject);

procedure OnMouseEnter(Sender: TObject);

procedure OnMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure OnMouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure UpdateTimer();

procedure TimeCounter(Sender: TObject);

procedure GameOver();

procedure OnKeyPress(Sender: TObject; var Key: Char);

procedure GameWon();

procedure ResetVariables();

procedure CallMe();

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);

procedure Change\_Forfeit\_Picture();

procedure ForfeitClick(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

GeometryGame: TGeometryGame;

// Responsável pela posição dos elétrons

Vertices : Array[1..8, 1..2] of Integer;

// Grupos aceitos na Ionização

Accepted : Array[1..7] of Byte = ( 1, 2, 13, 14, 15, 16, 17 );

// Elementos Não-Metais aceitos na Ionização

Accepted\_NonMetal : Array[1..11] of Byte = (6, 7, 8, 9, 15, 16, 17, 34, 35,

53, 85);

// Elemento Metal processado

Ionica\_Metal : Byte;

// Elemento Não-Metal processado

Ionica\_NonMetal : Byte;

// Quantidade de átomos do elemento Metal processado

Atoms\_Metal : Byte;

// Quantidade de átomos do elemento Não-Metal processado

Atoms\_NonMetal : Byte;

// Símbolo dos átomos

Symbol : Array[1..7] of String;

// Quantidade de elétrons dos atomos

Electron : Array[1..7] of Byte;

// Quantidade fixa de elétrons dos átomos

FixElectron : Array[1..7] of Byte;

// Conjunto que contém Labels dos 7 elementos

Obj\_Elementos : Array[1..7] of TLabel;

// Conjunto que contém as Imagens de todos os Elétrons

Pic\_Electrons : Array[1..7] of Array[1..8] of TImage;

// Armazena dados sobre os elétrons

Pic\_Active : Array[1..7] of Array[1..8] of Boolean;

// Modo Seleção de Alvo

WhereTo : Boolean;

// Último elétron selecionado

LastElectronSelected : TImage;

// ID do último elétron selecionado

LastIDElectronSelected : Byte;

// ID do último Elemento selecionado

LastElementSelected : Byte;

// Quantidade de Jogadas

SavesCount : Integer = 0;

// Elétrons das rodadas anteriores

ElectronsPast : Array of Array of Byte;

// Equações já jogadas

AlreadyPlayed : Array[1..5] of **String**;

// Tempo em minutos

Minute : Byte = 1;

// Tempo em segundos

Second : Byte = 1;

// Cursor

Cursor : Byte;

implementation

uses Menu;

{$R \*.dfm}

//------------------------------------------------------------------------------

// \* Define a posição dos elétrons na tela

//------------------------------------------------------------------------------

procedure DefVertices();

begin

Vertices[1, 1] := 304 - 322;

Vertices[1, 2] := 296 - 275;

Vertices[2, 1] := 304 - 322;

Vertices[2, 2] := 325 - 275;

Vertices[3, 1] := 5;

Vertices[3, 2] := 309 - 275;

Vertices[4, 1] := 5;

Vertices[4, 2] := 330 - 275;

Vertices[5, 1] := 5;

Vertices[5, 2] := 288 - 275;

Vertices[6, 1] := 322 - 322;

Vertices[6, 2] := 346 - 275;

Vertices[7, 1] := 322 - 322;

Vertices[7, 2] := 275 - 275;

Vertices[8, 1] := 350 - 322;

Vertices[8, 2] := 346 - 275;

end;

//------------------------------------------------------------------------------

// \* Cria um Vetor para automatizar desenhos na tela

//------------------------------------------------------------------------------

procedure TGeometryGame.MakeArray;

begin

//----------

// Elementos

//----------

Obj\_Elementos[1] := Elemento\_1;

Obj\_Elementos[2] := Elemento\_2;

Obj\_Elementos[3] := Elemento\_3;

Obj\_Elementos[4] := Elemento\_4;

Obj\_Elementos[5] := Elemento\_5;

Obj\_Elementos[6] := Elemento\_6;

Obj\_Elementos[7] := Elemento\_7;

//----------

// Elétrons

//----------

// Elemento 1

Pic\_Electrons[1][1] := E\_1\_1;

Pic\_Electrons[1][2] := E\_1\_2;

Pic\_Electrons[1][3] := E\_1\_3;

Pic\_Electrons[1][4] := E\_1\_4;

Pic\_Electrons[1][5] := E\_1\_5;

Pic\_Electrons[1][6] := E\_1\_6;

Pic\_Electrons[1][7] := E\_1\_7;

Pic\_Electrons[1][8] := E\_1\_8;

// Elemento 2

Pic\_Electrons[2][1] := E\_2\_1;

Pic\_Electrons[2][2] := E\_2\_2;

Pic\_Electrons[2][3] := E\_2\_3;

Pic\_Electrons[2][4] := E\_2\_4;

Pic\_Electrons[2][5] := E\_2\_5;

Pic\_Electrons[2][6] := E\_2\_6;

Pic\_Electrons[2][7] := E\_2\_7;

Pic\_Electrons[2][8] := E\_2\_8;

// Elemento 3

Pic\_Electrons[3][1] := E\_3\_1;

Pic\_Electrons[3][2] := E\_3\_2;

Pic\_Electrons[3][3] := E\_3\_3;

Pic\_Electrons[3][4] := E\_3\_4;

Pic\_Electrons[3][5] := E\_3\_5;

Pic\_Electrons[3][6] := E\_3\_6;

Pic\_Electrons[3][7] := E\_3\_7;

Pic\_Electrons[3][8] := E\_3\_8;

// Elemento 4

Pic\_Electrons[4][1] := E\_4\_1;

Pic\_Electrons[4][2] := E\_4\_2;

Pic\_Electrons[4][3] := E\_4\_3;

Pic\_Electrons[4][4] := E\_4\_4;

Pic\_Electrons[4][5] := E\_4\_5;

Pic\_Electrons[4][6] := E\_4\_6;

Pic\_Electrons[4][7] := E\_4\_7;

Pic\_Electrons[4][8] := E\_4\_8;

// Elemento 5

Pic\_Electrons[5][1] := E\_5\_1;

Pic\_Electrons[5][2] := E\_5\_2;

Pic\_Electrons[5][3] := E\_5\_3;

Pic\_Electrons[5][4] := E\_5\_4;

Pic\_Electrons[5][5] := E\_5\_5;

Pic\_Electrons[5][6] := E\_5\_6;

Pic\_Electrons[5][7] := E\_5\_7;

Pic\_Electrons[5][8] := E\_5\_8;

// Elemento 6

Pic\_Electrons[6][1] := E\_6\_1;

Pic\_Electrons[6][2] := E\_6\_2;

Pic\_Electrons[6][3] := E\_6\_3;

Pic\_Electrons[6][4] := E\_6\_4;

Pic\_Electrons[6][5] := E\_6\_5;

Pic\_Electrons[6][6] := E\_6\_6;

Pic\_Electrons[6][7] := E\_6\_7;

Pic\_Electrons[6][8] := E\_6\_8;

// Elemento 7

Pic\_Electrons[7][1] := E\_7\_1;

Pic\_Electrons[7][2] := E\_7\_2;

Pic\_Electrons[7][3] := E\_7\_3;

Pic\_Electrons[7][4] := E\_7\_4;

Pic\_Electrons[7][5] := E\_7\_5;

Pic\_Electrons[7][6] := E\_7\_6;

Pic\_Electrons[7][7] := E\_7\_7;

Pic\_Electrons[7][8] := E\_7\_8;

end;

//------------------------------------------------------------------------------

// \* Obtém a quantidade de elétrons da camada de valência

// ID : Núnero Atômico do Elemento

//------------------------------------------------------------------------------

function GetElectron( ID : Byte ) : Byte;

var I : Byte;

begin

// Procura o grupo do elemento para determinar seus elétrons

for I := 1 to 7 do

if Elemento[ID].Get\_Elemento(4) = IntToStr(Accepted[I]) then

// Retorna o grupo

Result := I;

end;

//------------------------------------------------------------------------------

// \* Substitui números naturais por números moleculares

// Equacao : Equação molecular

//------------------------------------------------------------------------------

function FormatEquation( Equacao : **String**) : **String**;

var I, J : Byte;

Sub : **String**;

begin

// Obtém o tamanho da String

for I := 1 to Length(Equacao) + 1 do

begin

// Checa número por número

for J := 0 to 9 do

// Se o caracter X da String for um número

if Equacao[I] = IntToStr(J) then

begin

// Substitue-o pelo número atômico correspondente

case J of

0: Sub := 'Í';

1: Sub := '';

2: Sub := '¿';

3: Sub := 'À';

4: Sub := 'Á';

5: Sub := 'Â';

6: Sub := 'Ã';

7: Sub := 'Î';

8: Sub := 'Ï';

9: Sub := 'ß';

end;

// Deleta o número

Delete( Equacao, I, 1 );

// Insere o número atômico

Insert( Sub, Equacao, I );

// Retorna a Equação com números atômicos

Result := Equacao;

end;

end;

end;

//------------------------------------------------------------------------------

// \* Cria a equação

//------------------------------------------------------------------------------

function DefMakeEquation() : **String**;

begin

Result := Elemento[Ionica\_Metal].Get\_Elemento(2) + IntToStr(Atoms\_Metal) +

Elemento[Ionica\_NonMetal].Get\_Elemento(2) + IntToStr(Atoms\_NonMetal);

end;

//------------------------------------------------------------------------------

// \* Algoritmo gerador de Ligações Iônicas

//------------------------------------------------------------------------------

procedure DefEquation();

var Rand, I : Byte;

Checked : Boolean;

ValenceMetal : Byte;

ValenceNonMetal : Byte;

begin

// Atribuição às Variáveis

Rand := 0;

//-------------------

// Escolha do Metal

//-------------------

// Enquanto o número atômico for 0

while Rand = 0 do

begin

// Randomiza o número atômico

Rand := Random(88);

// Se número atômico for diferente de 0

if Rand <> 0 then

begin

// Checa se o elemento faz parte dos grupos I,II ou III da tabela

for I := 1 to 3 do

if ( Elemento[Rand].Get\_Elemento(4) = IntToStr(Accepted[I]) ) and

// Checa se o elemento não é o Boro

( Elemento[Rand].Get\_Elemento(0) <> IntToStr(5) ) then

begin

// Armazena o número atômico

Ionica\_Metal := Rand;

// Permite parar o loop primário

Checked := True;

// Para o loop secundário

Break;

end;

// Se é permitido parar o Loop primário

if Checked then

// Para o loop primário

Break

// Se não for permitido para o Loop primário

else

// Repete o loop

Rand := 0;

end;

end;

// Fim da escolha do Metal

// Atribuição às Variáveis

Rand := 0;

Checked := False;

//-------------------

// Escolha do Ametal

//-------------------

// Enquanto o número atômico for 0

while Rand = 0 do

begin

// Randomiza o número atômico

Rand := Random(86);

// Se número atômico for diferente de 0

if Rand <> 0 then

begin

// Checa se o elemento é um não-metal permitido para a ionização

for I := 1 to 11 do

if Elemento[Rand].Get\_Elemento(0) = IntToStr(Accepted\_NonMetal[I]) then

begin

// Armazena o Não-Metal

Ionica\_NonMetal := Rand;

// Permite a quebra do loop principal

Checked := True;

// Quebra o loop secundário

Break;

end;

// Se é permitido parar o loop

if Checked then

// Para o loop principal

Break

// Se não é permitido

else

// Repete o loop

Rand := 0;

end;

end;

// Fim da escolha do Ametal

// Atribuição dos Elétrons da Câmada de Valência dos elementos

ValenceMetal := GetElectron(Ionica\_Metal);

ValenceNonMetal := GetElectron(Ionica\_NonMetal);

//---------------------

// Formação de Equação

//---------------------

// Se a união de 1 átomo do Metal e 1 do Não-Metal concluir o octeto

if ValenceMetal + ValenceNonMetal = 8 then

begin

// Define 1 átomo para o não-metal

Atoms\_NonMetal := 1;

// Define 1 átomo para o Metal

Atoms\_Metal := 1;

end

// Caso não seja, processa a regra da tesoura

else

begin

// Átomos do Não-Metal é igual aos câtions do Metal

Atoms\_NonMetal := ValenceMetal;

// Átomos do Metal é igual aos ânios do Não-Metal

Atoms\_Metal := 8 - ValenceNonMetal;

end;

// Para as 5 rodadas possíveis

for I := 1 to 5 do

// Se a equação selecionada já tiver sido jogada

if AlreadyPlayed[I] = FormatEquation(DefMakeEquation) then

// Refaz o processo de seleção de equação

DefEquation();

// Para as 5 rodadas possíveis

for I := 1 to 5 do

// Se a rodada não tiver sido jogada

if AlreadyPlayed[I] = '' then

begin

// Adiciona a equação a esta rodada

AlreadyPlayed[I] := FormatEquation(DefMakeEquation);

// Para o loop

Break;

end;

end;

//------------------------------------------------------------------------------

// \* Processamento para facilitar a organização das informações na tela

//------------------------------------------------------------------------------

procedure DefGeometry();

var I : Byte;

begin

// Caso a quantidade de átomos de Não-metal seja menor que as do de Metal

if Atoms\_NonMetal < Atoms\_Metal then

begin

// Até a quantidade de Átomos de Não-Metal

for I := 1 to Atoms\_NonMetal do

begin

// Armazena o símbolo do átomo Não-Metal

Symbol[I] := Elemento[Ionica\_NonMetal].Get\_Elemento(2);

// Armazena a quantidade de elétrons do átomo Não-Metal

Electron[I] := GetElectron(Ionica\_NonMetal);

// Armazena a quantidade original de elétrons do átomo Não-Metal

FixElectron[I] := GetElectron(Ionica\_NonMetal);

end;

// Até a quantidade de átomos de Metal + Não-Metal

for I := Atoms\_NonMetal + 1 to Atoms\_Metal + Atoms\_NonMetal do

begin

// Armazena o símbolo do átomo Metal

Symbol[I] := Elemento[Ionica\_Metal].Get\_Elemento(2);

// Armazena a quantidade de elétrons do átomo Metal

Electron[I] := GetElectron(Ionica\_Metal);

// Armazena a quantidade original de elétrons do átomo Metal

FixElectron[I] := GetElectron(Ionica\_Metal);

end;

end

// Caso a quantidade de átomos de Metal seja menor que as do de Não-Metal

else

begin

// Até a quantidade de átomos de Metal

for I := 1 to Atoms\_Metal do

begin

// Armazena o símbolo do átomo Metal

Symbol[I] := Elemento[Ionica\_Metal].Get\_Elemento(2);

// Armazena a quantidade de elétrons do átomo Metal

Electron[I] := GetElectron(Ionica\_Metal);

// Armazena a quantidade original de elétrons do átomo Metal

FixElectron[I] := GetElectron(Ionica\_Metal);

end;

// Até a quantidade de Átomos de Não-Metal + Metal

for I := Atoms\_Metal + 1 to Atoms\_NonMetal + Atoms\_Metal do

begin

// Armazena o símbolo do átomo Não-Metal

Symbol[I] := Elemento[Ionica\_NonMetal].Get\_Elemento(2);

// Armazena a quantidade de elétrons do átomo Não-Metal

Electron[I] := GetElectron(Ionica\_NonMetal);

// Armazena a quantidade original de elétrons do átomo Não-Metal

FixElectron[I] := GetElectron(Ionica\_NonMetal);

end;

end;

// Desenha o elemento na tela

GeometryGame.DrawText();

// Desenha os elétrons na tela

GeometryGame.DrawElectrons();

end;

//------------------------------------------------------------------------------

// \* Desenha os elementos na tela

//------------------------------------------------------------------------------

procedure TGeometryGame.DrawText();

var I : Byte;

begin

// Para todos os átomos de Metal e de Não-Metal

for I := 1 to Atoms\_NonMetal + Atoms\_Metal do

begin

// Atribui o símbolo do átomo ao Label

Obj\_Elementos[I].Caption := Symbol[I];

// Torna o Label visível

Obj\_Elementos[I].Visible := True;

// Torna o Label acessível

Obj\_Elementos[I].Enabled := True;

end;

// Atribui a equação a um Label

Equacao.Caption := FormatEquation(DefMakeEquation);

end;

//------------------------------------------------------------------------------

// \* Conversão de String para PWideChar

// Text : Texto qualquer

//------------------------------------------------------------------------------

function StrToWide( Text : **String** ) : PWideChar;

begin

// Retorna String como WideChar

Result := PWideChar(WideString(Text));

end;

//------------------------------------------------------------------------------

// \* Salva o estado da rodada

// ID : Elemento

//------------------------------------------------------------------------------

procedure TGeometryGame.SaveTime( ID : Byte );

begin

// Define tamanho do vetor multidimensional de acordo com as rodadas

SetLength(ElectronsPast,SavesCount, 3);

// Organiza informações na Array para que possa retornar rodadas

ElectronsPast[SavesCount - 1][0] := ID;

ElectronsPast[SavesCount - 1][1] := LastElementSelected;

ElectronsPast[SavesCount - 1][2] := LastIDElectronSelected;

end;

//------------------------------------------------------------------------------

// \* Evento ao clicar em um elétron

// Sender : Componente responsável pela execução

//------------------------------------------------------------------------------

procedure TGeometryGame.ElectronClick(Sender: TObject);

var I, J, M : Byte;

var Controller : Boolean;

var FileName : PWideChar;

var Least : Byte;

begin

// Atribuição às variáveis

Controller := True;

// Se estiver selecionando o alvo

if WhereTo then

begin

// Procura o Elemento que foi clicado

for I := 1 to Atoms\_Metal + Atoms\_NonMetal do

begin

// Bloqueia clique ao último elemento selecionado

if Sender = Obj\_Elementos[LastElementSelected] then

Exit();

if Sender = Obj\_Elementos[I] then

begin

// Caso não haja 8 elétrons na camada de Valência

if Electron[I] < 8 then

begin

// Soma 1 ao contador de jogadas

SavesCount := SavesCount + 1;

// Salva a jogada anterior à rodada

SaveTime(I);

// Define o nome do arquivo

FileName := StrToWide('Saves/' + IntToStr(SavesCount) + '.gaspar');

// Salva o Bitmap da jogada

Back.Picture.SaveToFile(FileName);

// Proteção para o arquivo ( Torna-o oculto )

SetFileAttributes(FileName, 2);

// Altera a posição do novo elétron

ChangeElectronsPosition(I, Electron[I] + 1);

// Para todos os elétrons

for M := 1 to 8 do

// Checa se o elemento ganhou um cátion

if Pic\_Active[I][M] then

begin

// Define a posição do ânion doado

Least := M;

// Atribui como cátion não recebido

Pic\_Active[I][M] := False;

// Finaliza o loop

Break;

end

// Se o elemento não ganhou um cátion

else

// Define a posição do novo ânion como o próximo elétron

Least := Electron[I] + 1;

// Com o Novo Elétron

With Pic\_Electrons[I][Least] do

begin

// Altera sua imagem

Picture.LoadFromFile('Graphics/Geometry/Little Ball\_Came.png');

// Desenha uma linha do elétron selecionado até o elétron alvo

Back.Canvas.LineTo(Left,Top);

// Torna-o inacessível

Enabled := False;

// Torna-o visível

Visible := True;

end;

// Com o último elétron selecionado

With LastElectronSelected do

begin

// Altera sua imagem

Picture.LoadFromFile('Graphics/Geometry/Little Ball\_Went.png');

// Torna-o inacessível

Enabled := False;

end;

// Adiciona 1 elétron ao controle da camada de Valência

Electron[I] := Electron[I] + 1;

// Subtrai 1 elétron ao controle da camada de Valência

Electron[LastElementSelected] := Electron[LastElementSelected] - 1;

// Permite o elemento anterior ser acessado

LastElementSelected := 0;

// Executa efeito sonoro

Play\_Se('Okay.wav');

end

// Se houver 8 elétrons no elemento

else

// Sai do método

Exit();

// Desativa o modo seleção de alvo

WhereTo := False;

// Para o loop

Break;

end;

end;

end

else

// Procura o elétron selecionado

for I := 1 to 7 do

for J := 1 to 7 do

if Sender = Pic\_Electrons[I][J] then

begin

// Com o elétron selecionado

With Pic\_Electrons[I][J] do

begin

// Altera sua imagem

Picture.LoadFromFile('Graphics/Geometry/Little Ball\_To.png');

// Move o ponto de desenho do bitmap para a posição do elétron

Back.Canvas.MoveTo(Left, Top);

end;

// Armazena o elétron selecionado

LastElectronSelected := Pic\_Electrons[I][J];

// Informa que o elétron é removido

Pic\_Active[I][J] := True;

// Armazena o ID do elétron selecionado

LastIDElectronSelected := J;

// Armazena o elemento selecionado

LastElementSelected := I;

// Ativa o modo seleção de alvo

WhereTo := True;

// Para o loop

Break;

end;

// Checa se todas as ligações corretas foram feitas

for I := 1 to Atoms\_Metal + Atoms\_NonMetal do

// Se a quantidade de elétrons não for 0 nem 8

if ( Electron[I] <> 0 ) and ( Electron[I] <> 8 ) then

// Não permite o jogo ser finalizado

Controller := False;

// Se for permitido finalizar o jogo

if Controller then

// Próxima rodada

NextMatch();

end;

//------------------------------------------------------------------------------

// \* Desenha os elétrons na tela

//------------------------------------------------------------------------------

procedure TGeometryGame.DrawElectrons;

var I, J : Byte;

begin

// Para todos os elétrons dos átomos

for I := 1 to Atoms\_NonMetal + Atoms\_Metal do

for J := 1 to Electron[I] do

begin

// Altera a posição do Elétron

ChangeElectronsPosition(I, J);

// Altera a imagem do Elétron

Pic\_Electrons[I][J].Picture.LoadFromFile('Graphics/Geometry/Little Ball.png');

// Torna o elétron visível

Pic\_Electrons[I][J].Visible := True;

// Torna o elétron acessível

Pic\_Electrons[I][J].Enabled := True;

end;

end;

//------------------------------------------------------------------------------

// \* Altera posição dos elétrons

// I : Índice da Linha

// J : Índice da Coluna

//------------------------------------------------------------------------------

procedure TGeometryGame.ChangeElectronsPosition(I : Byte; J : Byte);

begin

// Se for necessário que se some a largura do Elemento

if ( J = 3 ) or ( J = 4 ) or ( J = 5 ) then

// Soma a largura do elemento à posição X do Elétron

Pic\_Electrons[I][J].Left := Obj\_Elementos[I].Left + Vertices[J,1]

+ Obj\_Elementos[I].Width

else

// Não soma a largura do elemento à posição X do Elétron

Pic\_Electrons[I][J].Left := Obj\_Elementos[I].Left + Vertices[J,1];

// Define a nova posição Y do elétron

Pic\_Electrons[I][J].Top := Obj\_Elementos[I].Top + Vertices[J,2];

end;

//------------------------------------------------------------------------------

// \* Inicializa o jogo

//------------------------------------------------------------------------------

procedure TGeometryGame.StartGame();

begin

// Inicializa o armazenador de posições dos elétrons

DefVertices();

// Inicializa o construtor de conjuntos para Elétrons e Elementos

MakeArray();

// Gera a equação para a Ionização

DefEquation();

// Organiza os objetos na tela

DefGeometry();

end;

//------------------------------------------------------------------------------

// \* Processo chamado a cada segundo

// Sender : Componente responsável pela execução

//------------------------------------------------------------------------------

procedure TGeometryGame.TimeCounter(Sender: TObject);

begin

// Atualiza o relógio

UpdateTimer();

end;

//------------------------------------------------------------------------------

// \* Processo de atualização do tempo

//------------------------------------------------------------------------------

procedure TGeometryGame.UpdateTimer();

begin

// Diminui os segundos em 1.

Second := Second - 1;

// Se o tempo esgotar

if ( Second = 0 ) and ( Minute = 0 ) then

begin

// Chama processo de Game Over

GameOver();

// Atribui novo tempo ao relógio

Tempo.Caption := '00:00';

// Altera a cor do relógio para preto

Tempo.Font.Color := RGB(0,0,0);

// Sai do método

Exit();

end;

// Se os segundos for 0

if Second = 0 then

begin

// Passa para o minuto antecessor

Second := 59;

Minute := Minute - 1;

end;

// Com o label que exibe o tempo

With Tempo do

begin

// Se os minutos fore inferiores ao 10

if Minute < 10 then

begin

// Se os segundos forem inferiores ao 10

if Second < 10 then

// Adiciona um 0 antes do número do segundo e do minuto

Caption := '0' + IntToStr(Minute) + ':0' + IntToStr(Second)

else

// Adiciona um 0 antes do minuto

Caption := '0' + IntToStr(Minute) + ':' + IntToStr(Second);

end

else

// Se os segundos forem inferiores ao 10

if Second < 10 then

// Adiciona um 0 antes do número do segundo

Caption := IntToStr(Minute) + ':0' + IntToStr(Second)

else

// Adiciona somente os pontos de separação

Caption := IntToStr(Minute) + ':' + IntToStr(Second);

// Se o tempo do jogador for inferior aos 31 segundos

if ( Minute = 0 ) and ( Second < 31 ) then

// Se a fonte que exibe o tempo for preta

if Font.Color = RGB(0,0,0) then

begin

// Mude-a para vermelha

Font.Color := RGB(255,0,0);

end

// Se a fonte for vermelha

else

// Mude-a para preto

Font.Color := RGB(0,0,0);

end;

end;

//------------------------------------------------------------------------------

// \* Processo de fim de jogo

//------------------------------------------------------------------------------

procedure TGeometryGame.GameOver();

begin

// Reproduz efeito musical de derrota

Play\_Me('Derrota1.wav');

// Desativa o contador de tempo

TimeCount.Enabled := False;

// Torna a imagem de fade out visível

FadeOut.Visible := True;

// Torna a imagem de game over visível

GameOverPicture.Visible := True;

end;

//------------------------------------------------------------------------------

// \* Clique ao botão

//------------------------------------------------------------------------------

procedure TGeometryGame.Button1Click(Sender: TObject);

begin

// Retornar uma jogada

ReturnOneTime();

end;

//------------------------------------------------------------------------------

// \* Retorno de jogadas

//------------------------------------------------------------------------------

procedure TGeometryGame.ReturnOneTime();

var A, B, I : Byte;

begin

// Se a quantidade de jogadas para retornar for nula

if SavesCount < 1 then

// Sai do método

Exit();

// Não permite selecionar um alvo

WhereTo := False;

// Altera a imagem do elétron

LastElectronSelected.Picture.LoadFromFile('Graphics/Geometry/Little Ball.png');

// Executa efeito sonoro de retorno de jogada

Play\_Se('Back.wav');

// Muda o formato do arquivo para .bmp

RenameFile('Saves/' + IntToStr(SavesCount) + '.gaspar',

'Saves/' + IntToStr(SavesCount) + '.bmp');

// Carrega a imagem da jogada anterior

Back.Picture.LoadFromFile('Saves/' + IntToStr(SavesCount) + '.bmp');

// Deleta a imagem da jogada anterior

DeleteFile('Saves/' + IntToStr(SavesCount) + '.bmp');

// Reconfigura o Bitmap

CanvasConfig();

// Limpeza de código ( Atribuição dos elétrons da rodada anterior )

A := ElectronsPast[SavesCount - 1][0];

B := ElectronsPast[SavesCount - 1][1];

// Retorna elétrons à rodada anterior

Electron[A] := Electron[A] - 1;

Electron[B] := Electron[B] + 1;

// Para todos os elétrons do elemento

for I := 1 to 8 do

// Se o elétron for diferente da valência natural do elemento

if ( I > Electron[A] ) and ( I > FixElectron[A] ) and not ( Pic\_Active[A][I] ) then

// Torna o elétron invisível

Pic\_Electrons[A][I].Visible := False;

// Com o Cátion: Substituir cátion por ânion

with Pic\_Electrons[B][ElectronsPast[SavesCount - 1][2]] do

begin

// Altera sua imagem

Picture.LoadFromFile('Graphics/Geometry/Little Ball.png');

// Define como cátion não recebido

Pic\_Active[B][ElectronsPast[SavesCount - 1][2]] := False;

// Torna o elétron acessível

Enabled := True;

end;

// Diminui 1 na quantidade de jogadas para retornar

SavesCount := SavesCount - 1;

end;

//------------------------------------------------------------------------------

// \* Configura atributos de Canvas do Bitmap para desenho

//------------------------------------------------------------------------------

procedure TGeometryGame.CanvasConfig();

begin

// Define o tamanho do pincel de desenho

Back.Canvas.Pen.Width := 4;

// Define a cor do pincel de desenho

Back.Canvas.Pen.Color := RGB(200,100,100);

end;

//------------------------------------------------------------------------------

// \* Processo para iniciar nova rodada

//------------------------------------------------------------------------------

procedure TGeometryGame.NextMatch();

var I, J : Byte;

begin

// Para todos os elementos

for I := 1 to 7 do

begin

// Torne-os invisíveis

Obj\_Elementos[I].Visible := False;

// Para todos os elétrons

for J := 1 to 8 do

// Torne-os invisíveis

Pic\_Electrons[I][J].Visible := False;

end;

// Para todas as rodadas jogadas

for I := 1 to SavesCount do

// Retorna uma rodada

ReturnOneTime();

// Para o máxima de rodadas possível

for I := 1 to 5 do

// Se ainda não estiver na última rodada

if AlreadyPlayed[I] = '' then

begin

Play\_Se('Clean.wav');

// Inicia nova rodada

CallMe();

// Altera a exibição da atual rodada

Rodada.Caption := OrdinalToCardinal(I);

// Sai do Loop

Exit();

end;

// Processa o fim de jogo ao ter ganhado

GameWon();

end;

//------------------------------------------------------------------------------

// \* Processo executado ao apertar qualquer tecla dentro do formulário

// Sender : Componente responsável pela execução

// Key : Tecla

//------------------------------------------------------------------------------

procedure TGeometryGame.OnKeyPress(Sender: TObject; var Key: Char);

begin

// Se o enter for pressionado

if Key = #13 then

begin

// Se o game over não tiver sido chamado

if not FadeOut.Visible then

// Sai do método

Exit();

// Se o game over tiver sido chamado

if GameOverPicture.Visible then

begin

// Fecha a janela

Self.Close();

// Cria uma instância da classe de seleção de jogos

Inicio := TInicio.Create(Application);

// Exibe o formulário

Inicio.Show();

end

// Se a pontuação é mostrada

else

begin

// Fecha a janela

Self.Close();

{ Código pré-pronto para o envio de pontuação

// Cria uma instância para a janela de envio de pontuação

SendingPoints := TSendingPoints.Create(Application);

SendingPoints.FormCreate(Application);

SendingPoints.Visible(); } { Código a seguir deste método será

eliminado }

// Cria uma instância da classe de seleção de jogos

Inicio := TInicio.Create(Application);

// Exibe o formulário

Inicio.Show();

end;

end

// Se a tecla escape for pressionada

else if Key = #27 then

begin

// Executa efeito sonoro de cancelamento

Sound.Play\_Cancel();

// Torna a imagem de escolha de desistência visível

Forfeit.Visible := True;

// Reseta o cursor

Cursor := 1;

// Para a contagem

TimeCount.Enabled := False;

end;

end;

//------------------------------------------------------------------------------

// \* Altera a imagem de desistência

// Kind : Opção

//------------------------------------------------------------------------------

procedure TGeometryGame.Change\_Forfeit\_Picture();

begin

// Executa efeito sonoro de cursor

Sound.Play\_Cursor(1);

// Altera imagem de desistência

With Forfeit.Picture do

LoadFromFile('Graphics/System/Forfeit\_' + IntToStr(Cursor) + '.png');

end;

//------------------------------------------------------------------------------

// \* Processo executado ao apertar o botão do mouse sobre o botão

// Sender : Componente responsável pela execução

// Action : Tipo de ação

// Button : Tipo de botão ( Esquerdo, Direito e etc .. )

// Shift : Estado da tecla CTRL

// X : Posição X do Cursor dentro do Botão

// Y : Posição Y do Cursor dentro do Botão

//------------------------------------------------------------------------------

procedure TGeometryGame.OnMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

begin

// Altera a imagem do botão

Self.Button.Picture.LoadFromFile('Graphics/Geometry/Desfazer Jogada\_3.png');

end;

//------------------------------------------------------------------------------

// \* Processo executado ao adentrar as coordenadas do botão

// Sender : Componente responsável pela execução

//------------------------------------------------------------------------------

procedure TGeometryGame.OnMouseEnter(Sender: TObject);

begin

// Altera a imagem do botão

Button.Picture.LoadFromFile('Graphics/Geometry/Desfazer Jogada\_2.png');

end;

//------------------------------------------------------------------------------

// \* Processo executado ao sair dos limites das coordenadas do botão

// Sender : Componente responsável pela execução

//------------------------------------------------------------------------------

procedure TGeometryGame.OnMouseLeave(Sender: TObject);

begin

// Altera a imagem do botão

Button.Picture.LoadFromFile('Graphics/Geometry/Desfazer Jogada\_1.png');

end;

//------------------------------------------------------------------------------

// \* Processo executado ao soltar o botão esquerdo do mouse sobre o botão

// Sender : Componente responsável pela execução

// Button : Tipo de botão ( Esquerdo, Direito e etc .. )

// Shift : Estado da tecla CTRL

// X : Posição X do Cursor dentro do Botão

// Y : Posição Y do Cursor dentro do Botão

//------------------------------------------------------------------------------

procedure TGeometryGame.OnMouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

begin

// Altera a imagem do botão

Self.Button.Picture.LoadFromFile('Graphics/Geometry/Desfazer Jogada\_1.png');

end;

//------------------------------------------------------------------------------

// \* Processo ao vencer o jogo

//------------------------------------------------------------------------------

procedure TGeometryGame.GameWon();

begin

// Reproduz efeito musical de vitória

Play\_Me('Vitoria1.wav');

// Desativa o atualizador de relógio

TimeCount.Enabled := False;

// Torna a imagem escura visível

FadeOut.Visible := True;

// Se os minutos forem maiores que 0

if Minute > 0 then

// Determina a pontuação de acordo com o tempo ( Segundos ao total )

Pontuacao.Caption := IntToStr( Minute \* 60 + Second )

else

// Determina a pontuação de acordo com o tempo ( Segundos )

Pontuacao.Caption := IntToStr( Second );

// Torna a janela de pontuação visível

PontuacaoPicture.Visible := True;

// Torna a pontuação visível

Pontuacao.Visible := True;

end;

//------------------------------------------------------------------------------

// \* Deleta arquivos

// Til : Deletar arquivos 'até...'

//------------------------------------------------------------------------------

procedure DeleteFiles( Til : Short );

var I : Short;

begin

// Deleta todos os arquivos de save

for I := 1 to Til do

DeleteFile('Saves/' + IntToStr(I) + '.gaspar');

end;

//------------------------------------------------------------------------------

// \* Criação do formulário

// Sender : Componente responsável pela execução

//------------------------------------------------------------------------------

procedure TGeometryGame.FormCreate(Sender: TObject);

var I : Byte;

begin

// Adapta formulário às configurações padrões do programa

PerformForm(Self);

end;

//------------------------------------------------------------------------------

// \* Processo ao fechar o formulário

// Sender : Componente responsável pela execução

// Action : Tipo de ação

//------------------------------------------------------------------------------

procedure TGeometryGame.ForfeitClick(Sender: TObject);

begin

end;

procedure TGeometryGame.FormClose(Sender: TObject; var Action: TCloseAction);

begin

// Deletar Cache

DeleteFiles( SavesCount );

end;

//------------------------------------------------------------------------------

// \* Inicialização do jogo

//------------------------------------------------------------------------------

procedure TGeometryGame.CallMe();

begin

// Configura o Bitmap

CanvasConfig();

// Inicia o jogo

StartGame();

end;

//------------------------------------------------------------------------------

// \* Destruição do formulário

// Sender : Componente responsável pela execução

//------------------------------------------------------------------------------

procedure TGeometryGame.FormDestroy(Sender: TObject);

begin

// Deletar Cache

DeleteFiles( SavesCount );

// Deletar Variáveis

ResetVariables();

// Finaliza o aplicativo

DataModule1.DataModuleDestroy(Self);

end;

procedure TGeometryGame.FormKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

begin

if Key = VK\_F4 then

Key := 0;

// Se a tela de desistência estiver visível

if Forfeit.Visible then

begin

// Se a seta direita for pressionada

if Key = VK\_Right then

begin

case Cursor of

1: Cursor := 2;

2: Cursor := 1;

end;

// Altera a imagem

Change\_Forfeit\_Picture();

end

// Se a seta esquerda for pressionada

else if Key = VK\_Left then

begin

case Cursor of

1: Cursor := 2;

2: Cursor := 1;

end;

// Altera a imagem

Change\_Forfeit\_Picture();

end

// Se a tecla escape for pressionada

else if Key = VK\_ESCAPE then

begin

// Torna a contagem ativa

TimeCount.Enabled := True;

// Invisibiliza a janela de escolhas

Forfeit.Visible := False;

end

// Se a tecla Enter for pressionada

else if Key = VK\_Return then

begin

// Executa efeito sonoro de confirmação

Sound.Play\_Decision(1);

// Se continuar no jogo foi selecionado

if Cursor = 1 then

begin

// Torna a contagem ativa

TimeCount.Enabled := True;

// Invisibiliza a janela de escolhas

Forfeit.Visible := False;

end

// Se a opção de desistir foi selecionada

else if Cursor = 2 then

begin

// Fecha a janela

Self.Close();

// Cria uma instância da classe de seleção de jogos

Inicio := TInicio.Create(Application);

// Exibe o formulário

Inicio.Show();

end;

end;

end;

end;

//------------------------------------------------------------------------------

// \* Reseta todas as variáveis da memória ram ( Delphi lixo não tem GC )

//------------------------------------------------------------------------------

procedure TGeometryGame.ResetVariables();

var I, J : Byte;

begin

// Reseta todas as variáveis ( Exceto as com valores constantes )

Ionica\_Metal := 0;

Ionica\_NonMetal := 0;

Atoms\_Metal := 0;

Atoms\_NonMetal := 0;

// Para todos os elementos

for I := 1 to 7 do

begin

Symbol[I] := '';

Electron[I] := 0;

FixElectron[I] := 0;

// Para todos os elétrons

for J := 1 to 8 do

Pic\_Active[I][J] := False;

end;

WhereTo := False;

LastElectronSelected := Nil;

LastIDElectronSelected := 0;

LastElementSelected := 0;

SavesCount := 0;

ElectronsPast := Nil;

// Para todas as jogadas

for I := 1 to 5 do

AlreadyPlayed[I] := '';

Minute := 1;

Second := 1;

end;

end.

unit Insert\_Stoichiometric;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, pngimage, jpeg, ExtCtrls, StdCtrls, DB\_Integrator, GameStoichiometric,

Sound;

type

//==============================================================================

// \*\* TInsertStoichiometric

//------------------------------------------------------------------------------

// Esta classe é responsável pela inserção de equações, no banco de dados, para

// o jogo Cálculos estequiométricos.

//==============================================================================

TInsertStoichiometric = class(TForm)

BackGround: TImage;

PreShow: TImage;

Save: TImage;

Clean: TImage;

Exit: TImage;

EquacaoNaoBalanceada: TEdit;

Reagente: TEdit;

Produto: TEdit;

ReagenteBalanceado: TEdit;

ProdutoBalanceado: TEdit;

MassaReagente: TEdit;

MassaProduto: TEdit;

procedure OnKeyPress(Sender: TObject; var Key: Char);

procedure OnMouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure OnFormCreate(Sender: TObject);

procedure ActionPreShow();

procedure ActionSave();

procedure ActionClean();

procedure ActionExit();

procedure FormDestroy(Sender: TObject);

function CorrectBoxes() : Boolean;

procedure OnMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure BackGroundClick(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

InsertStoichiometric: TInsertStoichiometric;

implementation

uses Menu;

{$R \*.dfm}

// Conjunto de Campos para inserção

var Boxes : Array[1..7] of TEdit;

// Conjunto de Nomes para campos de Inserção

var Names : Array[1..7] of String;

//------------------------------------------------------------------------------

// \* Checa se os campos para inserção têm informações corretas

//------------------------------------------------------------------------------

function TInsertStoichiometric.CorrectBoxes() : Boolean;

var I : Byte;

J : Byte;

S : String;

T : String;

A : Array[1..5] of Boolean;

B : Boolean;

N : Array[1..2] of Real;

U : String;

Y : Boolean;

V : String;

begin

// Para todos os campos de inserção

for I := 1 to 7 do

begin

// Se for todos os campos menos os de Massas molares e Resposta Final

if I < 6 then

// Para todos os elementos químicos

for J := 1 to 118 do

// Se os campos de inserção realmente contiverem elementos químicos

if Pos(Elemento[J].Get\_Elemento(2), Boxes[I].Text) > 0 then

begin

// Se o campo de inserção for o de equação-não-balanceada

if ( I = 1 ) and ( A[1] ) then

// Não autoriza criar mensagem sobre falta de informação do campo

B := True;

// Campos de inserção que estão corretos

A[I] := True;

end;

// Se os campos de inserção não contiverem nada inserido

if Boxes[I].Text = '' then

begin

// Se o campo de inserção não for os de massas molares ou resposta final

if I < 6 then

// Campos de Inserção que estão corretos

A[I] := True;

// Armazena os campos que precisam de modificação

S := S + ^m + Names[I];

end;

end;

// Se o equação-não-balanceada estiver incorreto

if not ( A[1] ) or not ( B ) then

// Se o equação-não-balanceada contiver algo escrito

if ( Boxes[1].Text <> '') then

begin

// Se Equação-Não-Balanceada não contiver '->'

if pos('->', SpaceDelete(Boxes[1].Text)) > 0 then

// Adiciona aviso sobre dados incorretos

else T := T + ^m + Names[1];

// Se já não contiver avisos sobre dados incorretos

if T = '' then

// Adiciona aviso sobre dados incorretos

T := T + ^m + Names[1];

end;

// Para os campos Reagente, Produto, ReagenteBalanceado e ProdutoBalanceado

for I := 2 to 5 do

// Se os dados do campo estiver incorreto

if not A[I] then

// Adiciona aviso sobre campos incorretos

T := T + ^m + Names[I];

// Se há campos que não foram preenchidos

if S <> '' then

// Adiciona a mensagem

S := 'Os seguintes campos não foram preenchidos:' + ^m + S;

// Se há campos com dados incorretos

if T <> '' then

// Adiciona a mensagem

T := ^m + 'Os seguintes campos não parecem possuir ' +

'realmente equações químicas:' + ^m + T + ^m;

// Para os campos de Massas Molares

for I := 6 to 7 do

// Armazena seus valores ( já com vírgulas ).

N[I - 5] := RemovePoints(Boxes[I].Text);

// Para valores de massas molares

for I := 1 to 2 do

// Se forem menores ou iguais a zero e conterem algo inserido

if ( N[I] <= 0 ) and ( Boxes[I + 5].Text <> '' ) then

// Armazena campo como errado

U := U + ^m + Names[I + 5];

// Se houve um campo de massa molar incorreto

if U <> '' then

// Adiciona a mensagem

U := ^m + 'Os seguintes campos devem ser preenchidos apenas com números' +

' positivos:' + ^m + U;

// Limpa comandos SQL

DB\_Integrator.DataModule1.ADOQuery1.SQL.Clear;

// Adiciona comando SQL ( Obtém a quantidade de equações )

DB\_Integrator.DataModule1.ADOQuery1.SQL.Add('Select Count(\*) as ID from' +

' Estequiometrico');

// Ativa o SQL

DataModule1.Adoquery1.Active := True;

// Executa comandos SQL

DataModule1.Adoquery1.ExecSQL;

// Para todos as equações

for I := 1 to DataModule1.ADOQuery1.FieldByName('ID').AsInteger do

begin

// Limpa comandos SQL

DataModule1.ADOQuery1.SQL.Clear();

// Adiciona comando SQL ( Para todas Equações-Não-Balanceadas )

DataModule1.ADOQuery1.SQL.Add('Select EquacaoNaoBalanceada from Estequi' +

'ometrico where ID = ' + IntToStr(I));

// Ativa SQL

DataModule1.ADOQuery1.Active := True;

// Executa SQL

DataModule1.ADOQuery1.ExecSQL;

// Se já houver a equação não balanceada

if SpaceDelete(Boxes[1].Text) = SpaceDelete(

DataModule1.ADOQuery1.FieldByName('EquacaoNaoBalanceada').AsString) then

// Permite informar o erro

Y := True;

end;

// Se permitir informar o erro de equação já inserida

if Y then

// Adiciona mensagem

V := ^m + 'A Equação-não-Balanceada já está presente no banco de dados.';

// Se não houver nenhum erro nos campos de inserção

if ( S = '' ) and ( T = '' ) and ( N[1] <> 0 ) and ( N[2] <> 0 ) and ( U = '' )

and not ( Y ) then

begin

// Permite armazenar no banco de dados e/ou testar as informações inseridas

Result := True;

// Executa efeito sonoro de bem sucedido

Play\_Se('Okay.wav');

end

// Se houver erros nas informações inseridas

else

begin

// Executa efeito sonoro de falta de informação

Play\_SE('Miss.wav');

// Exibe mensagens de erros

ShowMessage(S + T + U + V);

// Não permite armazenar nem testar as informações

Result := False

end;

end;

//------------------------------------------------------------------------------

// \* Ação de Pré-Visualizar

//------------------------------------------------------------------------------

procedure TInsertStoichiometric.ActionPreShow();

var I : Byte;

begin

// Se as informações inseridas estiverem corretas

if CorrectBoxes then

begin

// Destrói a instância ( do jogo ) e chama o Garbage Collector

FreeAndNil(CalculoEstequiometrico);

// Reinstancia o jogo de Cálculos estequiométricos

CalculoEstequiometrico := TCalculoEstequiometrico.Create(Application);

// Diz ao jogo que o modo teste será executado

GameStoichiometric.CalculoEstequiometrico.InTest := True;

{ Passa as informações para o jogo }

for I := 1 to 5 do

CalculoEstequiometrico.Test[I] := Boxes[I].Text;

for I := 6 to 7 do

CalculoEstequiometrico.Test[I] := FloatToStr(RemovePoints(Boxes[I].Text));

{ Informações passadas }

// Cria o formulário do jogo

CalculoEstequiometrico.CallMe();

// Informa à instância que haverá somente a rodada de teste

GameStoichiometric.CalculoEstequiometrico.Times := 5;

// Exibe o formulário do jogo

CalculoEstequiometrico.Show();

// Esconde o formulário atual

Self.Hide();

end;

end;

//------------------------------------------------------------------------------

// \* Ação de Salvar Informações no Banco de Dados

//------------------------------------------------------------------------------

procedure TInsertStoichiometric.ActionSave();

var I : Byte;

begin

// Se as informações inseridas estiverem corretas

if CorrectBoxes then

begin

// ------------------------------------------------------------------------ //

// Limpa comandos SQL

DB\_Integrator.DataModule1.ADOQuery1.SQL.Clear;

// Cria parâmetros para inserção

DB\_Integrator.DataModule1.ADOQuery1.SQL.Add('Insert into Estequiometrico' +

' (EquacaoNaoBalanceada, ReagenteBalanceado, ProdutoBalanceado, Reagente' +

', Produto, MassaReagente, MassaProduto)' +

' Values (:A, :B, :C, :D, :E, :F, :G)');

{ Para os campos Equação-Não-Balanceada, Reagente, Produto,

Reagente Balanceado e Produto Balanceado }

for I := 0 to 4 do

// Insera-os no banco de dados

DataModule1.ADOQuery1.Parameters[I].Value := Boxes[I + 1].Text;

// Insere os campos de massas molares

DataModule1.ADOQuery1.Parameters[5].Value := StrToFloat(Boxes[6].Text);

DataModule1.ADOQuery1.Parameters[6].Value := StrToFloat(Boxes[7].Text);

// Executa comandos SQL

DataModule1.ADOQuery1.ExecSQL();

// ------------------------------------------------------------------------ //

end;

end;

procedure TInsertStoichiometric.BackGroundClick(Sender: TObject);

begin

end;

//------------------------------------------------------------------------------

// \* Destruição do formulário

// Sender : Componente responsável pela execução

//------------------------------------------------------------------------------

procedure TInsertStoichiometric.FormDestroy(Sender: TObject);

begin

// Finaliza o programa

DataModule1.DataModuleDestroy(Self);

end;

//------------------------------------------------------------------------------

// \* Ação de limpar campos de inserção

//------------------------------------------------------------------------------

procedure TInsertStoichiometric.ActionClean();

var I : Byte;

begin

// Executa efeito sonoro de limpeza

Play\_Se('Clean.wav');

// Para todos os campos

for I := 1 to 7 do

// Limpe-os

Boxes[I].Clear();

// Manda cursor ao Equação-Não-Balanceada

Boxes[1].SetFocus();

end;

//------------------------------------------------------------------------------

// \* Ação de sair do Inserção de equações

//------------------------------------------------------------------------------

procedure TInsertStoichiometric.ActionExit();

begin

// Exibe o Menu Principal

Inicio.Show();

// Fecha a Janela atual

Self.Close();

end;

//------------------------------------------------------------------------------

// \* Criação do Formulário

// Sender : Componente responsável pela execução

//------------------------------------------------------------------------------

procedure TInsertStoichiometric.OnFormCreate(Sender: TObject);

begin

// Adapta formulário aos demais

PerformForm(Self);

// Cria um conjunto de campos de inserção

Boxes[1] := EquacaoNaoBalanceada;

Boxes[2] := Reagente;

Boxes[3] := Produto;

Boxes[4] := ReagenteBalanceado;

Boxes[5] := ProdutoBalanceado;

Boxes[6] := MassaReagente;

Boxes[7] := MassaProduto;

// Nomeia os campos de inserção

Names[1] := 'Equação não Balanceada';

Names[2] := 'Reagente';

Names[3] := 'Produto';

Names[4] := 'Reagente Balanceado';

Names[5] := 'Produto Balanceado';

Names[6] := 'Massa molar do Reagente Balanceado';

Names[7] := 'Massa molar do Produto Balanceado'

end;

//------------------------------------------------------------------------------

// \* Processo ao pressionar uma tecla

// Sender : Componente responsável pela execução

// Key : Tecla pressionada

//------------------------------------------------------------------------------

procedure TInsertStoichiometric.OnKeyPress(Sender: TObject; var Key: Char);

begin

// Se a tecla 'Enter' for pressionada

if Key = #13 then

begin

// Impede Bip

Key := #0;

// Executa efeito sonoro de próximo

Play\_Move();

// Move o cursor até o próximo Campo

Perform(WM\_nextdlgctl,0,0);

end

// Se a tecla 'Escape' for pressionada

else if Key = #27 then

// Impede Bip

Key := #0;

end;

//------------------------------------------------------------------------------

// \* Processo ao pressionar o botão do mouse

// Sender : Componente responsável pela execução

// Action : Tipo de ação

// Button : Tipo de botão ( Esquerdo, Direito e etc .. )

// Shift : Estado da tecla CTRL

// X : Posição X do Cursor dentro do Botão

// Y : Posição Y do Cursor dentro do Botão

//------------------------------------------------------------------------------

procedure TInsertStoichiometric.OnMouseDown(Sender: TObject;

Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

begin

// Se o componente executador for o botão 'Pré-Visualizar'

if Sender = PreShow then

// Altera a imagem do botão

PreShow.Picture.LoadFromFile('Graphics/Insert/Pre\_2.png')

// Se o componente executador for o botão 'Salvar'

else if Sender = Save then

// Altera a imagem do botão

Save.Picture.LoadFromFile('Graphics/Insert/Save\_2.png')

// Se o componente executador for o botão 'Limpar'

else if Sender = Clean then

// Altera a imagem do botão

Clean.Picture.LoadFromFile('Graphics/Insert/Clean\_2.png')

// Se o componente executador for o botão 'Sair'

else if Sender = Exit then

// Altera a imagem do botão

Exit.Picture.LoadFromFile('Graphics/Insert/Exit\_2.png');

end;

//------------------------------------------------------------------------------

// \* Processo ao deixar de pressionar o botão do mouse

// Sender : Componente responsável pela execução

// Button : Tipo de botão ( Esquerdo, Direito e etc .. )

// Shift : Estado da tecla CTRL

// X : Posição X do Cursor dentro do Botão

// Y : Posição Y do Cursor dentro do Botão

//------------------------------------------------------------------------------

procedure TInsertStoichiometric.OnMouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

begin

// Se o componente executador for o botão 'Pré-Visualizar'

if Sender = PreShow then

begin

// Altera a imagem do botão

PreShow.Picture.LoadFromFile('Graphics/Insert/Pre\_1.png');

// Executa o processo de Pré-Visualizar

ActionPreShow()

end

// Se o componente executador for o botão 'Salvar'

else if Sender = Save then

begin

// Altera a imagem do botão

Save.Picture.LoadFromFile('Graphics/Insert/Save\_1.png');

// Executa o processo de Salvar

ActionSave()

end

// Se o componente executador for o botão 'Limpar'

else if Sender = Clean then

begin

// Altera a imagem do botão

Clean.Picture.LoadFromFile('Graphics/Insert/Clean\_1.png');

// Executa o processo de Limpar

ActionClean()

end

// Se o componente executador for o botão 'Sair'

else if Sender = Exit then

begin

// Executa efeito sonoro de cancelamento

Play\_Cancel();

// Altera a imagem do botão

Exit.Picture.LoadFromFile('Graphics/Insert/Exit\_1.png');

// Executa o processo de Sair

ActionExit();

end;

end;

end.

unit JogoDistribuicao;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, ExtCtrls, pngimage, jpeg, DB\_Integrator, Sound;

type

TEletronicDistribution = class(TForm)

MoveLabel: TTimer;

BackGround: TImage;

Element: TLabel;

Number: TLabel;

PopUp: TImage;

ToChoose: TTimer;

ToConfirm: TTimer;

Points: TLabel;

Tempo: TLabel;

CountTime: TTimer;

WinCount: TTimer;

\_1s: TLabel;

\_2p: TLabel;

\_2s: TLabel;

\_3d: TLabel;

\_3p: TLabel;

\_3s: TLabel;

\_4d: TLabel;

\_4f: TLabel;

\_4p: TLabel;

\_4s: TLabel;

\_5d: TLabel;

\_5f: TLabel;

\_5p: TLabel;

\_5s: TLabel;

\_6d: TLabel;

\_6p: TLabel;

\_6s: TLabel;

\_7p: TLabel;

\_7s: TLabel;

FastTimer: TTimer;

Table: TImage;

FinalPoint: TLabel;

Forfeit: TImage;

procedure FormCreate(Sender: TObject);

procedure \_\_\_1sOnMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure \_\_1sOnMouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure MoveLabelTimer(Sender: TObject);

procedure TableClick(Sender: TObject);

procedure SetDirection;

procedure Back( Index : Byte );

procedure FinishPress;

procedure GameBegins( TLevel : Byte );

procedure ToChooseTimer(Sender: TObject);

procedure ChangePicture;

procedure DisposeSelect;

procedure ToConfirmTimer(Sender: TObject);

procedure RemovePoints( Value : Integer );

procedure UpdateTime;

procedure CountTimeTimer(Sender: TObject);

procedure ShowResult( Result : Boolean );

procedure WinCountTimer(Sender: TObject);

procedure Choices();

procedure SelectChoice( Index : Byte );

procedure ElementClick(Sender: TObject);

procedure \_TableOnMouseEnter(Sender: TObject);

procedure \_TableOnMouseLeave(Sender: TObject);

procedure \_TableOnMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure \_TableOnMouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure Change\_Forfeit\_Picture();

procedure OnKeyPress(Sender: TObject; var Key: Char);

procedure OnKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);

procedure ForfeitClick(Sender: TObject);

private

{ Private declarations }

public

Scene : String;

end;

var

EletronicDistribution: TEletronicDistribution;

Niveis : Array[1..19] of TLabel;

SubNiveis : Array[1..19] of Byte =

( 2 { 1s }, 2 { 2s }, 6 { 2p }, 2 { 3s },

6 { 3p }, 2 { 4s }, 10 { 3d }, 6 { 4p },

2 { 5s }, 10 { 4d }, 6 { 5p }, 2 { 6s },

14 { 4f }, 10 { 5d }, 6 { 6p }, 2 { 7s },

14 { 5f }, 10 { 6d }, 6 { 7p } );

MouseActive : Byte;

Dir : Byte = 1;

Inside : Array[1..19] of TLabel;

Count : Byte = 0;

Answer : String = '';

ElementRand : Byte = 1;

Cursor : Byte;

Point : Integer = 1000;

Second : Byte = 0;

Minute : Byte = 0;

WinCoun : Byte = 0;

Level : Byte;

ButtonPressing : Boolean;

Forfeit\_Cursor : Byte;

implementation

uses Tabela, Menu, Propriedades;

{$R \*.dfm}

procedure TEletronicDistribution.SelectChoice(Index: Byte);

var I : Integer;

begin

Play\_Decision(1);

if Index = 3 then

begin

ToChoose.Enabled := false;

ToConfirm.Enabled := false;

Inicio := TInicio.Create(Application);

Inicio.FormCreate(Application);

Inicio.Show();

Cursor := 1;

Table.Enabled := true;

Self.Hide();

end

else if Index = 2 then

begin

if Self <> Nil then

begin

Self.Free();

Self := TEletronicDistribution.Create(Application);

end;

ToChoose.Enabled := true;

ToConfirm.Enabled := true;

Cursor := 1;

Table.Enabled := True;

Self.Show;

end

else if Index = 1 then

begin

I := Level;

if Self <> Nil then

begin

FreeAndNil(EletronicDistribution);

EletronicDistribution := TEletronicDistribution.Create(Application);

end;

PopUp.Visible := False;

Cursor := 1;

Points.Caption := '1000';

Table.Enabled := True;

Self.Show;

GameBegins( I );

end;

end;

procedure TEletronicDistribution.Choices;

begin

PopUp.Picture.LoadFromFile('Graphics/Distribuição/Escolha/Retry.png');

Scene := 'Menu';

Cursor := 1;

ToChoose.Enabled := True;

ToConfirm.Enabled := True;

end;

procedure TEletronicDistribution.ShowResult;

var I : Byte;

begin

for I := 1 to 19 do

Niveis[I].Enabled := false;

CountTime.Enabled := False;

Table.Enabled := False;

WinCount.Enabled := True;

if Result then

begin

PopUp.Picture.LoadFromFile('Graphics/Distribuição/Result/Won.png');

Play\_Me('Victory1.wav');

end

else

begin

PopUp.Picture.LoadFromFile('Graphics/Distribuição/Result/Lost.png');

Play\_Me('GameOver2.wav');

end;

PopUp.Visible := True;

end;

procedure TEletronicDistribution.UpdateTime;

begin

Second := Second + 1;

if Second mod 60 = 0 then

begin

Second := 0;

Minute := Minute + 1;

end;

With Tempo do

begin

if Minute < 10 then

begin

if Second < 10 then

Caption := '0' + IntToStr(Minute) + ':0' + IntToStr(Second)

else

Caption := '0' + IntToStr(Minute) + ':' + IntToStr(Second);

end

else

if Second < 10 then

Caption := IntToStr(Minute) + ':0' + IntToStr(Second)

else

Caption := IntToStr(Minute) + ':' + IntToStr(Second);

end;

end;

procedure TEletronicDistribution.WinCountTimer(Sender: TObject);

begin

if ( WinCoun = 8 )then

begin

Choices();

FinalPoint.Visible := False;

FinalPoint.Caption := '';

self.Enabled := False;

end;

if ( WinCoun = 1 ) then

begin

FinalPoint.Visible := True;

FinalPoint.Caption := IntToStr( Point div ( ( Minute \* 60 ) + Second ) \* ElementRand );

end;

WinCoun := WinCoun + 1;

end;

procedure TEletronicDistribution.RemovePoints(Value: Integer);

begin

Point := Point - Value;

if Point < 0 then

Point := 0;

With Points do

Caption := IntToStr(Point);

end;

procedure TEletronicDistribution.DisposeSelect;

begin

PopUp.Visible := False;

ToChoose.Enabled := False;

ToConfirm.Enabled := False;

end;

procedure TEletronicDistribution.ElementClick(Sender: TObject);

begin

Play\_Decision(3);

PeriodicTable.Hide();

WindowPropertie.WentFromGame( ElementRand );

WindowPropertie.ShowModal;

end;

procedure TEletronicDistribution.GameBegins(TLevel: Byte);

var Index, I, J, K : Byte;

begin

Level := TLevel;

Index := 0;

while Index = 0 do

case Level of

1: begin Index := Random(40); K := 20; end;

2: begin Index := Random(80); K := 5; end;

3: begin Index := Random(118); K := 3; end;

end;

ElementRand := Index;

Number.Caption := IntToStr(Index);

Element.Caption := Elemento[Index].Get\_Elemento(2);

CountTime.Enabled := True;

for I := 1 to 19 do

if Index - SubNiveis[I] >= 0 then

begin

Niveis[I].Caption := Niveis[I].Caption + IntToStr(SubNiveis[I]);

Niveis[I].Visible := True;

Niveis[I].Enabled := True;

Index := Index - SubNiveis[I];

end

else if Index > 0 then

begin

Niveis[I].Caption := Niveis[I].Caption + IntToStr(Index);

Niveis[I].Visible := True;

Niveis[I].Enabled := True;

Index := 0;

end

else

begin

if ( Random(K) = 1 ) then

begin

J := 0;

while J = 0 do

J := Random(SubNiveis[I]);

Niveis[I].Caption := Niveis[I].Caption + IntToStr(J);

Niveis[I].Visible := True;

Niveis[I].Enabled := True;

end

end;

end;

procedure TEletronicDistribution.FinishPress;

begin

ClipCursor(Nil);

SetDirection();

MouseActive := 0;

end;

procedure TEletronicDistribution.Back(Index: Byte);

var I : Byte;

begin

for I := Index to Count do

Inside[I] := Inside[I + 1];

end;

procedure TEletronicDistribution.SetDirection;

var I : Byte;

begin

Answer := '';

if Count > 0 then

for I := 1 to Count do

begin

if ( I > 1 ) then

begin

if ( Length(Inside[I - 1].Caption) > 3 ) then

Inside[I].Left := Inside[I - 1].Left + 55

else

Inside[I].Left := Inside[I - 1].Left + 45;

end

else

Inside[I].Left := 130;

Answer := Answer + Inside[I].Caption + ' ';

end;

if Answer = Elemento[ElementRand].Get\_Elemento(12) then

ShowResult( true );

end;

procedure TEletronicDistribution.ForfeitClick(Sender: TObject);

begin

end;

procedure TEletronicDistribution.FormClose(Sender: TObject;

var Action: TCloseAction);

begin

DataModule1.DataModuleDestroy(Self);

end;

//------------------------------------------------------------------------------

// \* Altera a imagem de desistência

// Kind : Opção

//------------------------------------------------------------------------------

procedure TEletronicDistribution.Change\_Forfeit\_Picture();

begin

Sound.Play\_Cursor(1);

With Forfeit.Picture do

LoadFromFile('Graphics/System/Forfeit\_' + IntToStr(Forfeit\_Cursor) + '.png');

end;

procedure TEletronicDistribution.FormCreate(Sender: TObject);

var

I: Integer;

begin

Niveis[1] := \_1s;

Niveis[2] := \_2s;

Niveis[3] := \_2p;

Niveis[4] := \_3s;

Niveis[5] := \_3p;

Niveis[6] := \_4s;

Niveis[7] := \_3d;

Niveis[8] := \_4p;

Niveis[9] := \_5s;

Niveis[10] := \_4d;

Niveis[11] := \_5p;

Niveis[12] := \_6s;

Niveis[13] := \_4f;

Niveis[14] := \_5d;

Niveis[15] := \_6p;

Niveis[16] := \_7s;

Niveis[17] := \_5f;

Niveis[18] := \_6d;

Niveis[19] := \_7p;

for I := 1 to 19 do

begin

Niveis[I].Enabled := false;

Niveis[I].Visible := false;

end;

MouseActive := 0;

Cursor := 1;

PerformForm(Self);

for I := 1 to 19 do

Inside[I] := TLabel.Create(Inside[I]);

With PopUp do

begin

Left := ( Self.Width div 2 ) - ( Width div 2 );

Top := ( Self.Height div 2 ) - ( Height div 2 );

end;

With ToChoose do

Enabled := False;

With ToConfirm do

Enabled := False;

With Points do

Caption := IntToStr(Point);

With Tempo do

Caption := '00:00';

With Table do

Enabled := False;

Scene := 'InGame';

Second := 0;

Minute := 0;

ElementRand := 0;

Point := 1000;

for I := 1 to 19 do

Inside[I] := TLabel.Create(Inside[I]);

Count := 0;

WinCoun := 0;

end;

procedure TEletronicDistribution.TableClick(Sender: TObject);

begin

Tabela.WhereTo := 1;

Play\_Decision(1);

Tabela.PeriodicTable.Show;

Tabela.PeriodicTable.Enabled := true;

Tabela.PeriodicTable.Visible := true;

Tabela.PeriodicTable.ActiveForm := true;

Tabela.PeriodicTable.CursorIndex := Tabela.PeriodicTable.SaveCursorIndex;

EletronicDistribution.Hide();

end;

procedure TEletronicDistribution.ToChooseTimer(Sender: TObject);

begin

if Windows.GetAsyncKeyState(VK\_DOWN) <> 0 then

begin

Play\_Cursor(1);

if Cursor < 3 then

Cursor := Cursor + 1

else

Cursor := 1;

ChangePicture();

end

else if Windows.GetAsyncKeyState(VK\_UP) <> 0 then

begin

Play\_Cursor(1);

if Cursor > 1 then

Cursor := Cursor - 1

else

Cursor := 3;

ChangePicture();

end;

end;

procedure TEletronicDistribution.ToConfirmTimer(Sender: TObject);

begin

if Windows.GetAsyncKeyState(VK\_Return) <> 0 then

begin

Play\_Decision(1);

if Scene = 'InGame' then

begin

GameBegins( Cursor );

Table.Enabled := True;

DisposeSelect;

end

else if Scene = 'Menu' then

SelectChoice( Cursor );

end;

end;

procedure TEletronicDistribution.ChangePicture();

begin

if Scene = 'InGame' then

case Cursor of

1: PopUp.Picture.LoadFromFile('Graphics/Dificuldade/Facil.png');

2: PopUp.Picture.LoadFromFile('Graphics/Dificuldade/Medio.png');

3: PopUp.Picture.LoadFromFile('Graphics/Dificuldade/Dificil.png');

end

else if Scene = 'Menu' then

case Cursor of

1: PopUp.Picture.LoadFromFile('Graphics/Distribuição/Escolha/Retry.png');

2: PopUp.Picture.LoadFromFile('Graphics/Distribuição/Escolha/Choose.png');

3: PopUp.Picture.LoadFromFile('Graphics/Distribuição/Escolha/Other.png');

end;

end;

procedure TEletronicDistribution.CountTimeTimer(Sender: TObject);

begin

UpdateTime();

end;

procedure TEletronicDistribution.MoveLabelTimer(Sender: TObject);

var I : Byte;

begin

if MouseActive <> 0 then

begin

Niveis[MouseActive].Left := Mouse.CursorPos.X - 10;

Niveis[MouseActive].Top := Mouse.CursorPos.Y - 10;

if Dir = 1 then

begin

Table.Picture.LoadFromFile('Graphics/Distribuição/TableDark.png');

BackGround.Picture.LoadFromFile('Graphics/Distribuição/AllDark.jpg');

Dir := 2;

end;

end

else

if Dir = 2 then

begin

Table.Picture.LoadFromFile('Graphics/Distribuição/Table.png');

BackGround.Picture.LoadFromFile('Graphics/Distribuição/All.jpg');

Dir := 1;

end;

end;

procedure TEletronicDistribution.OnKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

begin

// Se a tela de desistência estiver visível

if Forfeit.Visible then

begin

// Se a seta direita for pressionada

if Key = VK\_Right then

begin

case Forfeit\_Cursor of

1: Forfeit\_Cursor := 2;

2: Forfeit\_Cursor := 1;

end;

// Altera a imagem

Change\_Forfeit\_Picture();

end

// Se a seta esquerda for pressionada

else if Key = VK\_Left then

begin

case Forfeit\_Cursor of

1: Forfeit\_Cursor := 2;

2: Forfeit\_Cursor := 1;

end;

// Altera a imagem

Change\_Forfeit\_Picture();

end

// Se a tecla escape for pressionada

else if Key = VK\_ESCAPE then

begin

// Invisibiliza a janela de escolhas

Forfeit.Visible := False;

// Regulariza o tempo

CountTime.Enabled := True;

end

// Se a tecla Enter for pressionada

else if Key = VK\_Return then

begin

Sound.Play\_Decision(1);

// Se continuar no jogo foi selecionado

if Forfeit\_Cursor = 1 then

begin

// Invisibiliza a janela de escolhas

Forfeit.Visible := False;

// Regulariza o tempo

CountTime.Enabled := True;

end

// Se a opção de desistir foi selecionada

else if Forfeit\_Cursor = 2 then

begin

ToChoose.Enabled := false;

ToConfirm.Enabled := false;

Inicio := TInicio.Create(Application);

Inicio.FormCreate(Application);

Inicio.Show();

Cursor := 1;

Table.Enabled := true;

Self.Hide();

end;

end;

end;

end;

procedure TEletronicDistribution.OnKeyPress(Sender: TObject; var Key: Char);

begin

// Se a tecla escape for pressionada

if Key = #27 then

begin

Sound.Play\_Cancel;

// Torna a imagem de escolha de desistência visível

Forfeit.Visible := True;

// Reseta o cursor

Forfeit\_Cursor := 1;

// Paraliza o tempo

CountTime.Enabled := False;

end;

end;

procedure TEletronicDistribution.\_TableOnMouseDown(Sender: TObject;

Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

begin

ButtonPressing := True;

Table.Picture.LoadFromFile('Graphics/Distribuição/Table\_2.png');

end;

procedure TEletronicDistribution.\_TableOnMouseEnter(Sender: TObject);

begin

if not ButtonPressing then

Table.Picture.LoadFromFile('Graphics/Distribuição/Table\_1.png');

end;

procedure TEletronicDistribution.\_TableOnMouseLeave(Sender: TObject);

begin

if not ButtonPressing then

Table.Picture.LoadFromFile('Graphics/Distribuição/Table.png');

end;

procedure TEletronicDistribution.\_TableOnMouseUp(Sender: TObject;

Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

begin

ButtonPressing := False;

Table.Picture.LoadFromFile('Graphics/Distribuição/Table.png');

end;

procedure TEletronicDistribution.\_\_\_1sOnMouseDown(Sender: TObject;

Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

var A : Boolean;

I : Byte;

R : TRect;

begin

for I := 1 to 19 do

begin

a := Sender.Equals(Niveis[I]);

if a then

MouseActive := I;

end;

R.Left := 105;

R.Right := 880;

R.Top := 236;

R.Bottom := 607;

ClipCursor(@R);

end;

procedure TEletronicDistribution.\_\_1sOnMouseUp(Sender: TObject;

Button: TMouseButton; Shift: TShiftState; X, Y: Integer);

var I, J : Integer;

T : Boolean;

begin

I := Niveis[MouseActive].Left;

J := Niveis[MouseActive].Top;

if ( I > 120 ) and ( I < 880 ) and ( J > 540 ) and ( J < 620 ) then

begin

for I := 1 to 19 do

if Inside[I].Caption = '' then

begin

Niveis[MouseActive].Top := 570;

Inside[I] := Niveis[MouseActive];

Count := Count + 1;

FinishPress();

Exit();

end

else

if Inside[I].Equals(Niveis[MouseActive]) then

begin

Niveis[MouseActive].Top := 570;

FinishPress();

Exit();

end;

end

else

begin

for I := 1 to 19 do

if Inside[I].Equals(Niveis[MouseActive]) then

begin

Inside[I] := TLabel.Create(Inside[I]);

Back(I);

Count := Count - 1;

RemovePoints( Level \* 50 );

FinishPress();

Exit();

end

end;

FinishPress();

end;

end.

unit Load;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, ExtCtrls, pngimage, jpeg;

type

TLoading = (TForm)

Image3: TImage;

Timer1: TTimer;

Image2: TImage;

Image1: TImage;

procedure Timer1Timer(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure FormCreate(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Loading: TLoading;

implementation

uses DB\_Integrator, ManualVideo, Menu;

{$R \*.dfm}

procedure TLoading.FormClose(Sender: TObject; var Action: TCloseAction);

begin

DataModule1.DataModuleDestroy(Self);

end;

procedure TLoading.FormCreate(Sender: TObject);

begin

DB\_Integrator.PerformForm( Self );

end;

procedure TLoading.Timer1Timer(Sender: TObject);

var Rand : Byte;

begin

Rand := 0;

Image3.Width := Image3.Width + 10;

Image3.Repaint;

while Rand < 10 do

Rand := Random(100);

Timer1.Interval := Rand;

if Image3.Width >= 885 then

begin

Timer1.Enabled := False;

FreeAndNil(Self);;

Inicio.AlphaBlendValue := 255;

Inicio.Show();

end;

end;

end.

unit Login;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, pngimage, ExtCtrls;

type

TTeacherLogin = class(TForm)

Forgot\_PassWord: TLabel;

BackGround: TImage;

User: TEdit;

PassWord: TEdit;

New: TImage;

Enter: TImage;

Cancel: TImage;

Logo: TImage;

procedure OnEnter(Sender: TObject);

procedure OnLeave(Sender: TObject);

procedure OnCreate(Sender: TObject);

procedure OnMouseEnter(Sender: TObject);

procedure UpdatePicture( Who : Byte; ID : Byte );

procedure OnMouseLeave(Sender: TObject);

procedure OnMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure OnMouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure CancelClick(Sender: TObject);

procedure OnKeyPress(Sender: TObject; var Key: Char);

procedure NewClick(Sender: TObject);

procedure Forgot\_PassWordClick(Sender: TObject);

procedure EnterClick(Sender: TObject);

procedure OnShow(Sender: TObject);

procedure BackGroundClick(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

TeacherLogin: TTeacherLogin;

implementation

uses DB\_Integrator, Sound, Register, Tip, Menu;

{$R \*.dfm}

procedure TTeacherLogin.BackGroundClick(Sender: TObject);

begin

end;

procedure TTeacherLogin.CancelClick(Sender: TObject);

begin

Sound.Play\_Cancel();

Self.Close();

end;

procedure TTeacherLogin.EnterClick(Sender: TObject);

var I : Byte;

var CanLogin : Boolean;

begin

// Se o campo de usuário estiver vazio.

if User.Text = '' then

begin

Application.MessageBox('Insira o usuário.', 'Login', MB\_ICONWARNING );

Exit;

end

else if PassWord.Text = '' then

begin

Application.MessageBox('Insira a senha.', 'Login', MB\_ICONWARNING );

Exit;

end;

// Executa efeito sonoro de decisão

Sound.Play\_Decision(3);

// Limpa comandos SQL

DB\_Integrator.DataModule1.ADOQuery2.SQL.Clear;

// Adiciona comando SQL ( Obtém a quantidade de usuários )

DB\_Integrator.DataModule1.ADOQuery2.SQL.Add('Select Count(\*) as ID from' +

' Professor');

// Ativa o SQL

DataModule1.Adoquery2.Active := True;

// Executa comandos SQL

DataModule1.Adoquery2.ExecSQL;

// Para todos as usuários

for I := 1 to DataModule1.ADOQuery2.FieldByName('ID').AsInteger do

// Limpa comandos SQL

DataModule1.ADOQuery2.SQL.Clear();

// Adiciona comando SQL ( Para todos os Usuários )

DataModule1.ADOQuery2.SQL.Add('Select Usuario from Professo' +

'r where ID = ' + IntToStr(I));

// Ativa SQL

DataModule1.ADOQuery2.Active := True;

// Executa SQL

DataModule1.ADOQuery2.ExecSQL;

// Se encontrar o usuário

if Crypt(User.Text) = DataModule1.ADOQuery2.FieldByName(

'Usuario').AsString then

// Limpa comandos SQL

DataModule1.ADOQuery2.SQL.Clear();

// Adiciona comando SQL ( Para todas as Senhas )

DataModule1.ADOQuery2.SQL.Add('Select \* from Professo' +

'r where ID = ' + IntToStr(I));

// Ativa SQL

DataModule1.ADOQuery2.Active := True;

// Executa SQL

DataModule1.ADOQuery2.ExecSQL;

// Se a senha corresponder à senha do usuário

if Crypt(PassWord.Text) = DataModule1.ADOQuery2.FieldByName(

'Senha').AsString then

ShowMessage('Bem-vindo, ' + DataModule1.ADOQuery2.FieldByName(

'Nome').AsString + '.');

// Permite logar

CanLogin := True;

// Sai do loop

Break;

end;

end;

end;

// Se puder logar

if CanLogin then

// Fecha a janela

Self.Close();

// Loga

Inicio.Login( True );

end

else

Application.MessageBox('Usuário e/ou senha incorreto(s)', 'Login', MB\_ICONWARNING );

end;

procedure TTeacherLogin.Forgot\_PassWordClick(Sender: TObject);

begin

Sound.Play\_Cursor(1);

Tip.PassWordFaq.ShowModal();

end;

procedure TTeacherLogin.NewClick(Sender: TObject);

begin

// Executa efeito sonoro de confirmação

Sound.Play\_Decision(3);

RegisterUser.ShowModal();

end;

procedure TTeacherLogin.OnCreate(Sender: TObject);

begin

DB\_Integrator.PerformForm( Self );

Self.Width := 400;

Self.Height := 400;

end;

procedure TTeacherLogin.OnEnter(Sender: TObject);

begin

Forgot\_PassWord.Font.Color := RGB(200,200,255)

end;

procedure TTeacherLogin.OnKeyPress(Sender: TObject; var Key: Char);

begin

// Se a tecla Escape é pressionada

if Key = #27 then

begin

Key := #0;

Sound.Play\_Cancel();

Self.Close();

end

else if Key = #13 then { Tecla Enter }

begin

// Impede Bip

Key := #0;

// Executa efeito sonoro de próximo

Play\_Move();

// Move o cursor até o próximo Campo

Perform(WM\_nextdlgctl,0,0);

end;

end;

procedure TTeacherLogin.OnLeave(Sender: TObject);

begin

Forgot\_PassWord.Font.Color := clGray

end;

procedure TTeacherLogin.OnMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

begin

if Sender = Enter then

UpdatePicture(1, 3)

else if Sender = New then

UpdatePicture(2, 3)

else if Sender = Cancel then

UpdatePicture(3, 3);

end;

procedure TTeacherLogin.OnMouseEnter(Sender: TObject);

begin

if Sender = Enter then

UpdatePicture(1, 2)

else if Sender = New then

UpdatePicture(2, 2)

else if Sender = Cancel then

UpdatePicture(3, 2);

end;

procedure TTeacherLogin.OnMouseLeave(Sender: TObject);

begin

if Sender = Enter then

UpdatePicture(1, 1)

else if Sender = New then

UpdatePicture(2, 1)

else if Sender = Cancel then

UpdatePicture(3, 1);

end;

procedure TTeacherLogin.OnMouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

begin

if Sender = Enter then

UpdatePicture(1, 1)

else if Sender = New then

UpdatePicture(2, 1)

else if Sender = Cancel then

UpdatePicture(3, 1);

end;

procedure TTeacherLogin.OnShow(Sender: TObject);

begin

PassWord.Clear();

User.Clear();

end;

procedure TTeacherLogin.UpdatePicture( Who : Byte; ID: Byte);

begin

case Who of

1: Enter.Picture.LoadFromFile('Graphics/Login/Enter\_' + IntToStr(ID) + '.png');

2: New.Picture.LoadFromFile('Graphics/Login/New\_' + IntToStr(ID) + '.png');

3: Cancel.Picture.LoadFromFile('Graphics/Login/Cancel\_' + IntToStr(ID) + '.png');

end;

end;

end.

unit Manual;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, ExtCtrls, StdCtrls, pngimage, jpeg;

type

TAjuda = class(TForm)

PictureManual: TImage;

ToolPicture: TImage;

Preview: TImage;

Next: TImage;

Page: TLabel;

procedure FormCreate(Sender: TObject);

procedure SetManual(Game: Byte; Kind : Byte);

procedure OnClose(Sender: TObject; var Action: TCloseAction);

procedure ImageClick(Sender: TObject);

procedure UpdateImage();

procedure OnMouseEnter(Sender: TObject);

procedure OnMouseLeave(Sender: TObject);

procedure UpdateLevel();

procedure OnKeyPress(Sender: TObject; var Key: Char);

procedure ToolPictureClick(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Ajuda: TAjuda;

Pages: Integer;

Index: Integer = 1;

Directory: String;

implementation

uses DB\_Integrator, Menu, Sound;

{$R \*.dfm}

procedure TAjuda.FormCreate(Sender: TObject);

begin

DB\_Integrator.PerformForm( Self );

end;

procedure TAjuda.UpdateImage();

begin

PictureManual.Picture.LoadFromFile(Directory + IntToStr(Index) + '.jpg');

end;

procedure TAjuda.UpdateLevel;

begin

Page.Caption := IntToStr(Index) + ' / ' + IntToStr(Pages);

end;

procedure TAjuda.ImageClick(Sender: TObject);

begin

if Sender = Next then

begin

if Index < Pages then

Index := Index + 1

end

else if Sender = Preview then

begin

if Index > 1 then

Index := Index - 1;

end;

UpdateImage();

Self.UpdateLevel();

end;

procedure TAjuda.OnClose(Sender: TObject; var Action: TCloseAction);

begin

DataModule1.DataModuleDestroy(Self);

end;

procedure TAjuda.OnKeyPress(Sender: TObject; var Key: Char);

begin

// Se a tecla Escape é pressionada

if Key = #27 then

begin

Sound.Play\_Cancel;

FreeAndNil(Self);

Inicio.Mode := 5;

Inicio.Show();

end;

end;

procedure TAjuda.OnMouseEnter(Sender: TObject);

begin

if Sender = Next then

Next.Picture.LoadFromFile('Graphics/Manual/Button\_Right\_2.png')

else if Sender = Preview then

Preview.Picture.LoadFromFile('Graphics/Manual/Button\_Left\_2.png')

end;

procedure TAjuda.OnMouseLeave(Sender: TObject);

begin

if Sender = Next then

Next.Picture.LoadFromFile('Graphics/Manual/Button\_Right\_1.png')

else if Sender = Preview then

Preview.Picture.LoadFromFile('Graphics/Manual/Button\_Left\_1.png')

end;

procedure TAjuda.SetManual(Game: Byte; Kind : Byte);

var I : Integer;

begin

Directory := 'Graphics/Manuais/' + IntToStr(Game) + '/' + IntToStr(Kind) + '/';

for I := 1 to 99 do

if not FileExists(Directory + IntToStr(I) + '.jpg') then

begin

Pages := I - 1;

Break

end;

Index := 1;

Self.UpdateLevel();

UpdateImage();

end;

procedure TAjuda.ToolPictureClick(Sender: TObject);

begin

end;

end.

unit ManualVideo;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, MPlayer, ExtCtrls, pngimage;

type

TVideoManual = class(TForm)

Panel\_video: TPanel;

MediaPlayer1: TMediaPlayer;

Timer1: TTimer;

procedure SetVideo( ID : Byte );

procedure PlayVideo();

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure FormCreate(Sender: TObject);

procedure OnKeyPress(Sender: TObject; var Key: Char);

procedure OpenVideo();

procedure Timer1Timer(Sender: TObject);

procedure Panel\_videoClick(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

protected

procedure CreateParams(var Params: TCreateParams); override;

end;

var

VideoManual: TVideoManual;

Dir : String;

implementation

uses DB\_Integrator, Sound, Menu;

{$R \*.dfm}

procedure TVideoManual.CreateParams(var Params: TCreateParams);

begin

inherited;

Params.WindowClass.Style := Params.WindowClass.Style or $00020000;

end;

procedure TVideoManual.SetVideo(ID: Byte);

begin

Dir := 'Vídeos/' + IntToStr(ID) + '.wmv';

end;

procedure TVideoManual.Timer1Timer(Sender: TObject);

begin

if ( Self.Width < 600 ) then

begin

Self.Width := Self.Width + 4;

Self.Left := ( 1024 - Self.Width ) div 2;

end

else

begin

MediaPlayer1.DisplayRect := Self.ClientRect;

PlayVideo();

Timer1.Enabled := False;

end;

if ( Self.Height < 500 ) then

begin

Self.Height := Self.Height + 4;

Self.Top := ( 768 - Self.Height ) div 2;

end;

end;

procedure TVideoManual.OpenVideo;

begin

MediaPLayer1.FileName := Dir;

MediaPlayer1.Open();

end;

procedure TVideoManual.FormClose(Sender: TObject; var Action: TCloseAction);

begin

DataModule1.DataModuleDestroy(Self);

end;

procedure TVideoManual.FormCreate(Sender: TObject);

begin

DB\_Integrator.PerformForm( Self );

Self.Width := 1;

Self.Height := 1;

Self.Left := ( 1024 - Self.Width ) div 2;

Self.Top := ( 768 - Self.Height ) div 2;

end;

procedure TVideoManual.OnKeyPress(Sender: TObject; var Key: Char);

begin

// Se a tecla Escape é pressionada

if Key = #27 then

begin

MediaPlayer1.Stop();

Sound.Play\_Cancel;

Inicio.Enabled := True;

Inicio.Mode := 5;

Inicio.Show();

Inicio.FadeOut.Visible := False;

FreeAndNil(Self);

end;

end;

procedure TVideoManual.Panel\_videoClick(Sender: TObject);

begin

end;

procedure TVideoManual.PlayVideo;

begin

MediaPLayer1.Play();

end;

end.

unit Master;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, pngimage, ExtCtrls;

type

TAdmin = class(TForm)

BackGround: TImage;

Name: TEdit;

Image3: TImage;

Continue: TImage;

procedure OnMouseEnter(Sender: TObject);

procedure OnMouseLeave(Sender: TObject);

procedure OnMouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure OnMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure OnCreate(Sender: TObject);

procedure OnKeyPress(Sender: TObject; var Key: Char);

procedure ContinueClick(Sender: TObject);

procedure OnShow(Sender: TObject);

procedure BackGroundClick(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Admin: TAdmin;

implementation

uses DB\_Integrator, Sound, Register, Login;

{$R \*.dfm}

procedure TAdmin.BackGroundClick(Sender: TObject);

begin

end;

procedure TAdmin.ContinueClick(Sender: TObject);

begin

if Name.Text = '' then

// Informa que o usuário já existe

Application.MessageBox('Insira a senha.', 'Login', MB\_ICONWARNING );

// Limpa comandos SQL

DB\_Integrator.DataModule1.ADOQuery3.SQL.Clear;

// Adiciona comando SQL ( Obtém a senha )

DB\_Integrator.DataModule1.ADOQuery3.SQL.Add('Select Senha from' +

' Administrador');

// Ativa o SQL

DataModule1.Adoquery3.Active := True;

// Executa comandos SQL

DataModule1.Adoquery3.ExecSQL;

if Crypt(Name.Text) = DataModule1.ADOQuery3.FieldByName('Senha').AsString then

begin

RegisterUser.Make();

// Informa que o usuário já existe

Application.MessageBox('Usuário cadastrado com sucesso', 'Login', MB\_ICONINFORMATION );

Self.Close();

RegisterUser.Close();

Login.TeacherLogin.Close();

end

else

// Informa que o usuário já existe

Application.MessageBox('Senha incorreta.', 'Login', MB\_ICONWARNING );

end;

procedure TAdmin.OnCreate(Sender: TObject);

begin

DB\_Integrator.PerformForm( Self );

end;

procedure TAdmin.OnKeyPress(Sender: TObject; var Key: Char);

begin

// Se a tecla Escape é pressionada

if Key = #27 then

begin

Key := #0;

Sound.Play\_Cancel();

Self.Close();

end

else if Key = #13 then { Tecla Enter }

begin

// Impede Bip

Key := #0;

// Continua

end;

end;

procedure TAdmin.OnMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

begin

Continue.Picture.LoadFromFile('Graphics/Master/Continue\_3.png');

end;

procedure TAdmin.OnMouseEnter(Sender: TObject);

begin

Continue.Picture.LoadFromFile('Graphics/Master/Continue\_2.png');

end;

procedure TAdmin.OnMouseLeave(Sender: TObject);

begin

Continue.Picture.LoadFromFile('Graphics/Master/Continue\_1.png');

end;

procedure TAdmin.OnMouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

begin

Continue.Picture.LoadFromFile('Graphics/Master/Continue\_1.png');

end;

procedure TAdmin.OnShow(Sender: TObject);

begin

Name.Clear();

end;

end.

unit Menu;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, pngimage, ExtCtrls, Tabela, jpeg, DB\_Integrator,

Insert\_Stoichiometric, Sound, JogoDistribuicao, GameStoichiometric;

type

TInicio = class(TForm)

Back: TImage;

FadeOut: TImage;

procedure FormCreate(Sender: TObject);

procedure ChangePicture();

procedure Mode1();

procedure Mode2();

procedure Mode3();

procedure Mode4();

procedure Mode5();

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);

procedure Login( Kind : Boolean );

procedure FadeOutClick(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

// Menu atual

Mode : Byte;

end;

var

Inicio : TInicio;

// Armazena o índice da atual escolha selecionada

Cursor : Byte = 1;

// Jogo selecionado

Game : Byte;

implementation

uses Geometry, Manual, ManualVideo, Load, Login;

{$R \*.dfm}

procedure TInicio.Login(Kind: Boolean);

begin

FadeOut.Visible := False;

if Kind then

begin

Mode := 4;

Cursor := 1;

end

else

begin

Mode := 1;

Cursor := 1;

end;

ChangePicture();

end;

procedure TInicio.Mode2;

begin

Mode := 3;

Game := Cursor;

Cursor := 1;

ChangePicture();

end;

procedure TInicio.Mode5();

begin

if Cursor < 3 then

begin

Ajuda := TAjuda.Create(Self);

Ajuda.SetManual(Game, Cursor);

Ajuda.Show();

Self.Hide();

Exit();

end

else if Cursor = 4 then

begin

Mode := 3;

Cursor := 1;

ChangePicture();

end

else if Cursor = 3 then

begin

FadeOut.Visible := True;

VideoManual := TVideoManual.Create(Self);

VideoManual.SetVideo(Game);

VideoManual.OpenVideo();

Self.Enabled := False;

VideoManual.Show();

end;

end;

procedure TInicio.Mode3;

begin

Play\_Decision(1);

if Cursor = 3 then

begin

Mode := 5;

Cursor := 1;

ChangePicture();

Exit();

end

else if Cursor = 4 then

begin

Mode := 2;

Cursor := 1;

ChangePicture();

Exit();

end

else if Cursor = 2 then

case Game of

1:

begin

FreeAndNil(EletronicDistribution);

EletronicDistribution := TEletronicDistribution.Create(Self);

EletronicDistribution.FormCreate(Application);

EletronicDistribution.ToChoose.Enabled := true;

EletronicDistribution.ToConfirm.Enabled := True;

EletronicDistribution.Show();

Self.Hide();

end;

2:

begin

GeometryGame := TGeometryGame.Create(Self);

GeometryGame.ResetVariables();

GEometryGame.CallMe();

GeometryGame.Show();

Self.Hide();

end;

3:

begin

CalculoEstequiometrico := TCalculoEstequiometrico.Create(Application);

CalculoEstequiometrico.CallMe();

CalculoEstequiometrico.Show();

Self.Hide();

end;

end;

end;

procedure TInicio.Mode4();

begin

if Cursor = 4 then

begin

Mode := 1;

Cursor := 1;

ChangePicture();

end

else if Cursor = 1 then

begin

InsertStoichiometric := TInsertStoichiometric.Create(Application);

Self.Hide();

InsertStoichiometric.OnFormCreate(Application);

InsertStoichiometric.Show();

end

end;

procedure TInicio.Mode1();

begin

case Cursor of

1:

begin

Mode := 2;

Cursor := 1;

ChangePicture();

end;

2:

begin

FadeOut.Visible := True;

TeacherLogin.ShowModal;

FadeOut.Visible := False;

end;

3:

begin

Tabela.PeriodicTable.Show();

Tabela.PeriodicTable.Enabled := true;

Tabela.PeriodicTable.Visible := true;

Tabela.PeriodicTable.ActiveForm := true;

Tabela.PeriodicTable.CursorIndex := Tabela.PeriodicTable.SaveCursorIndex;

Tabela.WhereTo := 3;

Self.Hide();

end;

4: Self.Close();

end;

end;

procedure TInicio.ChangePicture();

begin

Back.Picture.LoadFromFile('Graphics/Inicio/' + IntToStr(Mode) + '/' + IntToStr

(Cursor) + '.jpg');

end;

procedure TInicio.FadeOutClick(Sender: TObject);

begin

end;

procedure TInicio.FormClose(Sender: TObject; var Action: TCloseAction);

begin

DataModule1.DataModuleDestroy(Self);

end;

procedure TInicio.FormCreate(Sender: TObject);

begin

PerformForm(Self);

Cursor := 1;

Mode := 1;

ShowCursor(False);

end;

procedure TInicio.FormKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

begin

if ( Key = VK\_ESCAPE ) and ( Mode = 2 ) then

begin

Sound.Play\_Cancel;

Mode := 1;

Cursor := 1;

ChangePicture();

end;

if Key = VK\_DOWN then

begin

Play\_Cursor(1);

if Cursor < 4 then

Cursor := Cursor + 1

else

Cursor := 1;

ChangePicture();

end

else if Key = VK\_UP then

begin

Play\_Cursor(1);

if Cursor > 1 then

Cursor := Cursor - 1

else

Cursor := 4;

ChangePicture();

end;

if Key = VK\_Return then

begin

Play\_Decision(1);

case Mode of

1: Mode1();

2: Mode2();

3: Mode3();

4: Mode4();

5: Mode5();

end;

end;

end;

end.

unit Propriedades;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, pngimage, ExtCtrls, MMSystem, JogoDistribuicao, jpeg,

DB\_Integrator, Sound;

type

//==============================================================================

// \*\* TWindowPropertie

//------------------------------------------------------------------------------

// Esta classe é responsável por exibir informações sobre cada elemento

// químico da tabela periódica.

//==============================================================================

TWindowPropertie = class(TForm)

Propriedades: TImage;

Nome: TLabel;

NumeroAtomico: TLabel;

MassaAtomica: TLabel;

Simbolo: TLabel;

Grupo: TLabel;

Periodo: TLabel;

Bloco: TLabel;

Eletrons: TLabel;

SerieQuimica: TLabel;

RaioAtomico: TLabel;

RaioCovalente: TLabel;

RaioVanDerWalls: TLabel;

ConfiguracaoEletronica: TLabel;

EstruturaCristalina: TLabel;

EstadoDaMateria: TLabel;

PontoDeFusao: TLabel;

PontoDeEbulicao: TLabel;

PressaoAoVapor: TLabel;

VelocidadeDoSom: TLabel;

CalorEspecifico: TLabel;

CondutividadeTermica: TLabel;

ClasseMagnetica: TLabel;

Eletronegatividade: TLabel;

Descricao: TLabel;

Imagem: TImage;

Botao: TImage;

procedure OnShow(Sender: TObject);

procedure OnClose(Sender: TObject; var Action: TCloseAction);

procedure OnMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure OnMouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure OnMouseEnter(Sender: TObject);

procedure OnMouseLeave(Sender: TObject);

procedure OnClick(Sender: TObject);

procedure WentFromTable;

procedure WentFromGame( Index : Byte );

procedure FormDestroy(Sender: TObject);

procedure PropriedadesClick(Sender: TObject);

private

public

end;

var

WindowPropertie: TWindowPropertie;

// Número atômico do elemento químico a ser exibido

ID : Byte = 1;

// Redirecionado a partir da tabela periódica ?

FromTable : Boolean;

implementation

uses Tabela;

{$R \*.dfm}

//------------------------------------------------------------------------------

// \* Quando chamada a partir da tabela periódica

//------------------------------------------------------------------------------

procedure TWindowPropertie.WentFromTable;

begin

// Atribui que a classe foi chamada a partir da tabela

FromTable := True;

end;

//------------------------------------------------------------------------------

// \* Quando chamada de um game

// Index : Elemento químico

//------------------------------------------------------------------------------

procedure TWindowPropertie.WentFromGame(Index: Byte);

begin

// Atribui que a classe não foi chamada da tabela periódica

FromTable := False;

// Armazena o número atômico do elemento químico

ID := Index;

end;

//------------------------------------------------------------------------------

// \* Destruição do formulário

// Sender : Componente responsável pela execução

//------------------------------------------------------------------------------

procedure TWindowPropertie.FormDestroy(Sender: TObject);

begin

// Chama o destruidor de formulários

DataModule1.DataModuleDestroy(Self);

end;

//------------------------------------------------------------------------------

// \* Evento ao clicar no botão Voltar

// Sender : Componente responsável pela execução

//------------------------------------------------------------------------------

procedure TWindowPropertie.OnClick(Sender: TObject);

begin

// Executa efeito sonoro

Play\_Cancel();

// Fecha a janela

Close();

end;

//------------------------------------------------------------------------------

// \* Processa o fechamento da janela

// Sender : Componente responsável pela execução

// Action : Tipo de ação

//------------------------------------------------------------------------------

procedure TWindowPropertie.OnClose(Sender: TObject; var Action: TCloseAction);

var GambzX, GambzY : Integer;

begin

// Se a classe foi chamada a partir da tabela periódica

if FromTable then

begin

// Exibe a tabela periódica

Tabela.PeriodicTable.Show;

// Torna a tabela periódica acessível

Tabela.PeriodicTable.Enabled := true;

// Torna a tabela periódica visível

Tabela.PeriodicTable.Visible := true;

// Permite a ativação da tabela periódica

Tabela.PeriodicTable.ActiveForm := true;

// Reestabelece a posição do cursor na tabela

Tabela.PeriodicTable.CursorIndex := Tabela.PeriodicTable.SaveCursorIndex;

// Altera a imagem do botão

PeriodicTable.Botao.Picture.LoadFromFile('Graphics\Tabela\Botão\Normal.png');

// Esconde a janela atual

WindowPropertie.Hide();

end

// Se foi chamado pelo jogo Distribuição Eletrônica

else

begin

// Exibe o jogo

EletronicDistribution.Show();

// Esconde a janela atual

WindowPropertie.Hide();

end;

end;

//------------------------------------------------------------------------------

// \* Processo executado ao apertar o botão do mouse sobre o botão 'Voltar'

// Sender : Componente responsável pela execução ( Botão Voltar )

// Action : Tipo de ação

// Button : Tipo de botão ( Esquerdo, Direito e etc .. )

// Shift : Estado da tecla CTRL

// X : Posição X do Cursor dentro do Botão

// Y : Posição Y do Cursor dentro do Botão

//------------------------------------------------------------------------------

procedure TWindowPropertie.OnMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

begin

// Altera a imagem do botão 'Voltar'

Botao.Picture.LoadFromFile('Graphics/Properties/Botao/Press.png');

end;

//------------------------------------------------------------------------------

// \* Processo executado ao adentrar as coordenadas do botão 'Voltar'

// Sender : Componente responsável pela execução

//------------------------------------------------------------------------------

procedure TWindowPropertie.OnMouseEnter(Sender: TObject);

begin

// Altera a imagem do botão 'Voltar'

Botao.Picture.LoadFromFile('Graphics/Properties/Botao/Sob.png');

end;

//------------------------------------------------------------------------------

// \* Processo executado ao sair dos limites das coordenadas do botão 'Voltar'

// Sender : Componente responsável pela execução

//------------------------------------------------------------------------------

procedure TWindowPropertie.OnMouseLeave(Sender: TObject);

begin

// Altera a imagem do botão 'Voltar'

Botao.Picture.LoadFromFile('Graphics/Properties/Botao/Normal.png');

end;

//------------------------------------------------------------------------------

// \* Processo executado ao soltar o botão esquerdo do mouse sobre o botão Voltar

// Sender : Componente responsável pela execução

// Button : Tipo de botão ( Esquerdo, Direito e etc .. )

// Shift : Estado da tecla CTRL

// X : Posição X do Cursor dentro do Botão

// Y : Posição Y do Cursor dentro do Botão

//------------------------------------------------------------------------------

procedure TWindowPropertie.OnMouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

begin

// Altera a imagem do botão 'Voltar'

Botao.Picture.LoadFromFile('Graphics/Properties/Botao/Normal.png');

end;

//------------------------------------------------------------------------------

// \* Exibe a janela

// Sender : Componente responsável pela execução

//------------------------------------------------------------------------------

procedure TWindowPropertie.OnShow(Sender: TObject);

var Properties : Array[0..23] of TLabel;

I : Byte;

begin

// Atribuição de Labels a um conjunto

Properties[0] := NumeroAtomico;

Properties[1] := Nome;

Properties[2] := Simbolo;

Properties[3] := MassaAtomica;

Properties[4] := Grupo;

Properties[5] := Periodo;

Properties[6] := Bloco;

Properties[7] := Eletrons;

Properties[8] := SerieQuimica;

Properties[9] := RaioAtomico;

Properties[10] := RaioCovalente;

Properties[11] := RaioVanDerWalls;

Properties[12] := ConfiguracaoEletronica;

Properties[13] := EstruturaCristalina;

Properties[14] := EstadoDaMateria;

Properties[15] := PontoDeFusao;

Properties[16] := PontoDeEbulicao;

Properties[17] := PressaoAoVapor;

Properties[18] := VelocidadeDoSom;

Properties[19] := CalorEspecifico;

Properties[20] := CondutividadeTermica;

Properties[21] := ClasseMagnetica;

Properties[22] := Eletronegatividade;

Properties[23] := Descricao;

// Altera o Cursor para o Foguetinho

Screen.Cursor := 5;

// Para todos os labels do formulário

for I := 0 to 23 do

// Se diferente de Configuração Eletrônica

if I <> 12 then

// Se chamado pela tabela periódica

if FromTable then

begin

// Obtém a informação do elemento e a atribui ao label

Properties[I].Caption := Elemento[PeriodicTable.AtomicPicture[PeriodicTable.SaveCursorIndex]].Get\_Elemento(I);

// Carrega a imagem do elemento químico

Imagem.Picture.LoadFromFile(Elemento[PeriodicTable.AtomicPicture[PeriodicTable.SaveCursorIndex]].Get\_Elemento(25));

end

// Se chamado pelo jogo Distribuição Eletrônica

else

begin

// Obtém a informação do elemento e a atribui ao label

Properties[I].Caption := Elemento[ID].Get\_Elemento(I);

// Carrega a imagem do elemento químico

Imagem.Picture.LoadFromFile(Elemento[ID].Get\_Elemento(25));

end;

// Adapta formulário aos demais

PerformForm(Self);

end;

end.

unit Register;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, pngimage, ExtCtrls;

type

TRegisterUser = class(TForm)

BackGround: TImage;

Cancel: TImage;

Register: TImage;

Logo: TImage;

PassWord: TEdit;

User: TEdit;

Tip: TEdit;

ToTable: TImage;

Image1: TImage;

Image2: TImage;

Name: TEdit;

Image3: TImage;

procedure OnMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure UpdatePicture( Who : Byte; ID: Byte);

procedure OnMouseEnter(Sender: TObject);

procedure OnMouseLeave(Sender: TObject);

procedure OnMouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure FormCreate(Sender: TObject);

procedure OnKeyPress(Sender: TObject; var Key: Char);

procedure CancelClick(Sender: TObject);

procedure RegisterClick(Sender: TObject);

function UserSearching() : Boolean;

procedure Make();

procedure OnShow(Sender: TObject);

procedure BackGroundClick(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

RegisterUser: TRegisterUser;

implementation

uses DB\_Integrator, Sound, Master;

{$R \*.dfm}

procedure TRegisterUser.Make();

begin

Sound.Play\_Decision(1);

// ------------------------------------------------------------------------ //

// Limpa comandos SQL

DB\_Integrator.DataModule1.ADOQuery2.SQL.Clear;

// Cria parâmetros para inserção

DB\_Integrator.DataModule1.ADOQuery2.SQL.Add('Insert into Professor' +

' (Nome, Usuario, Senha, Dica)' +

' Values (:A, :B, :C, :D)');

// Insera-os no banco de dados

DataModule1.ADOQuery2.Parameters[0].Value := Name.Text;

DataModule1.ADOQuery2.Parameters[1].Value := Crypt(User.Text);

DataModule1.ADOQuery2.Parameters[2].Value := Crypt(PassWord.Text);

DataModule1.ADOQuery2.Parameters[3].Value := Tip.Text;

// Executa comandos SQL

DataModule1.ADOQuery2.ExecSQL();

// ------------------------------------------------------------------------ //

end;

procedure TRegisterUser.BackGroundClick(Sender: TObject);

begin

end;

procedure TRegisterUser.CancelClick(Sender: TObject);

begin

// Executa efeito sonoro de cancelamento

Sound.Play\_Cancel();

// Fecha cadastro de usuário

Self.Close();

end;

procedure TRegisterUser.FormCreate(Sender: TObject);

begin

DB\_Integrator.PerformForm( Self );

end;

procedure TRegisterUser.OnKeyPress(Sender: TObject; var Key: Char);

begin

// Se a tecla Escape é pressionada

if Key = #27 then

begin

Key := #0;

Sound.Play\_Cancel();

Self.Close();

end

else if Key = #13 then { Tecla Enter }

begin

// Impede Bip

Key := #0;

// Executa efeito sonoro de próximo

Play\_Move();

// Move o cursor até o próximo Campo

Perform(WM\_nextdlgctl,0,0);

end;

end;

procedure TRegisterUser.OnMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

begin

if Sender = Register then

UpdatePicture(1, 3)

else if Sender = Cancel then

UpdatePicture(2, 3)

end;

procedure TRegisterUser.OnMouseEnter(Sender: TObject);

begin

if Sender = Register then

UpdatePicture(1, 2)

else if Sender = Cancel then

UpdatePicture(2, 2);

end;

procedure TRegisterUser.OnMouseLeave(Sender: TObject);

begin

if Sender = Register then

UpdatePicture(1, 1)

else if Sender = Cancel then

UpdatePicture(2, 1);

end;

procedure TRegisterUser.OnMouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

begin

if Sender = Register then

UpdatePicture(1, 1)

else if Sender = Cancel then

UpdatePicture(2, 1);

end;

procedure TRegisterUser.OnShow(Sender: TObject);

begin

PassWord.Clear();

Tip.Clear();

Name.Clear();

User.Clear();

end;

procedure TRegisterUser.RegisterClick(Sender: TObject);

begin

// Se o campo de usuário estiver vazio.

if Name.Text = '' then

begin

Application.MessageBox('Insira o nome.', 'Login', MB\_ICONWARNING );

Exit;

end

else if ( User.Text = '' ) or ( Length( User.Text ) < 4 ) then

begin

Application.MessageBox('Insira um usuário correto', 'Login', MB\_ICONWARNING );

Exit;

end

else if ( PassWord.Text = '' ) or ( Length( PassWord.Text ) < 4 ) then

begin

Application.MessageBox('Insira uma senha correta.', 'Login', MB\_ICONWARNING );

Exit;

end

else if Tip.Text = '' then

begin

Application.MessageBox('Insira a dica.', 'Login', MB\_ICONWARNING );

Exit;

end;

if UserSearching() then

begin

// Executa efeito sonoro de decisão

Sound.Play\_Decision(3);

// Abre a janela de permissão

Master.Admin.ShowModal();

end

else

// Informa que o usuário já existe

Application.MessageBox('Usuário já cadastrado', 'Login', MB\_ICONWARNING );

end;

function TRegisterUser.UserSearching() : Boolean;

var I : Byte;

begin

// Limpa comandos SQL

DB\_Integrator.DataModule1.ADOQuery2.SQL.Clear;

// Adiciona comando SQL ( Obtém a quantidade de usuários )

DB\_Integrator.DataModule1.ADOQuery2.SQL.Add('Select Count(\*) as ID from' +

' Professor');

// Ativa o SQL

DataModule1.Adoquery2.Active := True;

// Executa comandos SQL

DataModule1.Adoquery2.ExecSQL;

// Para todos os usuários

for I := 1 to DataModule1.ADOQuery2.FieldByName('ID').AsInteger do

begin

// Limpa comandos SQL

DataModule1.ADOQuery2.SQL.Clear();

// Adiciona comando SQL ( Para todos usuários )

DataModule1.ADOQuery2.SQL.Add('Select Usuario from Professo' +

'r where ID = ' + IntToStr(I));

// Ativa SQL

DataModule1.ADOQuery2.Active := True;

// Executa SQL

DataModule1.ADOQuery2.ExecSQL;

// Se já houver a equação não balanceada

if Crypt(User.Text) = DataModule1.ADOQuery2.FieldByName('Usuario').AsString then

begin

// Retorna que usuário existe

Result := False;

// Sai da função

Exit;

end;

end;

// Retorna que não existe o usuário

Result := True;

end;

procedure TRegisterUser.UpdatePicture( Who : Byte; ID: Byte);

begin

case Who of

1: Register.Picture.LoadFromFile('Graphics/Register/Register\_' + IntToStr(ID) + '.png');

2: Cancel.Picture.LoadFromFile('Graphics/Register/Cancel\_' + IntToStr(ID) + '.png');

end;

end;

end.

unit Solubilidade;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, math, ExtCtrls, pngimage, jpeg, MPlayer;

type

TForm5 = class(TForm)

edt\_solvente: TEdit;

edt\_soluto: TEdit;

lbl\_elemento: TLabel;

lbl\_exemplo: TLabel;

lbl\_pontuacao: TLabel;

img\_background: TImage;

btn\_botao\_off: TImage;

btn\_botao\_on: TImage;

img\_menuzinho: TImage;

img\_jogardnv: TImage;

img\_voltar\_jogos: TImage;

img\_voltar\_menu: TImage;

resultado\_saturada: TImage;

resultado\_insaturada: TImage;

resultado\_supersaturada: TImage;

img\_continuar: TImage;

img\_solucao\_insaturada: TImage;

img\_solucao\_saturada: TImage;

img\_solucao\_super: TImage;

lbl\_pontos\_final: TLabel;

Media: TMediaPlayer;

Panel\_video: TPanel;

img\_solvente: TImage;

img\_soluto: TImage;

Timer\_Solubilidade: TTimer;

procedure FormShow(Sender: TObject);

procedure btn\_botao\_offMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure img\_jogardnvClick(Sender: TObject);

procedure img\_voltar\_jogosClick(Sender: TObject);

procedure img\_voltar\_menuClick(Sender: TObject);

procedure img\_continuarClick(Sender: TObject);

procedure Timer\_SolubilidadeTimer(Sender: TObject);

procedure img\_backgroundClick(Sender: TObject);

private

{ Private declarations }

public

RD\_opcao,RD\_substancia,novo,vezes:integer;

v1: array [1..31] of string;

v2: array [1..31] of string;

v3: array [1..31] of double;

v4: array [1..31] of double;

pontuacao:real;

end;

var

Form5: TForm5;

implementation

{$R \*.dfm}

procedure TForm5.btn\_botao\_offMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

var

soluto,solvente:real;

begin

if not Edt\_Soluto.Enabled then

begin

timer\_Solubilidade.Enabled:=false;

timer\_Solubilidade.Interval:=6000;

panel\_video.Visible:=true;

media.FileName:='Graphics\Animações(Final)\Original\solvente.avi';

media.Open;

media.Play;

timer\_solubilidade.Enabled:=true;

img\_solvente.visible:=false;

solvente:=strtofloat(edt\_solvente.text);

sleep(6000);

if(solvente=v4[RD\_substancia])then

begin

timer\_Solubilidade.Enabled:=false;

timer\_Solubilidade.Interval:=6000;

panel\_video.Visible:=true;

media.FileName:='Graphics\Animações(Final)\Original\Saturada.avi';

media.Open;

media.Play;

timer\_solubilidade.Enabled:=true;

resultado\_saturada.Visible:=true;

img\_solucao\_saturada.Visible:=true;

pontuacao:=pontuacao+1000;

end;

if(solvente<v4[RD\_substancia])then

begin

timer\_Solubilidade.Enabled:=false;

timer\_Solubilidade.Interval:=6000;

panel\_video.Visible:=true;

media.FileName:='Graphics\Animações(Final)\Original\Super\_Saturada.avi';

media.Open;

media.Play;

timer\_solubilidade.Enabled:=true;

resultado\_supersaturada.Visible:=true;

img\_solucao\_super.Visible:=true;

if(pontuacao>0)then

pontuacao:=pontuacao-150;

end;

if(solvente>v4[RD\_substancia])then

begin

timer\_Solubilidade.Enabled:=false;

timer\_Solubilidade.Interval:=6000;

panel\_video.Visible:=true;

media.FileName:='Graphics\Animações(Final)\Original\Insaturada.avi';

media.Open;

media.Play;

timer\_solubilidade.Enabled:=true;

resultado\_insaturada.Visible:=true;

img\_solucao\_insaturada.Visible:=true;

if(pontuacao>0)then

pontuacao:=pontuacao-100;

end;

end;

if(edt\_soluto.Enabled=true)then

begin

timer\_Solubilidade.Enabled:=false;

timer\_Solubilidade.Interval:=4000;

panel\_video.Visible:=true;

media.FileName:='Graphics\Animações(Final)\Original\soluto.avi';

media.Open;

media.Play;

timer\_solubilidade.Enabled:=true;

img\_soluto.visible:=false;

soluto:=strtofloat(edt\_soluto.Text);

sleep(4000);

if(soluto=v3[RD\_substancia])then

begin

timer\_Solubilidade.Enabled:=false;

timer\_Solubilidade.Interval:=6000;

panel\_video.Visible:=true;

media.FileName:='Graphics\Animações(Final)\Original\Saturada.avi';

media.Open;

media.Play;

timer\_solubilidade.Enabled:=true;

resultado\_saturada.Visible:=true;

img\_solucao\_saturada.Visible:=true;

pontuacao:=pontuacao+1000;

end;

if(soluto>v3[RD\_substancia])then

begin

timer\_Solubilidade.Enabled:=false;

timer\_Solubilidade.Interval:=6000;

panel\_video.Visible:=true;

media.FileName:='Graphics\Animações(Final)\Original\Super\_Saturada.avi';

media.Open;

media.Play;

timer\_solubilidade.Enabled:=true;

resultado\_supersaturada.Visible:=true;

img\_solucao\_super.Visible:=true;

if(pontuacao>0)then

pontuacao:=-150;

end;

if(soluto<v3[RD\_substancia])then

begin

timer\_Solubilidade.Enabled:=false;

timer\_Solubilidade.Interval:=6000;

panel\_video.Visible:=true;

media.FileName:='Graphics\Animações(Final)\Original\Insaturada.avi';

media.Open;

media.Play;

timer\_solubilidade.Enabled:=true;

resultado\_insaturada.Visible:=true;

img\_solucao\_insaturada.Visible:=true;

if(pontuacao>0)then

pontuacao:=pontuacao-100;

end;

end;

lbl\_pontuacao.caption:=floattostr(pontuacao);

edt\_soluto.Enabled:=false;

edt\_solvente.Enabled:=false;

btn\_botao\_off.enabled:=false;

img\_continuar.Visible:=true;

end;

procedure TForm5.FormShow(Sender: TObject);

begin

//Pontuacao Inicial

novo:=0;

pontuacao:=0;

vezes:=0;

//Alocando valores no Vetor 1

v1[1]:='NaCI';

v1[2]:='KNO3';

v1[3]:='C12H12O11';

v1[4]:='K2SO4';

v1[5]:='Li2CO3';

v1[6]:='KCl';

v1[7]:='KCI';

v1[8]:='KNO3';

v1[9]:='Li2CO3';

v1[10]:='KCI';

v1[11]:='K2Cr2O7';

v1[12]:='Temp.Esp';

v1[13]:='H3BO3';

v1[14]:='K2SO4';

v1[15]:='Li2CO3';

v1[16]:='NaCI';

v1[17]:='NaI';

v1[18]:='Na2CO3';

v1[19]:='CsNaCl';

v1[20]:='Ce2(SO4)3';

v1[21]:='MG(NO3)2';

v1[22]:='NaCI';

v1[23]:='Fe(OH)2';

v1[24]:='KCI';

v1[25]:='K2Cr2O7';

v1[26]:='Li2CO3';

v1[27]:='K2SO4';

v1[28]:='KNO3';

v1[29]:='Li2CO3';

v1[30]:='NaCI';

//Alocando valores no Vetor 2

v2[1]:='71,4 g/200 ml a 0°';

v2[2]:='26,6 g/200 ml a 0°';

v2[3]:='84 g/200 ml a 0°';

v2[4]:='14,2 g/200 ml a 0°';

v2[5]:='0,310 g/200 ml a 0°';

v2[6]:='51,2 g/200 ml a 0°';

v2[7]:='62 g/200 ml a 10°';

v2[8]:='40 g/200 ml a 10°';

v2[9]:='0,286 g/200 ml a 10°';

v2[10]:='68 g/200 ml a 20°';

v2[11]:='24 g/200 ml a 20°';

v2[12]:='68 g/200 ml a 20°';

v2[13]:='10 g/200 ml a 20°';

v2[14]:='20 g/200 ml a 20°';

v2[15]:='0,266 g/200 ml a 20°';

v2[16]:='72 g/200 ml a 20°';

v2[17]:='358 g/200 ml a 20°';

v2[18]:='42 g/200 ml a 20°';

v2[19]:='72 g/200 ml a 20°';

v2[20]:='40 g/200 ml a 20°';

v2[21]:='148 g/200 ml a 20°';

v2[22]:='72 g/200 ml a 25°';

v2[23]:='0,088 g/200 ml a 25°';

v2[24]:='74 g/200 ml a 30°';

v2[25]:='40 g/200 ml a 30°';

v2[26]:='0,25 g/200 ml a 30°';

v2[27]:='26 g/200 ml a 40°';

v2[28]:='120 g/200 ml a 40°';

v2[29]:='0,234 g/200 ml a 40°';

v2[30]:='79,6 g/200 ml a 100°';

//Alocando valores no Vetor 3

v3[1]:= 357;

v3[2]:=133;

v3[3]:= 336;

v3[4]:=71;

v3[5]:=77;

v3[6]:=256;

v3[7]:= 62;

v3[8]:=80;

v3[9]:=143;

v3[10]:=34;

v3[11]:=60;

v3[12]:=136;

v3[13]:=52;

v3[14]:=55;

v3[15]:=266;

v3[16]:=288;

v3[17]:=716;

v3[18]:=840;

v3[19]:=72;

v3[20]:=75;

v3[21]:=37;

v3[22]:=54;

v3[23]:=22;

v3[24]:=222;

v3[25]:=92;

v3[26]:=95;

v3[27]:=156;

v3[28]:=480;

v3[29]:=351;

v3[30]:=398;

//Alocando valores no Vetor 4

v4[1]:=1000;

v4[2]:=1000;

v4[3]:=800;

v4[4]:=1000;

v4[5]:=500000;

v4[6]:=1000;

v4[7]:=200;

v4[8]:=400;

v4[9]:=100000;

v4[10]:=100;

v4[12]:=500;(erro 2 números doze)

v4[12]:=400;

v4[13]:=1040;

v4[14]:=550;

v4[15]:=200000;

v4[16]:=800;

v4[17]:= 400;

v4[18]:=4000;

v4[19]:= 200;

v4[20]:=375;

v4[21]:= 50;

v4[22]:=150;

v4[23]:=500000;

v4[24]:= 600;

v4[25]:= 460;

v4[26]:= 76000;

v4[27]:=1200;

v4[28]:= 800;

v4[29]:=3000000;

v4[30]:=1000;

lbl\_pontuacao.caption:=floattostr(pontuacao);

Randomize;

begin

RD\_substancia:=RandomRange(1,30);

RD\_opcao:=RandomRange(1,20);

end;

if(RD\_opcao>10)then

begin

edt\_soluto.text:=floattostr(v3[RD\_substancia]);

edt\_soluto.Enabled:=false;

edt\_solvente.Enabled:=true;

img\_solvente.Visible:=true;

end

else

begin

edt\_solvente.text:=floattostr(v4[RD\_substancia]);

edt\_soluto.Enabled:=true;

edt\_solvente.Enabled:=false;

img\_soluto.Visible:=true;

end;

lbl\_elemento.caption:=v1[RD\_substancia];

lbl\_exemplo.caption:='dado: '+v2[RD\_substancia];

end;

procedure TForm5.img\_backgroundClick(Sender: TObject);

begin

end;

procedure TForm5.img\_continuarClick(Sender: TObject);

begin

resultado\_saturada.visible:=false;

resultado\_insaturada.visible:=false;

resultado\_supersaturada.visible:=false;

img\_continuar.visible:=false;

vezes:=vezes+1;

if(vezes=3)then

begin

lbl\_pontos\_final.Visible:=true;

lbl\_pontos\_final.Caption:=floattostr(pontuacao);

img\_menuzinho.Visible:=true;

img\_jogardnv.Visible:=true;

img\_voltar\_jogos.Visible:=true;

img\_voltar\_menu.Visible:=true;

end

else

begin

btn\_botao\_off.enabled:=true;

edt\_soluto.Text:='';

edt\_solvente.Text:='';

img\_solucao\_saturada.Visible:=false;

img\_solucao\_insaturada.Visible:=false;

img\_solucao\_super.Visible:=false;

Randomize;

begin

RD\_substancia:=RandomRange(1,30);

RD\_opcao:=RandomRange(1,20);

end;

if(RD\_opcao>10)then

begin

img\_solvente.Visible:=true;

edt\_soluto.text:=floattostr(v3[RD\_substancia]);

edt\_soluto.Enabled:=false;

edt\_solvente.Enabled:=true;

end

else

begin

img\_soluto.Visible:=true;

edt\_solvente.text:=floattostr(v4[RD\_substancia]);

edt\_soluto.Enabled:=true;

edt\_solvente.Enabled:=false;

end;

lbl\_elemento.caption:=v1[RD\_substancia];

lbl\_exemplo.caption:='dado: '+v2[RD\_substancia];

end;

end;

procedure TForm5.img\_jogardnvClick(Sender: TObject);

begin

vezes:=0;

pontuacao:=0;

btn\_botao\_off.enabled:=true;

lbl\_pontuacao.Caption:=floattostr(pontuacao);

lbl\_pontos\_final.Visible:=false;

edt\_soluto.Enabled:=true;

edt\_solvente.Enabled:=true;

img\_solucao\_saturada.Visible:=false;

img\_solucao\_insaturada.Visible:=false;

img\_solucao\_super.Visible:=false;

img\_menuzinho.Visible:=false;

img\_voltar\_jogos.Visible:=false;

img\_voltar\_menu.Visible:=false;

img\_jogardnv.Visible:=false;

edt\_soluto.Text:='';

edt\_solvente.Text:='';

Randomize;

begin

RD\_substancia:=RandomRange(1,30);

RD\_opcao:=RandomRange(1,20);

end;

if(RD\_opcao>10)then

begin

img\_solvente.Visible:=true;

edt\_soluto.text:=floattostr(v3[RD\_substancia]);

edt\_soluto.Enabled:=false;

edt\_solvente.Enabled:=true;

end

else

begin

img\_soluto.Visible:=true;

edt\_solvente.text:=floattostr(v4[RD\_substancia]);

edt\_soluto.Enabled:=true;

edt\_solvente.Enabled:=false;

end;

lbl\_elemento.caption:=v1[RD\_substancia];

lbl\_exemplo.caption:='dado: '+v2[RD\_substancia];

end;

procedure TForm5.img\_voltar\_jogosClick(Sender: TObject);

begin

close;

end;

procedure TForm5.img\_voltar\_menuClick(Sender: TObject);

begin

close;

end;

procedure TForm5.Timer\_SolubilidadeTimer(Sender: TObject);

begin

media.Stop;

panel\_video.Visible:=false;

end;

end.

//==============================================================================

// \*\* Sound

//------------------------------------------------------------------------------

// Módulo responsável pela execução de áudios diversos do programa.

//==============================================================================

unit Sound;

interface

uses MMSystem, SysUtils;

procedure Play\_SE( Dir : String; Kind : Byte = SND\_Async );

procedure Play\_Me( Dir : String; Kind : Byte = SND\_Async );

procedure Play\_Error();

procedure Play\_Decision( ID : Byte; Kind : Byte = SND\_Async );

procedure Play\_Cancel();

procedure Play\_Cursor( ID : Byte );

procedure Play\_Move();

implementation

//------------------------------------------------------------------------------

// \* Reproduz efeito sonoro

// Dir : Diretório do áudio

//------------------------------------------------------------------------------

procedure Play\_SE( Dir : String; Kind : Byte = SND\_Async );

begin

SndPlaySound( PWideChar(WideString('Sound/Se/' + Dir)), Kind );

end;

//------------------------------------------------------------------------------

// \* Reproduz efeito musical

// Dir : Diretório do áudio

//------------------------------------------------------------------------------

procedure Play\_ME( Dir : String; Kind : Byte = SND\_Async );

begin

SndPlaySound( PWideChar(WideString('Sound/Me/' + Dir)), Kind );

end;

//------------------------------------------------------------------------------

// \* Reproduz som de erro

//------------------------------------------------------------------------------

procedure Play\_Error();

begin

SndPlaySound('Sound/Se/Error', Snd\_Async );

end;

//------------------------------------------------------------------------------

// \* Reproduz som de decisão

// ID : Sufixo do áudio

//------------------------------------------------------------------------------

procedure Play\_Decision( ID : Byte; Kind : Byte = SND\_Async );

begin

SndPlaySound( PWideChar(WideString('Sound/Se/Decision' + IntToStr(ID))), Kind );

end;

//------------------------------------------------------------------------------

// \* Reproduz som de cancelamento

//------------------------------------------------------------------------------

procedure Play\_Cancel();

begin

SndPlaySound('Sound/Se/Cancel', Snd\_Async );

end;

//------------------------------------------------------------------------------

// \* Reproduz som de cursor

// ID : Sufixo do áudio

//------------------------------------------------------------------------------

procedure Play\_Cursor( ID : Byte );

begin

SndPlaySound( PWideChar(WideString('Sound/Se/Cursor' + IntToStr(ID))), Snd\_Async );

end;

//------------------------------------------------------------------------------

// \* Reproduz som de movimento

//------------------------------------------------------------------------------

procedure Play\_Move();

begin

SndPlaySound('Sound/Se/Move', Snd\_Async );

end;

end.

unit Tabela;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, Elementos, StdCtrls, pngimage, ExtCtrls, MMSystem, Propriedades,

JogoDistribuicao, jpeg, DB\_Integrator, Sound;

type

//==============================================================================

// \*\* TPeriodicTable

//------------------------------------------------------------------------------

// Esta classe é responsável pela tabela periódica

//==============================================================================

TPeriodicTable = class(TForm)

TabelaPeriodica: TImage;

Hidrogenio: TImage;

Helio: TImage;

Litio: TImage;

Berilio: TImage;

Boro: TImage;

Magnesio: TImage;

Aluminio: TImage;

Silicio: TImage;

Fosforo: TImage;

Enxofre: TImage;

Cloro: TImage;

Argon: TImage;

Potassio: TImage;

Calcio: TImage;

Escandio: TImage;

Titanio: TImage;

Vanadio: TImage;

Cromo: TImage;

Manganes: TImage;

Ferro: TImage;

Cobalto: TImage;

Niquel: TImage;

Cobre: TImage;

Zinco: TImage;

Galio: TImage;

Germanio: TImage;

Arsenio: TImage;

Selenio: TImage;

Bromo: TImage;

Kriptonio: TImage;

Sodio: TImage;

Fluor: TImage;

Neonio: TImage;

Oxigenio: TImage;

Nitrogenio: TImage;

Carbono: TImage;

Indio: TImage;

Cadmio: TImage;

Prata: TImage;

Paladio: TImage;

Rodio: TImage;

Rutenio: TImage;

Zirconio: TImage;

Tecnecio: TImage;

Molibdenio: TImage;

Niobio: TImage;

Bario: TImage;

Hafnio: TImage;

Cesio: TImage;

Rubidio: TImage;

Estanho: TImage;

Tantalio: TImage;

Estroncio: TImage;

Antimonio: TImage;

Xenomio: TImage;

Iodo: TImage;

Telurio: TImage;

Itrio: TImage;

Erbio: TImage;

Tulio: TImage;

Iterbio: TImage;

Lutecio: TImage;

Tungstenio: TImage;

Talio: TImage;

Mercurio: TImage;

Ouro: TImage;

Platina: TImage;

Iridio: TImage;

Osmio: TImage;

Renio: TImage;

Cerio: TImage;

Holmio: TImage;

Neodimio: TImage;

Praseodimio: TImage;

Disprosio: TImage;

Terbio: TImage;

Europio: TImage;

Promecio: TImage;

Gadolinio: TImage;

Samario: TImage;

Lantanio: TImage;

Ruterfordio: TImage;

Torio: TImage;

Dubnio: TImage;

Actinio: TImage;

Uranio: TImage;

Plutonio: TImage;

Protactinio: TImage;

Netunio: TImage;

Seaborgio: TImage;

Bohrio: TImage;

Hassio: TImage;

Meitnerio: TImage;

Darmstadio: TImage;

Roentgentum: TImage;

Ununbium: TImage;

Ununtrium: TImage;

Ununpentium: TImage;

Americio: TImage;

Curio: TImage;

Berquelio: TImage;

Californio: TImage;

Einstenio: TImage;

Ununquadium: TImage;

Fermio: TImage;

Laurencio: TImage;

Nobelio: TImage;

Ununhexio: TImage;

Mendelevio: TImage;

Ununoctium: TImage;

Ununseptium: TImage;

Branco2: TImage;

Radio: TImage;

Francio: TImage;

Radonio: TImage;

Astato: TImage;

Polonio: TImage;

Bismuto: TImage;

Chumbo: TImage;

Branco1: TImage;

Cursor: TImage;

CursorKeyboard: TTimer;

Nome: TLabel;

AtomicNumber: TLabel;

AtomicWeight: TLabel;

CursorMouse: TTimer;

Other: TTimer;

Botao: TImage;

Button: TTimer;

Timer1: TTimer;

procedure OnShow(Sender: TObject);

procedure OnClose(Sender: TObject; var Action: TCloseAction);

procedure CursorKeyboardTimer(Sender: TObject);

procedure CursorMouseTimer(Sender: TObject);

procedure OtherTimer(Sender: TObject);

procedure OnMouseEnter(Sender: TObject);

procedure OnMouseLeave(Sender: TObject);

procedure BotaoClick(Sender: TObject);

procedure OnMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure OnMouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure FormCreate(Sender: TObject);

procedure FormKeyUp(Sender: TObject; var Key: Word; Shift: TShiftState);

procedure TabelaPeriodicaClick(Sender: TObject);

private

procedure KeepPicture;

procedure MoveCursor( CursorSoundID : Byte = 1 );

procedure SetDirectionCursor( Direction : Byte );

procedure ChangeDescription();

public

{ Public declarations }

// Tabela periódica ativa ?

ActiveForm : Boolean;

// Cursor dos elementos

CursorIndex : Byte;

// Armazena o estado do cursor ao chamar outra janela

SaveCursorIndex : Byte;

// Simula o número atômico dos elementos químicos

AtomicPicture : Array[1..120] of Byte;

end;

var

// Instância da forma

PeriodicTable : TPeriodicTable;

// Resolução do monitor do usuário

Before\_Resolution : Array[1..2] of Integer;

// Todas as imagens dos elementos armazenadas em um conjunto

Picture : Array[1..121] of TImage;

// Botão de outras propriedades está sendo pressionado ?

ActiveButton : Boolean;

// Botão de outras propriedades está ativado ?

EnableButton : Boolean;

{ Variáveis de auxílio para bloquear pressionamento imediato de botões

assim que entrar no formulário }

Gambz : Boolean;

Gambz\_ : Array[1..2] of Integer;

// Veio de onde, irá para onde ?

WhereTo : Byte;

implementation

uses GameStoichiometric, Menu;

{$R \*.dfm}

//------------------------------------------------------------------------------

// \* Define a posição dos elétrons na tela

//------------------------------------------------------------------------------

procedure TPeriodicTable.ChangeDescription;

begin

{ Se o cursor estiver nos espaços de ligação para os Lantanídeos e os

Actínideos }

if ( CursorIndex = 75 ) or ( CursorIndex = 57 ) then

begin

{ Os seguintes labels tornam-se vazios: }

// Nome

Nome.Caption := '';

// Número atômico

AtomicNumber.Caption := '';

// Massa atômica

AtomicWeight.Caption := '';

// Torna o botão Outras Propriedades inacessível

EnableButton := false;

end

else

begin

{ Desenha os seguintes atributos de acordo com o cursor: }

// Nome

Nome.Caption := Elemento[AtomicPicture[CursorIndex]].Get\_Elemento(1);

// Número atômico

AtomicNumber.Caption := Elemento[AtomicPicture[CursorIndex]].Get\_Elemento(0);

// Massa atômica

AtomicWeight.Caption := Elemento[AtomicPicture[CursorIndex]].Get\_Elemento(3);

// Torna o botão Outras Propriedades acessível

EnableButton := true;

end;

end;

//------------------------------------------------------------------------------

// \* Muda a posição do Cursor

// CursorSoundID : Sufixo do áudio de movimento

//------------------------------------------------------------------------------

procedure TPeriodicTable.MoveCursor( CursorSoundID : Byte = 1 );

begin

// Se a tabela periódica não estiver ativa

if not ActiveForm then

// Sai do método

Exit();

// Executa efeito sonoro de cursor

Play\_Cursor(CursorSoundID);

// Estabelece nova posição X para o cursor

Cursor.Left := Picture[CursorIndex].Left - 3;

// Estabelece nova posição Y para o cursor

Cursor.Top := Picture[CursorIndex].Top;

// Altera a descrição para o atual elemento selecionado

ChangeDescription();

end;

//------------------------------------------------------------------------------

// \* Define a nova posição do cursor

// Direction : Ponto cardeal

//------------------------------------------------------------------------------

procedure TPeriodicTable.SetDirectionCursor(Direction: Byte);

begin

// Caso a Direção do Cursor seja:

case Direction of

6: CursorIndex := CursorIndex + 1; // Soma 1 ao cursor

4: CursorIndex := CursorIndex - 1; // Subtrai 1 ao cursor

end;

// Move o cursor de acordo com a nova posição

MoveCursor();

end;

procedure TPeriodicTable.TabelaPeriodicaClick(Sender: TObject);

begin

end;

//------------------------------------------------------------------------------

// \* Armazena todas as imagens dos símbolos dos elementos em um conjunto

//------------------------------------------------------------------------------

procedure TPeriodicTable.KeepPicture();

var I : Byte;

begin

// Imagens inseridas com perspectiva horizontal

Picture[1] := Hidrogenio;

Picture[2] := Helio;

Picture[3] := Litio;

Picture[4] := Berilio;

Picture[5] := Boro;

Picture[6] := Carbono;

Picture[7] := Nitrogenio;

Picture[8] := Oxigenio;

Picture[9] := Fluor;

Picture[10] := Neonio;

Picture[11] := Sodio;

Picture[12] := Magnesio;

Picture[13] := Aluminio;

Picture[14] := Silicio;

Picture[15] := Fosforo;

Picture[16] := Enxofre;

Picture[17] := Cloro;

Picture[18] := Argon;

Picture[19] := Potassio;

Picture[20] := Calcio;

Picture[21] := Escandio;

Picture[22] := Titanio;

Picture[23] := Vanadio;

Picture[24] := Cromo;

Picture[25] := Manganes;

Picture[26] := Ferro;

Picture[27] := Cobalto;

Picture[28] := Niquel;

Picture[29] := Cobre;

Picture[30] := Zinco;

Picture[31] := Galio;

Picture[32] := Germanio;

Picture[33] := Arsenio;

Picture[34] := Selenio;

Picture[35] := Bromo;

Picture[36] := Kriptonio;

Picture[37] := Rubidio;

Picture[38] := Estroncio;

Picture[39] := Itrio;

Picture[40] := Zirconio;

Picture[41] := Niobio;

Picture[42] := Molibdenio;

Picture[43] := Tecnecio;

Picture[44] := Rutenio;

Picture[45] := Rodio;

Picture[46] := Paladio;

Picture[47] := Prata;

Picture[48] := Cadmio;

Picture[49] := Indio;

Picture[50] := Estanho;

Picture[51] := Antimonio;

Picture[52] := Telurio;

Picture[53] := Iodo;

Picture[54] := Xenomio;

Picture[55] := Cesio;

Picture[56] := Bario;

Picture[57] := Branco1;

Picture[58] := Hafnio;

Picture[59] := Tantalio;

Picture[60] := Tungstenio;

Picture[61] := Renio;

Picture[62] := Osmio;

Picture[63] := Iridio;

Picture[64] := Platina;

Picture[65] := Ouro;

Picture[66] := Mercurio;

Picture[67] := Talio;

Picture[68] := Chumbo;

Picture[69] := Bismuto;

Picture[70] := Polonio;

Picture[71] := Astato;

Picture[72] := Radonio;

Picture[73] := Francio;

Picture[74] := Radio;

Picture[75] := Branco2;

Picture[76] := Ruterfordio;

Picture[77] := Dubnio;

Picture[78] := Seaborgio;

Picture[79] := Bohrio;

Picture[80] := Hassio;

Picture[81] := Meitnerio;

Picture[82] := Darmstadio;

Picture[83] := Roentgentum;

Picture[84] := Ununbium;

Picture[85] := Ununtrium;

Picture[86] := Ununquadium;

Picture[87] := Ununpentium;

Picture[88] := Ununhexio;

Picture[89] := Ununseptium;

Picture[90] := Ununoctium;

Picture[91] := Lantanio;

Picture[92] := Cerio;

Picture[93] := Praseodimio;

Picture[94] := Neodimio;

Picture[95] := Promecio;

Picture[96] := Samario;

Picture[97] := Europio;

Picture[98] := Gadolinio;

Picture[99] := Terbio;

Picture[100] := Disprosio;

Picture[101] := Holmio;

Picture[102] := Erbio;

Picture[103] := Tulio;

Picture[104] := Iterbio;

Picture[105] := Lutecio;

Picture[106] := Actinio;

Picture[107] := Torio;

Picture[108] := Protactinio;

Picture[109] := Uranio;

Picture[110] := Netunio;

Picture[111] := Plutonio;

Picture[112] := Americio;

Picture[113] := Curio;

Picture[114] := Berquelio;

Picture[115] := Californio;

Picture[116] := Einstenio;

Picture[117] := Fermio;

Picture[118] := Mendelevio;

Picture[119] := Nobelio;

Picture[120] := Laurencio;

Picture[121] := Laurencio;

// Do elemento Hidrogênio até o Bário

for I := 0 to 56 do

// Simula o número atômico

AtomicPicture[I] := I;

{ Torna a primeira casa em branco o elemento 0 ( Para que não

exiba informações ) }

AtomicPicture[57] := 0;

// Do elemento Lantânio até o Lutécio

for I := 58 to 74 do

// Simula o número atômico

AtomicPicture[I] := I + 14;

{ Torna a segunda casa em branco o elemento 0 ( Para que não

exiba informações ) }

AtomicPicture[75] := 0;

// Do elemento Háfnio ao Rádio

for I := 76 to 90 do

// Simula o número atômico

AtomicPicture[I] := I + 28;

// Do Actinio ao Laurêncio

for I := 91 to 105 do

// Simula o número atômico

AtomicPicture[I] := I - 34;

// Do elemento Rutenfórdio ao Unonoctium

for I := 106 to 120 do

// Simula o número atômico

AtomicPicture[I] := I - 17;

end;

//------------------------------------------------------------------------------

// \* Atualização do processo ( Movimento do Cursor da Tabela pelo Teclado )

// Sender : Timer responsável pela execução

//------------------------------------------------------------------------------

procedure TPeriodicTable.CursorKeyboardTimer(Sender: TObject);

begin

// Aumenta 1 ao controlador do clique não imediato

Gambz\_[1] := Gambz\_[1] + 1;

{ Se a seta direita for pressionada e o Cursor estiver em qualquer

elemento com número atômico simulado inferior ao 119 ( Unonoctium ) }

if ( GetAsyncKeyState(VK\_RIGHT) <> 0 ) and ( CursorIndex < 120 ) then

// Define direção direita para locomoção do Cursor

SetDirectionCursor(6)

{ Se a seta esquerda for pressionado e o Cursor estiver em qualquer

elemento com número atômico simulado superior ao 1 ( Hidrogênio ) }

else if ( GetAsyncKeyState(VK\_LEFT) <> 0 ) and ( CursorIndex >= 2 ) then

// Define direção esquerda para locomoção do Cursor

SetDirectionCursor(4)

{ Se a seta para baixa for pressionada e o Cursor estiver em qualquer

elemento com número atômico simulado inferior ao 106 ( Lutêncio ) }

else if ( GetAsyncKeyState(VK\_DOWN) <> 0 ) and ( CursorIndex < 106 ) then

begin

{ Se o cursor estiver num elemento com número atômico simulado superior

ao 18 ( Argonio ) }

if ( CursorIndex > 18 ) then

begin

{ Se o cursor estiver num elemento com número âtomico simulado

inferior ao 73 ( Frâncio ) }

if CursorIndex < 73 then

begin

// Move o cursor para o elemento Lantânio

CursorIndex := CursorIndex + 18;

end

{ Se o cursor estiver num elemento com número âtomico simulado

inferior ao 76 ( 2ª casa em branco ) }

else if ( CursorIndex < 76 ) then

// Move o cursor para o elemento Lantânio

CursorIndex := 91

{ Se o cursor estiver num elemento com número âtomico simulado

inferior ao 90 ( UnunSeptium ) }

else if ( CursorIndex < 90 ) then

// Move o cursor para Lutêncio

CursorIndex := CursorIndex + 16

{ Se o cursor estiver num elemento com número âtomico simulado

inferior ao 91 ( Unonoctium ) }

else if ( CursorIndex < 91 ) then

// Move o cursor para o Lutêncio

CursorIndex := 105

{ Se o cursor estiver num elemento da fileira aos Lantanídeos }

else if ( CursorIndex < 106 ) then

// Move para a fileira dos actinídeos

CursorIndex := CursorIndex + 15

end

// Se o cursor estiver num elemento simulado acima de 12 ( Magnésio )

else if ( CursorIndex > 12 ) then

// Define nova posição para o cursor

CursorIndex := CursorIndex + 18

// Se o cursor não estiver no Hidrogênio

else if ( CursorIndex > 1 ) then

// Define nova posição para o cursor

CursorIndex := CursorIndex + 8

else

// Envia o cursor até o Lítio

CursorIndex := 3;

// Movimenta o cursor

MoveCursor();

end

{ Se a tecla para cima foi pressionado e o cursor não estiver nem

no Hidrogênio nem no Hélio }

else if ( GetAsyncKeyState(VK\_UP) <> 0 ) and ( CursorIndex > 2 ) then

begin

// Se o cursor estiver nas fileira dos Actinídeos

if ( CursorIndex > 105 ) then

CursorIndex := CursorIndex - 15

// Se o cursor estiver nas fileira dos Lantanídeos

else if ( CursorIndex > 90 ) then

CursorIndex := CursorIndex - 16

// Se o cursor estiver num número atômico simulado acima de 30 ( Gálio )

else if ( CursorIndex > 30 ) then

CursorIndex := CursorIndex - 18

// Se o cursor estiver num número atômico simulado acima de 20 (Cálcio)

else if ( CursorIndex > 20 ) then

Exit() // I'm evil

// Se o cursor estiver num número atômico simulado acima de 9 ( Nêonio )

else if ( CursorIndex > 9 ) then

CursorIndex := CursorIndex - 8

// Se o cursor estiver num número atômico simulado acima de 3 ( Lítio )

else if ( CursorIndex > 3 ) then

Exit()

else

// Define nova posição do cursor: Hidrogênio

CursorIndex := 1;

// Move o Cursor para a posição definida

MoveCursor();

end;

end;

//------------------------------------------------------------------------------

// \* Atualização do processo ( Movimento do Cursor da Tabela pelo Mouse )

// Sender : Timer responsável pela execução

//------------------------------------------------------------------------------

procedure TPeriodicTable.CursorMouseTimer(Sender: TObject);

var I, J, K : Byte;

begin

// Se botão esquerdo do mouse pressionado

if GetAsyncKeyState(VK\_LButton) <> 0 then

// Para todos os elementos químicos

for I := 1 to 120 do

// Para todos os pixels da largura da imagem do símbolo dos elementos

for J := 0 to Picture[I].Width do

// Para todos os pixels da altura da imagem do símbolo dos elementos

for K := 0 to Picture[I].Height - 1 do

// Se o cursor foi pressionado na posição de algum dos elementos

if ( Mouse.CursorPos.X = Picture[I].Left + J ) and

( Mouse.CursorPos.Y = Picture[I].Top + K ) then

if ( CursorIndex <> I ) then

begin

// Define nova posição para o cursor

CursorIndex := I;

// Move o cursor

MoveCursor(2);

end

end;

//------------------------------------------------------------------------------

// \* Processo ao clicar no botão 'Outras Propriedades'

// Sender : Timer responsável pela execução

//------------------------------------------------------------------------------

procedure TPeriodicTable.BotaoClick(Sender: TObject);

begin

// Se Outras Propriedades não for acessível

if not EnableButton then

begin

// Reproduz efeito sonoro de erro

Play\_Error();

// Sai do método

Exit();

end;

// Reproduz efeito de confirmação

Play\_Decision(1);

// Tabela periódica é desativada

ActiveForm := False;

// Armazena o estado do cursor

SaveCursorIndex := CursorIndex;

// Avisa à janela Outras Propriedades que está chamando-a

WindowPropertie.WentFromTable;

// Exibe a janela de propriedades

WindowPropertie.Show();

// Esconde a tabela

PeriodicTable.Hide();

end;

//------------------------------------------------------------------------------

// \* Processo ao fechar a tabela periódica

//------------------------------------------------------------------------------

procedure TPeriodicTable.OnClose(Sender: TObject; var Action: TCloseAction);

begin

// Finaliza o programa

DataModule1.DataModuleDestroy(Self);

end;

//------------------------------------------------------------------------------

// \* Processo ao apertar o botão esquerdo do mouse

// Sender : Componente responsável pela execução (Botão Outras Propriedades)

// Button : Tipo de botão ( Esquerdo, Direito e etc .. )

// Shift : Estado da tecla CTRL

// X : Posição X do Cursor dentro do Botão

// Y : Posição Y do Cursor dentro do Botão

//------------------------------------------------------------------------------

procedure TPeriodicTable.OnMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

begin

// Altera a imagem do Botão

Botao.Picture.LoadFromFile('Graphics\Tabela\Botão\Pressionado.png');

// Armazena a confirmação de que o botão está sendo pressionado

ActiveButton := true;

end;

//------------------------------------------------------------------------------

// \* Processo executado ao adentrar as coordenadas do botão 'Outras Propriedades'

// Sender : Componente responsável pela execução

//------------------------------------------------------------------------------

procedure TPeriodicTable.OnMouseEnter(Sender: TObject);

begin

// Se o botão estiver sendo pressionado

if ActiveButton then

// Sai do método

Exit();

// Altera a imagem

Botao.Picture.LoadFromFile('Graphics\Tabela\Botão\MouseSob.png')

end;

//------------------------------------------------------------------------------

// \* Processo executado ao sair dos limites das coordenadas do botão

// Sender : Componente responsável pela execução

//------------------------------------------------------------------------------

procedure TPeriodicTable.OnMouseLeave(Sender: TObject);

begin

// Se o botão estiver sendo pressionado

if ActiveButton then

// Sai do método

Exit();

// Altera a imagem

Botao.Picture.LoadFromFile('Graphics\Tabela\Botão\Normal.png');

end;

//------------------------------------------------------------------------------

// \* Processo executado ao soltar o botão esquerdo do mouse sobre o botão

// Sender : Componente responsável pela execução

// Button : Tipo de botão ( Esquerdo, Direito e etc .. )

// Shift : Estado da tecla CTRL

// X : Posição X do Cursor dentro do Botão

// Y : Posição Y do Cursor dentro do Botão

//------------------------------------------------------------------------------

procedure TPeriodicTable.OnMouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

begin

// Armazena a confirmação de que o botão não está sendo pressionado

ActiveButton := false;

end;

//------------------------------------------------------------------------------

// \* Criação do Fomulário

// Sender : Componente responsável pela execução

//------------------------------------------------------------------------------

procedure TPeriodicTable.FormCreate(Sender: TObject);

begin

// Armazenas as imagens dos símbolos dos elementos

KeepPicture();

// Adapta o formulário aos demais

PerformForm(Self);

// Atribuição inicial às variaveis

WhereTo := 0;

CursorIndex := 1;

ActiveForm := False;

SaveCursorIndex := 1;

EnableButton := true;

// Exibe as informações do Hidrogênio como iniciais

Nome.Caption := Elemento[1].Get\_Elemento(1);

AtomicNumber.Caption := Elemento[1].Get\_Elemento(0);

AtomicWeight.Caption := Elemento[1].Get\_Elemento(3);

end;

//------------------------------------------------------------------------------

// \* Ao soltar qualquer tecla pressionada

// Sender : Componente responsável pela execução

// Key : Tecla

// Shift : Estado da tecla CTRL

//------------------------------------------------------------------------------

procedure TPeriodicTable.FormKeyUp(Sender: TObject; var Key: Word;

Shift: TShiftState);

begin

{ Se ainda não foi estabelecido de onde veio e o bloqueio de tecla

estiver ativo }

if ( WhereTo = 0 ) or ( Gambz\_[1] < 10 ) then

// Sai do método

Exit();

// Se a tecla Escape é pressionada

if GetAsyncKeyState(VK\_ESCAPE) <> 0 then

begin

// Executa efeito sonoro de cancelamento

Play\_Cancel();

// Define Tabela periódica como não ativa

ActiveForm := False;

// Torna a tabela inacessível

Tabela.PeriodicTable.Enabled := false;

// Torna-a inacessível

Tabela.PeriodicTable.Visible := False;

// Armazena o estado do cursor

SaveCursorIndex := CursorIndex;

// Checa de onde a tabela foi chamada

case WhereTo of

// Envia a cena para a última janela

1 : EletronicDistribution.Show;

2 : CalculoEstequiometrico.Show();

3 : Inicio.Show();

end;

// Esconde a tabela periódica

PeriodicTable.Hide();

// Define como tabela não chamada

WhereTo := 0;

end;

end;

//------------------------------------------------------------------------------

// \* Processo de exibição da tabela

// Sender : Componente responsável pela execução

//------------------------------------------------------------------------------

procedure TPeriodicTable.OnShow(Sender: TObject);

begin

// Torna a janela ativa

ActiveForm := True;

// Define novo cursor para o mouse

Screen.Cursor := 5;

end;

//------------------------------------------------------------------------------

// \* Update

// Sender : Componente responsável pela execução

//------------------------------------------------------------------------------

procedure TPeriodicTable.OtherTimer(Sender: TObject);

begin

// Se a tecla 'Enter' estiver pressionada

if GetAsyncKeyState(VK\_Return) <> 0 then

// Se estiver na 1ª casa branca

if ( CursorIndex = 57 ) then

begin

// Define posição do cursor como sendo o Lantânio

CursorIndex := 91;

// Move o cursor para posição definida

MoveCursor();

end

// Se estiver na 2ª casa branca

else if ( CursorIndex = 75 ) then

begin

// Define posição do cursor como sendo o Actíneo

CursorIndex := 106;

// Move o cursor para a posição definida

MoveCursor();

end;

end;

end.

unit Tip;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, pngimage, ExtCtrls;

type

TPassWordFaq = class(TForm)

BackGround: TImage;

Name: TEdit;

Image3: TImage;

Okay: TImage;

Tip: TMemo;

Back: TImage;

procedure UpdatePicture( Who : Byte; ID : Byte );

procedure OnMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure OnMouseEnter(Sender: TObject);

procedure OnMouseLeave(Sender: TObject);

procedure OnMouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure BackClick(Sender: TObject);

procedure OnKeyPress(Sender: TObject; var Key: Char);

procedure OnCreate(Sender: TObject);

procedure OnShow(Sender: TObject);

procedure OkayClick(Sender: TObject);

procedure BackGroundClick(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

PassWordFaq: TPassWordFaq;

implementation

uses Sound, DB\_Integrator;

{$R \*.dfm}

procedure TPassWordFaq.BackClick(Sender: TObject);

begin

// Executa efeito sonoro de cancelamento

Sound.Play\_Cancel();

// Fecha a dica de senha

Self.Close();

end;

procedure TPassWordFaq.BackGroundClick(Sender: TObject);

begin

end;

procedure TPassWordFaq.OkayClick(Sender: TObject);

var I : Byte;

begin

if Name.Text = '' then

begin

Sound.Play\_Error;

Application.MessageBox('Insira um usuário.', 'Login', MB\_ICONWARNING );

Exit;

end;

// Limpa comandos SQL

DB\_Integrator.DataModule1.ADOQuery2.SQL.Clear;

// Adiciona comando SQL ( Obtém a quantidade de usuários )

DB\_Integrator.DataModule1.ADOQuery2.SQL.Add('Select Count(\*) as ID from' +

' Professor');

// Ativa o SQL

DataModule1.Adoquery2.Active := True;

// Executa comandos SQL

DataModule1.Adoquery2.ExecSQL;

// Para todos os usuários

for I := 1 to DataModule1.ADOQuery2.FieldByName('ID').AsInteger do

begin

// Limpa comandos SQL

DataModule1.ADOQuery2.SQL.Clear();

// Adiciona comando SQL ( Para todos usuários )

DataModule1.ADOQuery2.SQL.Add('Select \* from Professo' +

'r where ID = ' + IntToStr(I));

// Ativa SQL

DataModule1.ADOQuery2.Active := True;

// Executa SQL

DataModule1.ADOQuery2.ExecSQL;

// Se já houver a equação não balanceada

if Crypt(Name.Text) = DataModule1.ADOQuery2.FieldByName('Usuario').AsString then

begin

Tip.Text := DataModule1.ADOQuery2.FieldByName('Dica').AsString;

// Sai da função

Exit;

end;

end;

end;

procedure TPassWordFaq.OnCreate(Sender: TObject);

begin

DB\_Integrator.PerformForm( Self );

Tip.Clear();

end;

procedure TPassWordFaq.OnKeyPress(Sender: TObject; var Key: Char);

begin

// Se a tecla Escape é pressionada

if Key = #27 then

begin

Key := #0;

Sound.Play\_Cancel();

Self.Close();

end

else if Key = #13 then { Tecla Enter }

begin

// Impede Bip

Key := #0;

// Executa efeito sonoro de próximo

Play\_Move();

// Move o cursor até o próximo Campo

Perform(WM\_nextdlgctl,0,0);

end;

end;

procedure TPassWordFaq.OnMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

begin

if Sender = Okay then

UpdatePicture(1, 3)

else if Sender = Back then

UpdatePicture(2, 3)

end;

procedure TPassWordFaq.OnMouseEnter(Sender: TObject);

begin

if Sender = Okay then

UpdatePicture(1, 2)

else if Sender = Back then

UpdatePicture(2, 2)

end;

procedure TPassWordFaq.OnMouseLeave(Sender: TObject);

begin

if Sender = Okay then

UpdatePicture(1, 1)

else if Sender = Back then

UpdatePicture(2, 1)

end;

procedure TPassWordFaq.OnMouseUp(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

begin

if Sender = Okay then

UpdatePicture(1, 1)

else if Sender = Back then

UpdatePicture(2, 1)

end;

procedure TPassWordFaq.OnShow(Sender: TObject);

begin

Tip.Clear();

Name.Clear();

end;

procedure TPassWordFaq.UpdatePicture( Who : Byte; ID: Byte);

begin

case Who of

1: Okay.Picture.LoadFromFile('Graphics/Tip/Okay\_' + IntToStr(ID) + '.png');

2: Back.Picture.LoadFromFile('Graphics/Tip/Back\_' + IntToStr(ID) + '.png');

end;

end;

end.