Alchemilla

Danilo Makoto Ikuta[1], **Gabriel Nopper Silva Roddrigues**[1]

[1]Centro Universitário Senacário Senac
04.696-000 – São Paulo – SP – Brasil

danilo0195@hotmail.com, ganopper@gmail.com

Resumo. *Este artigo tem como finalidade descrever o jogo Alchemilla: seus conceitos (tanto como um jogo eletrônico quanto pedagógico), código e resultados. Como resultado do trabalho descrito neste artigo, Alchemilla é um jogo feito na linguagem C, usando a biblioteca gráfica Allegro 5. O jogo é focado no ensino de química(reações) voltada ao Ensino Médio, de forma a servir como uma ferramenta pós-aula para fixação de conceitos.*

Abstract. *This article has the objective of describind the game Alchemilla: It's concepts(both of a electronic game and a form of teaching), codes and results. As the product of our work described in this article, Alchemilla is a game programmed in C language, with usage of the Allegro5 graphic library. The game is focused on the teaching of chemestry(And it's reactions) for High School, shaped to be a post-class tool for concept reinforcement.*

1. Introdução

O artigo contempla dois aspectos principais do Alchemilla(realizado como PI-2 do 2º período do curso de Bacharelado em Ciência da Computação do Centro Universitário SENAC, 2º semestre de 2013), além da divulgação do resultado como jogo. Os conceitos: os conceitoes de um jogo eletrônico que o Alchemilla usa para que ele não seja visto pelo público-alvo como apenas um famigerado "jogo educativo"e ao mesmo tempo os conceitos pedagógicos que permitem o aprendizado dentro do jogo. Consecutivamente, é apresentado a estrutura do código do jogo, divididas em: funções de leitura de entradas de texto, funções que necessitam do uso da biblioteca Allegro 5(menus, mouse e áudio), outros arquivos de código e entradas de texto.

2. Conceitos

2.1. O Conceito do Jogo Eletrônico

Jogos eletrônicos são uma forma comum de entretenimento nos dias de hoje. Existem de diversos tipos, voltados para diversos publicos alvos, e de variados periodos de tempo. Isto torna o Jogo Eletronico, na atualidade, a principal atividade para entretenimento rapido do mundo, ou uma das principais atividades para entretenimento de longo periodo para um específico nicho da sociedade. Eles são, entretanto, uma tendencia. Dia após dia, os jogos eletronicos popularizam-se, especialmente após a revolução dos smartphones e dispositivos móveis em geral. Os jogos eletrnônicos tem sido melhor recebidos e mais frequentemente usados, definindo um formato característico que estrutura maioria dos jogos eletronicos. Uma das mais importantes caracteristicas, é a de interação. Os jogos são até mesmo uma forma de mídia, mas cuja interação é o diferencial de outros. Até

mesmo em sites de caracter ”sério”, como a febraban, adota-se o uso de jogos eletronicos como um meio de mídia devido à sua capacidade de cativar o usuário, uma vez que proporciona prazer ao jogador, mesmo que às vezes em assunto não relacionado a diversão, simplesmente pela interação.

2.2. O Conceito do Jogo Educativo

O conceito do Aprender em meio ao lazer é algo extremamente antigo. Desde a Antiguidade, Platão espalhava a ideologia do Aprender Brincando, sugerindo que o aprendizado e a assimilação de informações em geral era mais simples e eficaz quando está era distraida, prazerosa e cuja exerção fosse o mais leve e simples possivel. Tal conceito hoje é usado amplamente, os Jogos Educacionais sendo constantemente elaborados e aplicados no ensino de conceitos espeficificos, conhecimentos de caracter cultural e histórico, ou aprendizagem de alguma habilidade especifica ou linguagem. Muito embora sejam mais recorrentes as aplicações para o publico infantil, é encontrado para todos os publicos, nas mais diversas plataformas, indo de tabuleiros e cartas ao Jogo Eletronico.

2.3. O Educar no Jogo Educativo

A mesclagem de dois conceitos é o objetivo deste projeto. O Alchemilla visa aplicar em um jogo eletronico com o maior grau de interatividade e controle do jogador, para repassar conceitos técnicos de Química sem que tal seja cansativo. O jogo é baseado em um sistema intuitivo de escolhas pseudoaleatórias, onde a intenção é causar a impressão da liberdade de escolhas entre qualuqer elemento ara o jogador sem pensar, entertanto ele rapidamente passa a procurar informações para guiar suas escolhas, como visto verdadeiro no teste realizado durante a E-comp e a apresentação do projeto, onde sete de oito jogadores procuraram ajuda mesmo no tutorial e nas fases em seguida. O jogador possui a escolha de misturar elementos à vontade, no intuito de criar substancias e adicioná-las à sua lista de ingredientes - Entretanto existem apenas combinações corretas, e a escolha aleatória torna-se frustrante, fazendo com que o jogador tenha interesse em ler informações do elemento com o clique direito para guiá-lo pelas fases.

Esta jogabilidade resulta numa atividade simples, pouco cansativa, entusiasmante, e que passa conceitos quimicos através de assimilação e repetição, ou por pesquisa impulsionada por vontade própria do jogador.

3. O Código

Esclareçendo um equívoco de nomenclatura química: no código e nesta seção, os ”reagentes”são na verdade, os catalisadores de reações químicas e os ”elementos”os reagentes.

O uso de uma lista dinâmica para os elementos e arrays estáticos para os reagentes: o número de elementos varia muito dependendo da fase a ser criada, ao contrário do número de reagentes, que costuma ser sempre estável.

3.1. Leitura de Entradas

Conjunto de funções envolvendo leitura de entrada de arquivos.

```
1 void filenamesgen();
```

Utilizado ao carregar uma fase, utilizando a variável global *fase*, gera os caminhos para as entradas de texto da fase correspondente (exceto *Entradas/reaglist.txt*, que é único para todas as fases).

```
1 int fgetline(FILE *fp, char s[], int lim);
```

Função que, dentro de determinado limite *lim*, recebe uma linha de uma entrada de texto e retorna uma string formatada com '\n' seguido de '\0'.

```
1 void useElement(int elem, lista *menu);
```

Efetua o envio de informações de um elemento da lista à uma das entradas de elementos para reação. Primeiramente verifica se as duas entradas de elementos estão cheios, sendo que se houver algum disponível, irá procurar dentro da lista *menu* por um elemento com o mesmo número que o parâmetro *elem*. Se o elemento for encontrado na lista, ocorrerá a atribuição do número e nome do elemento à *in1* e *in1name[30]* caso o primeiro espaço para elemento na reação estiver vazio, senão, a mesma atribuição será feita à *in2* e *in2name[30]* respectivamente.

```
1 void insert(int num, lista *menu, int type);
```

Chamada quanto um novo elemento deve ser inserido na lista *menu*(tanto durante o carregamento de uma fase quanto quando o jogador adiciona um novo elemento). Primeiramente, insere um elemento no final da lista, então abre o arquivo de texto e procura pelas informações do elemento *num* e insere no novo elemento. Além disso, dependendo do tipo *type* do elemento(normal = 1 ou bônus = 2), atribui uma mensagem diferente para ser vista pelo jogador. Essa última funcionalidade é desativada se a função for chamada durante a inicialização da fase (passa-se *type* como 0).

```
1 void start_menu(lista *menu);
```

Carrega os parâmetros iniciais de uma fase, a partir do *startlist* definido por *filenamesgen*. Começa carregando os elementos iniciais da fase(via **insert**), depois o objetivo e por fim, os reagentes(através de **nomeia_reag**).

```
1 void nomeia_reag(int reagente, int i);
```

Através do número do reagente(*reagente*) e de seu índice no array de reagentes *reagentes[10]*, procura em *Entradas/reaglist.txt* pelo reagente e copia seu nome para *reagname[5][30]*, que contém o nome de cada reagente no mesmo índice do array de valor.

```
1 void nomeia(int num, int casa);
```

Utilizado na função **checagem**, procura o nome do elemento de número *num* na *itemlist* da fase e copia para a string da *casa* determinada(*out1name* ou *out2name*). Diferente da função **insert**, uma vez que o elemento só é inserido na lista quando se clica sobre ele em uma dos espaços de saída de reação.

```
1 void info_reag(int reagente);
2 void info_elem(int elem);
```

Ambas as funções(para reagentes e elementos, respectivamente) buscam em uma entrada de texto(*infolist* e *reaglist*) nome, fórmula(se houver) e texto informativo do elemento/reagente de número correspondente ao parâmetro da função, armazenando respectivamente em *infoname[30]*, *infosymbol[30]* e *infotext[500]*.

```
1 int checagem(int in1, int in2, int reag, lista *menu);
```

Responsável por verificar os resultados de uma reação, primeiro ordena os valores de *in1* e *in2*, então procura em *checklist* da fase se a combinação de *in1*, *in2* e *reag* corresponde a alguma das reações possíveis listadas no .txt. Caso essa combinação resulte em algum elemento, a função **nomeia** é chamada para *out1* e caso a reação resulte em dois elementos, **nomeia** é chamada para *out2*. Caso a entrada de texto também informe que há um elemento bônus, **insert** é chamada, inserindo o elemento bônus diretamente na lista.

```
1 lista inicializa_lista();
2 void termina_lista(lista *l);
```

A primeira função cria o primeiro elemento de uma lista, enquanto a segunda libera o espaço consumido pelos elementos de uma lista.

3.2. Allegro 5

3.2.1. Funções de Display

Cada uma das ”telas”(interfacecs diferentes) podem ser divididas em 3 partes: inicialização(carrega os recursos necessários para o display), execução(a tela propriamente dita), encerramento(libera o espaço que os recursos para a fase consomem). A hierarquia das interfaces é descrita no esquema abaixo:

```
1 mainMenu
2     selectMenu
3         gameMenu
4             infoMenu
5     options
6     creditos
7     intro
```

Cada uma das interface pode utilizar recursos da interface precedente. Apesar da introdução ser executada antes do menu principal, como o menu é a primeira interface interativa, hierarquicamente, o mainMenu deve ser carregado primeiro, pois é ele que carrega todos os recursos necessários para o Allegro 5 funcionar. A introdução é carregada, executada e encerrada dentro do menu principal, antes do loop de execução do menu.

Todas as interfaces interativas funcionam com um loop de execução que detecta eventos e dependendo do tipo e posição da tela em que ocorreu, o evento ativa alguma ação ou não. Ao final de cada loop, é executado uma verificação para que se, na ausência de eventos, como o evento anterior passa a repetir o último evento, as funções de display não reconheçam essas repetições(*debouncer*). Além disso, se uma interface abre outra interface, esta nova interface, ao ser encerrada, envia um sinal de retorno(int) que determina a causa do encerramento: 1 para clique no ícone de fechar janela, 0 para

retornar à interface anterior(encerra somente a última interface), ou -1, indicando que a interface não pode ser carregada em decorrência de algum erro na inicialização desta.

```
1 bool mainInit();
2 int mainMenu();
3 void mainFinish();
```

Funções responsáveis pelo menu principal do jogo. No caso de **mainInit()**, esta inicializa todos os recursos de inicialização do Allegro 5 e seus componentes(*add-ons*, janela, mouse), os recursos do menu principal, e a introdução, que é executada na inicialização do menu principal. Não há grandes diferenças na arquitetura de código em cada conjunto de funções de display interativos.

```
1 bool introInit();
2 bool intro();
3 void introFinish();
4 void fadeInOut(ALLEGRO_BITMAP *img, int velocidade, int restTime);
```

Encarregadas de inicializar, executar e finalizar a introdução do jogo. Vale ressaltar que **intro** é uma chamada simplificada de **fadeInOut**, que recebe como parâmetros a imagem(**img*) a ser usada na sequência de fade-in, espera e fade-out, a velocidade do fade-in e do fade-out(*velocidade*) e a duração da espera entre os fades(*restTime*). No caso de **intro**, os fades duram aproximadamente 0,5 segundo e a espera configurada para 2 segundos.

```
1 bool selectInit();
2 int selectMenu();
3 void selectFinish();
```

Funções relativas à tela de seleção de fase.

```
1 bool gameInit();
2 int gameMenu(int NSNumeroDaFase);
3 void gameFinish();
```

Conjunto de funções resposáveis pelo display do jogo propriamente dito. Interface mais extensa e complpexa em termos de código. Não só por causa da quantidade de possibilidades de interação disponíveis, mas também por causa da interação com o conjunto de funções de leitura de entrada de arquivos, a variação do tamanho da fonte dependendo do tamanho de uma string e a forma na qual foi feita a lista de elementos(uma hierarquia que deve impedir a interação nos espaços em que nenhum elemento foi alocado, caso contrário, ocorre falha de segmentação).

```
1 bool infoInit();
2 int infoMenu();
3 void infoFinish();
```

Funções da tela de informações para elementos/reagentes. Exibe o nome(*infoname*), a fórmula(*infosymbol*) e um texto informativo(*infotext*).

O diferencial desta interface e o tratamento da variável *infotext*, que é dividida em até cinco arrays, cada uma representando uma linha que será exibida na tela.

```
1 bool creditoInit();
2 int creditos();
3 void creditoFinish();
```

Responsáveis pela tela de créditos do jogo.

```
1 bool optionsInit();
2 int options();
3 void optionsFinish();
```

Menu de opções do jogo: liga/desliga música de fundo e efeitos sonoros.

3.2.2. Funções de Áudio e Mouse

```
1 bool checkSair(ALLEGRO_EVENT *evento, ALLEGRO_EVENT_QUEUE *fila);
```

Esta função verifica se o jogador clicou no botão de fechar janela. Retorna *true* caso sim, senão retorna *false*.

```
1 bool checkBotao(float xa, float xb, float ya, float yb, ALLEGRO_EVENT *
    evento, ALLEGRO_EVENT_QUEUE *fila);
```

Verifica se o cursor do mouse está sobre determinada posição determinada pelos pontos *xa, xb, ya, yb*. Retorna *true* caso sim, senão retorna *false*.

```
1 bool clickBotaoL(float xa, float xb, float ya, float yb, ALLEGRO_EVENT
    *evento, ALLEGRO_EVENT_QUEUE *fila);
2 bool clickBotaoR(float xa, float xb, float ya, float yb, ALLEGRO_EVENT
    *evento, ALLEGRO_EVENT_QUEUE *fila);
```

De forma semelhante à função anterior, essas duas checam se houve um evento de clique do mouse(botão esquerdo e direito, respectivamente).

```
1 void playSample(ALLEGRO_SAMPLE *sample);
```

Função que quando chamada, verifica se o evento é único ou se ele é um evento ”repetido”em decorrência do efeito de *bouncing*. Somente caso o evento seja único, toca o *sample* especificado.

3.3. Outros Códigos

header.h: utilizado por todos os arquivos .c, importa todas as bibliotecas utilizadas no Alchemilla. *struct.h*: declaração do tipo responsável por guardar as informações de um elemento(número, nome, tipo e próximo elemento). *main.c*: arquivo a ser aberto na execução do jogo, executa **mainMenu**, e consequentemente, o resto do jogo.

3.4. Formatação dos Arquivos de Texto(.txt)

Caso a fase possua um dígito, por questões de formatação,adicionar um zero à esquerda. Dessa forma, o jogo suporta até 100 fases(0 à 99), bastando modificar o menu de seleção de fases.

O número do elemento está vinculado ao seu nome, portanto nos diferentes arquivos de texto, o mesmo elemento deve ter o mesmo número.

3.4.1. info(número da fase).txt

```
A primeira linha deve ser vazia para que o primeiro fgetline de
    info_elem retire caracteres indesejados.
Nome do Elemento
Formula do Elemento(se nao houver, pode ser uma linha vazia)
Texto informativo referente ao elemento
-       \\Uso de traco para separar os elementos.
Nome do Elemento
Formula do Elemento(se nao houver, pode ser uma linha vazia)
Texto informativo referente ao elemento
0       \\Ou outro caractere que nao seja numero para indicar o final
    do arquivo
```

3.4.2. check(número da fase).txt

```
0
Numero de elemento(menor)
Numero de elemento(maior)
Numero do reagente
Numero do primeiro elemento resultante
Numero do segundo elmento resultante(se nao houver, use 0)
Numero do elemento bonus(se nao houver, use 0)
-1      \\Separador entre reacoes
Numero de elemento(menor)
Numero de elemento(maior)
Numero do reagente
Numero do primeiro elemento resultante
Numero do segundo elemento resultante(se nao houver, use 0)
Numero do elemento bonus(se nao houver, use 0)
-2      \\Indicador de fim de arquivo
```

3.4.3. reaglist.txt

```
Numero do reagente
Formula do reagente
Texto informativo
Numero do reagente
Formula do reagente
Texto informativo
0       \\Indicador de fim de arquivo
```

3.4.4. stage(número da fase).txt

```
00
-1
Numero do elemento
Nome do elemento
Tipo do elemento(por padrao, use 1)
-1      \\Separador entre elementos
```

```
7  Numero do elemento
8  Nome do elemento
9  Tipo do elemento(por padrao, use 1)
10 -2       \\Indicador de fim de arquivo
```

3.4.5. start(número da fase).txt

```
1 0
2 Numero dos elementos iniciais(separar por quebra de linha)
3 -1
4 Numero do elemento objetivo
5 Nome do elemento objetivo
6 -1
7 Numero dos reagentes liberados(separar por quebra de linha)
8 -1
```

4. Resultados

Nesta seção será demostrado o funcionamento do Alchemilla.

Logo após a introdução aparecerá o menu principal, como visto na Figura 1.

Figura 1. Menu principal.

A Figura 2 mostra o menu de opções, onde se pode ligar/desligar a música de fundo e efeitos sonoros(Figura 2).

Ao clicar em jogar, o jogador poderá escolher a fase(na versão atual, 6 fases), como na figura abaixo(Figura 3):

Figura 2. Menu de opções.

Figura 3. Tela de seleção de fase.

A fase 1 serve como um tutorial, fazendo uso de cores, algo mais intuitivo que elementos químicos para mostrar o funcionamento das mecânicas de jogo(Figura 4).

Ao clicar em azul e depois em amarelo, como saída aparece a cor verde e o jogador também ganha uma cor bônus(pó vermelho)(Figura 5).

Quando o jogador recebe um elemento de saída, este somente irá para a lista

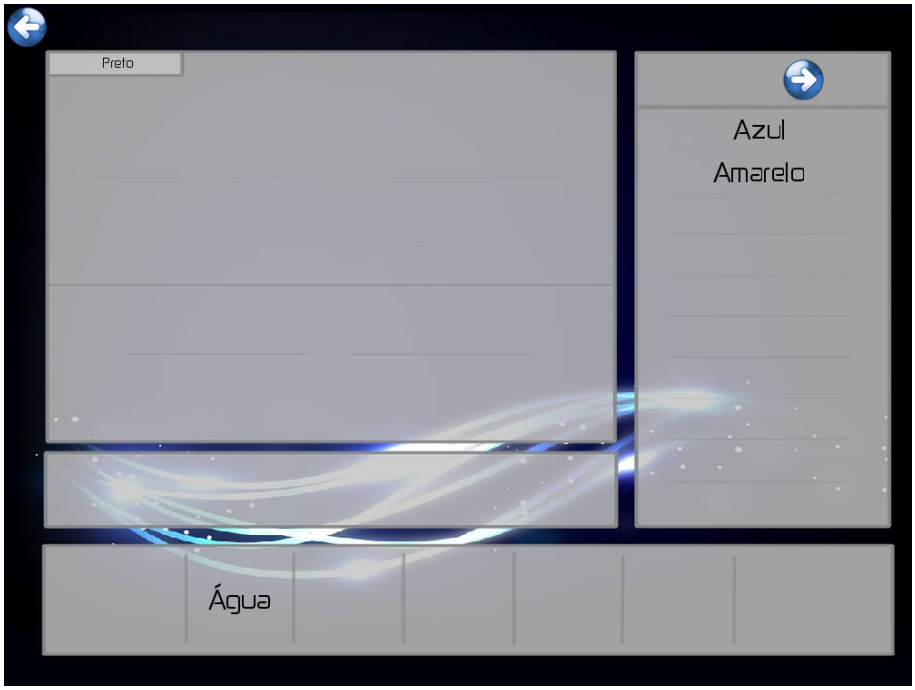

Figura 4. Início da fase 1(tutorial)

Figura 5. Mistura de azul e amarelo

quando o jogador clicar sobre o elemento(elementos bônus são adicionados automaticamente à lista)(Figura 6).

Para alcançar o objetivo, que é fazer a cor preta(exibido no canto superior esquerdo da tela, abaixo do botão de voltar), o mais óbvio é continuar fazendo misturas, utilizando o pó vermelho(Figura 7).

Figura 6. Cor verde adicionada à lista

Figura 7. Jogador tenta misturar amarelo e pó vermelho

Supostamente essa combinação deveria resultar em laranja. Mas deve-se atentar que vermelho é um pó. Como meio de transmitir o conteúdo no jogo, para solucionar esse problema, o jogador pode clicar com o botão direito do mouse sobre o elemento pó vermelho. Isso irá abrir uma janela com informações sobre o elemento(funciona também com os catalisadores)(Figura 8).

Figura 8. Tela de informações sobre pó vermelho.

Como forma de introduzir a mecânica de catalisadores, constata-se de que para o pó vermelho se misturar, o jogador precisará adicionar água na combinação para que ela funcione(Figura 9).

Figura 9. Com a adição de água, acontece a mistura.

Continuamente descobrindo novas cores, o jogador alcançará as cores necessárias para se fazer a cor preta e completar a fase(no caso da fase 1, por se tratar de um tutorial,

o caminho por entre as reações é bastante linear)(Figura 10).

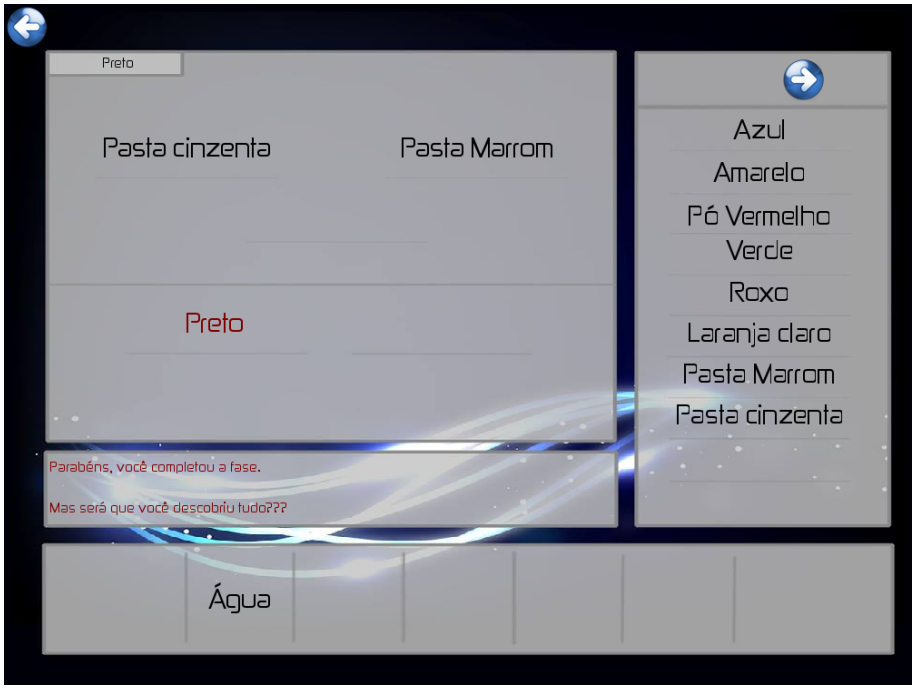

Figura 10. Cor preta descoberta. Tutorial completo

As fases seguintes, além de se tratarem realmente de química, terão mais possibilidades a serem exploradas na fase, de forma que nem sempre um caminho irá levar ao objetivo, nem um elemento bônus será usado no momento em que foi dado.

Até certo ponto, os métodos de tentativa e erro poderão ter algum efeito, mas em certo ponto da fase, o jogador se verá obrigado a checar as informações dos elementos e reagentes, afinal, é muito improvável encontrar uma combinação específica quando se possui, por exemplo, dez elementos na lista e três catalisadores.

5. Conclusão

A produção do Alchemilla foi um sucesso. Com uma estrututa organizada e um sistema de fácil edição, o programa fez excelente uso da linguagem C. Como visto acima, as fases do jogo independem de programação, todas as fases sendo fácilmente alteráveis. Ao mesmo tempo, temos um código claro e bastante seccionado, habilitando alterações e reparos eventuais com facilidade. Pelo outro lado, tivemos um resultado pedagógico satisfatório. O objetivo de um ensino espontâneo e leve foi atingido, acionado pela curiosidade e vontade de entender impusionado pelo lazer do jogo eletrônico. Em todas as formas, o projeto ficou como o esperado, satisfatório para todos os integrantes do grupo, e prometendo uma possivel continuação.

6. Agradecimentos

Alexandre Bencz: função fgetline.

Daniel Nopper Silva Rodrigues: auxílio na criação de conteúdo pedagógico e da música de fundo.

Rafael Toledo(http://www.rafaeltoledo.net/tutoriais-allegro-5/): conceitos básicos na utilização do Allegro 5.

Referências

Allegro 5 Development Team(2013).Allegro 5.0 reference manual, https://www.allegro.cc/manual/5/, acesso em novembro de 2013.