**UNIVERSIDADE FEDERAL DO RIO GRANDE DO RIO GRANDE DO SUL**

**TRABALHO FINAL DE ALGORITMOS E PROGRAMAÇÃO**

**MOUSE TRAP**

**GIORDANO SOUZA DE PAULA**

**Porto Alegre, novembro de 2018.**

**SUMÁRIO**

**1 INTRODUÇÃO .........................................................................................................3**

**2 O JOGO ...................................................................................................................4**

**3 FUNÇÕES DESENVOLVIDAS ................................................................................5**

**4 TRATAMENTO DE ERROS E OCORRÊNCIAS......................................................9**

**5 CONCLUSÃO ........................................................................................................10**

**REFERÊNCIAS ........................................................................................................11**

**1 INTRODUÇÃO**

O presente trabalho foi realizado com as mecânicas similares aos do clássico “Mouse Trap”, o qual apresenta um modo de jogo muito parecido com o do Pacman, além disso, os direitos do jogo foram vendidos para a plataforma Atari. Porém no jogo desenvolvido foi utilizada a linguagem de programação C, lecionada na disciplina de Algoritmos e Programação. Para o desenvolvimento, também foi utilizada a biblioteca visual para C - Allegro.

Desse modo, o trabalho possibilitou o aperfeiçoamento dos conhecimentos adquiridos ao longo do semestre na disciplina de Algoritmos e Programação, com ênfase no uso da linguagem C, ao mesmo tempo que permitiu o aprofundamento em outras áreas, como por exemplo o uso de uma biblioteca gráfica. O jogo desenvolvido, apesar de ter um modo de jogo muito próximo ao do original, apresenta um visual mais limpo e detalhado, comparável aos jogos - de mesmo gênero - mais atuais.



Figura 1 - Mouse Trap - 16 Bit Pocket Power (1991) - Atari 2600

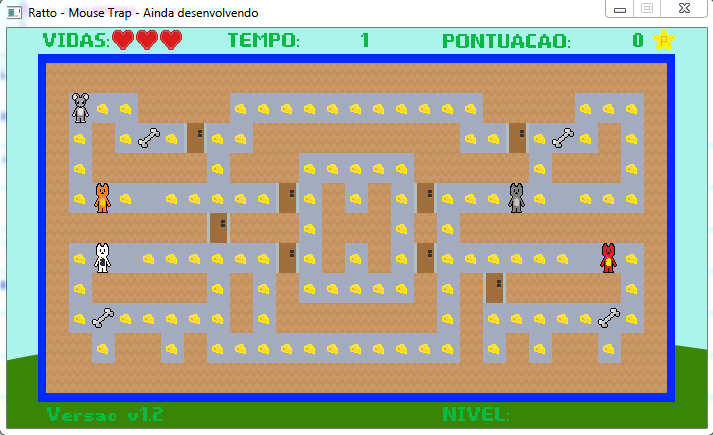


Figura 2 - Ratto - Jogo semelhante ao Mouse Trap desenvolvida neste trabalho

**2 O JOGO**

Baseado no Clássico “Mouse Trap”, o jogo desenvolvido, Ratto, apresenta uma forma de jogar muito parecida. Controlado pelo jogador através das setas do teclado ou pelas letras W, A, S e D, o rato tem inicialmente três vidas e sua missão consiste em fugir dos gatos através do labirinto e coletar todos os queijos de cada nível, passando assim para o próximo, sendo um total de três níveis distintos entre si. Na tela é mostrada a pontuação atual, em que cada queijo recolhido gera 10 pontos e cada gato derrotado gera 50 pontos.

Em cada fase, são gerados quatro gatos, os quais recebem os seguintes nomes: Askel, Charlie, Susan e Pyro, sendo Askel o primeiro “G” a aparecer na matriz lida a partir do arquivo de texto, e os demais são os subsequentes. Eles se movimentam de forma aleatória pelo labirinto uma unidade a cada segundo, e a cada vez que estes colidem com um obstáculo, passarão a se mover em uma outra direção também aleatória.

Além disso, existem ossos espalhados pelo labirinto que irão possibilitar ao rato se transformar em um cão durante um tempo de 5 segundos, conseguindo derrotar os gatos ao tocar e neles, os quais retornarão para a posição inicial. Há também portas que podem abrir ou fechar ao se pressionar a letra B, facilitando a fuga do rato. Porém, no caso de a porta ser aberta sobre o rato ou os ratos, estes perderão a vida. Outra forma para o rato perder a vida é ao tocar em um dos gatos, assim, ao perder todas as três vidas, o jogo será finalizado, não havendo possibilidade de voltar ao ponto em que se encerrou. Vale ressaltar que cada labirinto conta com exatamente 7 portas, 4 gatos e 4 ossos.

**3 FUNÇÕES DESENVOLVIDAS**

Sub-rotinas que foram desenvolvidas para um melhor funcionamento do código, evitando repetições e facilitando o acesso a determinada ordem de comandos.

* **void permite\_fechar():** Função que altera o estado da variável Fechar, permitindo o fechamento da janela principal, quando o comando for solicitado.
* **void menu\_principal():** Imprime na tela todos os principais elementos utilizados no menu principal do jogo, acessado logo ao abrir a aplicação, ou ao apertar a tecla TAB, a qualquer momento durante o jogo (dentro de algum nivel), como pode ser visto na figura abaixo:



Figura 3 - Ratto - Menu principal

* **void menu\_principal\_atualiza\_retangulo(int selecionador\_rr):** Função secundária desenha um quadrado azul nas posições em que o cursor (cabeça do rato) não está, para não duplicar o sprite. Recebe como parâmetro, uma variável do tipo int, que indica a posição do cursor (em pixels).
* **void m\_instrucoes():** Função similar ao menu principal, porém esta imprime as instruções sobre como jogar, e algumas dicas básicas sobre o jogo.

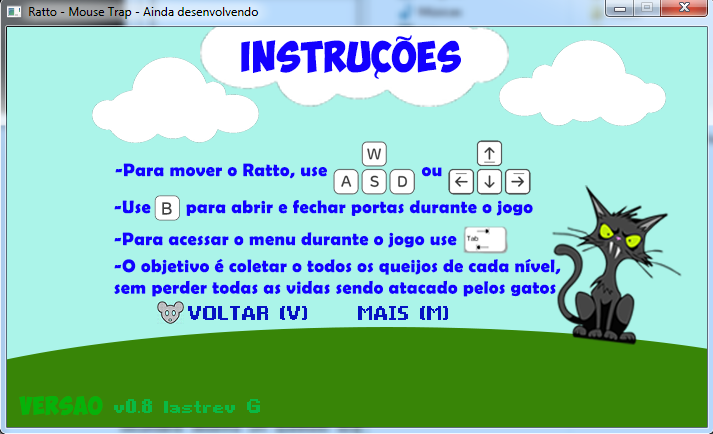
****

Figura 4 - Ratto - Menu de instruções

* **void init(int resol\_x, int resol\_y):** Inicia uma janela secundária (além do prompt de comando), um exemplo dessa pode ser visto nas figuras 2, 3 e 4, sendo originada da biblioteca gráfica Allegro, que permite a manipulação de sprites, áudio e demais ferramentas gráficas.
* **void deinit():** Função que limpa o buffer das teclas digitadas (originado da biblioteca Allegro).
* **int le\_mapa(int nivel):** Essa função acessa um arquivo de texto, localizado dentro da pasta “levels”, subpasta do jogo, e consegue atualizar uma matriz, com os valores obtidos, definindo a posição inicial dos gatos, ossos, queijos e do próprio rato. Essa também devolve estados particulares para a função principal, para cada tipo de erro encontrado. Um exemplo de arquivo de mapa (texto) utilizado é exibido na figura abaixo:

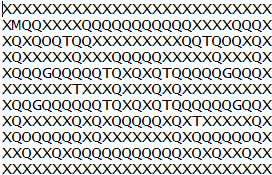


Figura 5 - Mapa em formato texto

* **int novo\_jogo(int caller):** É uma das funções principais do jogo, nela que são controlados os acessos a maioria das subrotinas utilizadas, é acessada quando o usuário seleciona “novo jogo” ou “continuar jogo” no menu principal. Ela recebe um valor inteiro por referência, este indica se uma nova partida foi solicitada, ou se o programa deve acessar um arquivo binário na subpasta “saves” e carregar uma partida anterior. Todo o controle de movimento do jogador, dos gatos, colisões, abertura de portas, início dos níveis, e fechamento do jogo é desenvolvido dentro desta.
* **void imprime\_vidas\_hud(int vidas\_iniciais):** Função que atualiza o número de vidas (corações) restantes, sendo impressos na tela, como na figura abaixo. Essa recebe como referência um valor inicial de vidas que o rato tem, pois ao perder uma vida, é impresso um coração vazio no lugar, indicando que antes ali havia uma vida.

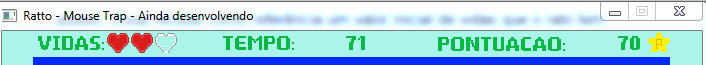


Figura 6 - Ratto - HUD

* **void imprime\_pontos\_hud(int pontos):** Semelhante a função descrita acima, porém esta converte o valor numérico de pontos para um vetor, e imprime em modo texto na interface gráfica.
* **void imprime\_mapa\_tela(BITMAP \*paredes\_mapa, int caller, bool portas\_alteradas):** Função que exibe o mapa na interface gráfica, dentre o HUD, a partir da matriz do mapa obtida pela função le\_mapa (extraída de um arquivo de texto). Recebe por referência um arquivo de sprites que contém a textura utilizada nas paredes do mapa, conforme pode ser visto na figura abaixo (a cor rosa é adotada como *true color* - transparente - pela biblioteca gráfica), essa função também considera se as portas estão abertas ou fechadas, de acordo com a variável booleana “portas\_alteradas”.

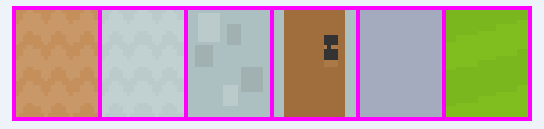


Figura 7 - Sprite de texturas das paredes do mapa

* **int calcula\_pontuacao():** Calcula os pontos que o jogador possui no nível atual, considerando os seguintes valores para cada item/ação realizado: Queijo comido (10 pontos), Gato morto (50 pontos).
* **void coleta\_queijo(int posicao\_rato\_matriz\_mapa\_x, int posicao\_rato\_matriz\_mapa\_y):** Função que retira um queijo da matriz mapa, quando este for coletado, atualizando essa ação visualmente.
* **void coleta\_osso(int posicao\_rato\_matriz\_mapa\_x, int posicao\_rato\_matriz\_mapa\_y):** Semelhante a função descrita acima, porém para ossos.
* **void reseta\_posicao\_criaturas(bool reseta\_rato, bool reseta\_Askel, bool reseta\_Charlie, bool reseta\_Susan, bool reseta\_Pyro):** Rotina secundaria desenvolvida para retornar a posição de determinada criatura à suas devidas posições iniciais, conforme for solicitado pelas variáveis booleanas recebidas por referência.
* **int gera\_numero\_aleatorio():** Rotina secundária que gera um número aleatório entre 0 e 3, esse indica a próxima posição que um gato irá seguir, assim que as condições para isso forem atendidas.
* **void imprime\_tempo\_hud():** Semelhante às funções que atualizam a pontuação e as vidas no HUD.
* **void zera\_variaveis\_nivel():** Rotina secundária que reseta as variáveis de pontuação, vidas, etc dos níveis, quando um novo jogo é solicitado.
* **bool salvarDados(VARIAVEIS memoria):** Salva os dados do jogo atual em um arquivo binário, é chamada a qualquer momento que o jogador retorna ao menu principal.
* **VARIAVEIS le\_memoria():** Rotina que acessa o arquivo binário criado na pasta “saves” e extrai o seu conteúdo, quando a rotina de continuar um jogo já existente é solicitada.

Abaixo, na figura, é exibida a forma de hierarquia dos arquivos utilizados e acessados durante a execução do jogo, cada tipo de arquivo foi organizado em uma pasta própria para o seu tipo.

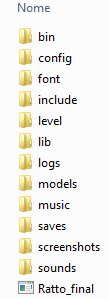


Figura 8 - Pastas utilizadas durante o jogo

**4 TRATAMENTO DE ERROS E OCORRÊNCIAS**

Durante o desenvolvimento do jogo foram adotados alguns códigos de retorno padrão para situações em que foram encontrados erros ou ocorrências marcantes, sendo esses diferentes, o processo de remoção de erros é facilitado. Esses códigos são interpretados pelo programa principal (main). Abaixo estão listados alguns desses códigos e seus significados:

**-Código 100:** Erro encontrado ao iniciar driver de som (interface gráfica).

**-Código 101:** Erro ao manipular um arquivo contendo um mapa.

**-Código 404:** Erro ao manipular o arquivo binário contendo o jogo salvo.

**-Código 15:** O rato coletou todos os queijos (passou de nível).

**-Código 20:** O jogador pressiona TAB durante o jogo (salva o jogo).

**-Código 25:** O rato encontra-se sem vidas (fim de jogo)

**5 CONCLUSÃO**

Ao longo de todo o processo de planejamento e realização do jogo, foi possível colocar em prática os conteúdos aprendidos na disciplina de Algoritmos e Programação. Dessa forma, o trabalho de elaboração do jogo proposto foi desafiador em diversos aspectos, como foi o caso de exigir o uso de habilidades extras com relação às aulas, como é o caso do uso de sprites de uma biblioteca gráfica para manipulação das imagens.

Também é importante destacar a dificuldade encontrada em proceder com a atividade de forma individual, tendo em vista a desistência por parte do segundo integrante do grupo. Devido a isso, foi necessário grande empenho na administração do tempo proposto.

Ademais, foram empregados de forma eficiente os conhecimentos de uso da linguagem C, das variáveis Booleanas e o uso de condicionais muito específicas, o qual foi aprofundado na disciplina de matemática discreta no conteúdo de circuitos lógicos, sendo assim, o trabalho permitiu também a relação entre as matérias do curso de Engenharia de Computação.

**REFERÊNCIAS**

**Jogos & Programação. Canal de vídeos no youtube:** Disponível em: <<https://www.youtube.com/channel/UC1ESKyw2NK_T0KDsw9N_UJQ>> Acesso em 21 de novembro de 2018.

**Introdução à Programação em C com Jogos 2D. Blog sobre desenvolvimento de jogos.** Disponível em: <<https://sites.google.com/a/liesenberg.biz/cjogos/home/software/ambiente-code-blocks-allegro-4>> Acesso em 21 de novembro de 2018.

**Página pública sobre informações de instalação, manipulação e desenvolvimento da biblioteca gráfica Allegro.** Disponível em: <<https://github.com/liballeg/allegro_wiki/wiki>> Acesso em 21 de novembro de 2018.

**Página principal de referência da biblioteca Allegro. Disponível em: <**[**https://liballeg.org/docs.html**](https://liballeg.org/docs.html)**>** Acesso em 21 de novembro de 2018.

**Stack Overflow. Site de perguntas e respostas sobre dúvidas comuns de programação.** Disponível em: <<https://stackoverflow.com/questions/14615148/how-to-copy-memory>

<http://www.cplusplus.com/reference/>> Acesso em 21 de novembro de 2018.

**Spriters Resource. Site de desenvolvimento de sprites para jogos.** Disponível em: <<https://www.spriters-resource.com/>>Acesso em 21 de novembro de 2018.