TRABALHO PARA A

DISCIPLINA DE FUNDAMENTOS DE PROGRAMAÇÃO 1 DO CURSO

DE ENGENHARIA ELETRÔNICA DA UTFPR

Guilherme Belchior Ogibowski

Disciplina: Fundamentos de Programação 1/S11 – Prof.Rebelatto

Departamento Acadêmico de Eletrônica – DAELN – Campus de Curitiba

Universidade Tecnológica Federal do Paraná – UTFPR

guilhermeguiba@hotmail.com

**Resumo** – O trabalho foi elaborado para a maior compreensão e utilização da biblioteca Allegro 4 , no qual foi desenvolvido um simulador de voo , onde uma nave deve desviar dos obstáculos , eliminar os inimigos e não deixar que seu combustível se esgote.

**Palavras-chave ou Expressões-chave:** Jogo em Allegro 4, Trabalho Acadêmico em Linguagem C ,Simulador de voo em Allegro 4.

**INTRODUÇÃO**

Visando uma maior compreensão de linguagem C juntamente com a biblioteca gráfica Allegro 4 foi desenvolvido um jogo baseado no famoso Raid River. O jogo consiste em uma nave que precisa desviar dos obstáculos , eliminar os inimigos e evitar que seu combustível se esgote, lembrando que se perde uma vida ao colidir com qualquer coisa exceto com o combustível.

**EXPLICAÇÃO DO JOGO EM SI**

Para melhor organização foram acrescentados ao jogo cabeçalhos em c e em h para os personagens com estruturas dentro com variáveis representando posição, velocidade, tamanho no eixo x , tamanho no eixo y ,um inteiro “ativo” para possíveis verificações e uma bitmap. No mesmo cabeçalho h foi criado uma estrutura para a nave espacial, contendo as mesmas variáveis anteriores acrescidas de uma variável para vidas , pontos e combustível. No cabeçalho c das respectivas estruturas foram criados funções que receberam como parâmetro ponteiros de cada estrutura e variáveis que serão transmitidas à função quando a mesma for chamada na função principal. A mesma divisão foi feita para o teclado, onde no cabeçalho h foram apresentados as funções apertou, segurou, soltou e uma para atualização do estado do teclado. No cabeçalho c do teclado foram definidas as funções mencionadas anteriormente para a certificação que a tecla de “esquerda” e “direita” funcione apenas se for apertada, para o movimento da nave.

No cabeçalho c principal do jogo ,primeiramente definiu-se a função que detecta colisões (pixel perfect) , na qual recebe como parâmetros as posições dos objetos que se colidirão e as respectivas imagens. Na mesma função tem a colisão por retângulos, onde é checado se a soma da altura ou largura com a posição do objeto é maior que a posição do outro objeto e então se determina se está ou não colidindo. Ainda antes da função principal são mencionadas estruturas derivadas dos presentes no cabeçalho c e h de “objetos” já citados, para o tiro, nave, ponte, e raio, gasolina e navio, estas ultimas como vetor para futuramente gerar posições randômicas.

Dentro da função principal são instalados alguns requisitos padrões do Allegro , como: timer, keyboard e outros. É criada uma variável milissegundos para futura contagem de tempo (combustível). Logo em seguida são definidos os “bitmaps” a serem utilizadas no jogo. São introduzidas também algumas variáveis como posição x e y da nave ,e algumas auxiliares. Logo abaixo são passadas como referência o endereço dos personagens, suas posições iniciais e as respectivas “bitmaps” para a estrutura personagem e nave espacial, citadas no começo do texto, no qual duas usam a ferramenta “rand ” para gerar posições aleatórias.

Dentro do ”looping” do jogo foi criada uma condição de fechamento de tela ao pressionar a tecla “esc”. Em seguida foi criada uma série de condições para o movimento da nave, na qual se for clicada a tecla cima, baixo, esquerda ou direita com ajuda de uma variável auxiliar e também da função teclado , mencionada no começo, impede que a nave se mova se a tecla direita ou esquerda estiver pressionada , e se a tecla for apenas clicada a posição da nave é acrescida ou diminuída. Foi criada também uma condição para que o tiro da nave seja arremessado caso seja apertada a tecla “espaço”, juntamente com uma verificação de ativo e não ativo se ele ultrapassar a tela para que seja lançado um tiro por vez.

Para a parte de aceleração da nave foi criada condições se a tecla “cima” ou “baixo” for apertada, na qual retardará seu movimento ou acelerará juntamente com o fundo. Logo em seguida foram estabelecidas as posições aleatórias do raio , navio e combustível, por meio de vetores e da ferramenta “rand”, já mencionada, juntamente com sua velocidade e condição de ativo e não ativo. Também se determinou a posição da ponte.

Na parte de atualização é decrescido o combustível (por meio do tempo) e deslocamento do fundo. Em seguida são chamadas as funções de colisão, citadas no começo do texto, para todos os objetos com a nave, tiro e fundo. Na parte de desenhos foram desenhados no “buffer” todos os personagens e tiro de acordo com sua classificação de ativo. Foram impressas na tela a situação de pontos, vidas e combustível que vão variar em tempo real no jogo. Por fim foram destruídos todos os “bitmaps” para liberação de espaço.

**CONCEITOS UTILIZADOS E NÃO UTILIZADOS**

**- Estruturas de Decisão**: sim

Linhas: 21,46,200,215,240,254,265,286,307,326,340,426.

**- Estruturas de Repetição**: sim

Linhas: 53,178,192,200,284,357,379,390,401,424,433,440.

**- Estruturas de Dados Homogêneas – Vetores**: sim

Linhas: 187, 193, 265, 273, 288, 309, 426,436.

- **Estruturas de Dados Homogêneos – Matrizes:** não

**- Funções:** sim

Linhas: 20, 28, 40, 166, 169, 173.

- **Funções com parâmetro por valor**: sim

Linhas: 20, 28, 40

- **Funções com parâmetro por referência**: sim

Linhas: 166, 169, 173, 180, 193.

- **Struct:** sim

Linhas: 7,20 (objetos.h)

- **Enum**: não

- **Ponteiros**: sim

Linhas: 5 e 4,19 (nave.c)

- **Alocação dinâmica de memória**: não

- **Arquivos de Texto**: não

**Justificativas:**

- Uso de estruturas de decisão: não vejo outra maneira de montar um jogo sem pelo menos usar uma estrutura de decisão, ele é quase que indispensável.

- Estruturas de Repetição: É fundamental para rodar o looping do jogo e percorrer vetores.

- Estruturas de Dados Homogêneas - Vetores: Para a utilização do rand e para determinar a quantidade de determinadas coisas, como o navio e gasolina.

- Estruturas de Dados Homogêneos – Matrizes.: Não tive a oportunidade de acrescentar no meu projeto, não surgiu ocasião.

- Funções: Maior facilidade para passar parâmetros e não deixar o programa “poluído”.

- Funções com parâmetro por valor. Para funções simples dentro do int main que não exigiam a mudança do valor.

-Funções com parâmetro por referência: Para passar valores de uma estruct inteira para futuramente mudar seus valores.

- Struct: Maior organização e a não necessidade de criar varias variáveis iguais para objetos diferentes.

- Enum: Foram criadas variáveis ao invés de utilizar enum.

- Ponteiros: ao passar struct por referência necessitou-se de ponteiros para apontar para determinados endereços;

- Alocação dinâmica de memória: não tive a oportunidade e adicionar ao meu projeto, talvez por ele estar incompleto.

- Arquivos de Texto: não tive a oportunidade e adicionar ao meu projeto, talvez por ele estar incompleto.

**DISCUSSÃO E CONCLUSÕES**

É possível notar que a biblioteca Allegro possui muitas funcionalidades que são fundamentais para a produção de um jogo. E apesar de o projeto não estar completo, com muito esforço, pôde ser aprendido como usar muitas das funções dessa biblioteca, que mesmo sendo parecida com as do DEV C++ possui muitas divergências, que só com muitas pesquisas foi possível de incrementar, como os tiros e as colisões. No que diz respeito a produção do jogo, não há a necessidade de tabela de divisão de tarefas, uma vez que esse projeto apresenta apenas um integrante.

**CONSIDERAÇÕES PESSOAIS**

Encontrei muitas dificuldades ao decorrer da produção do jogo, uma vez que a biblioteca utilizada possui muitas funcionalidades diferentes das aprendidas em sala.

**REFERÊNCIAS**

[1] URBANO, A. Tutoriais Allegro 4. Disponível em: <https://www.youtube.com/playlist?list=PLVg7a6fFGaj6DyuqygwqHqGhCu2w7Iub>.

[2] HOLMES, J. Programar em Allegro. Disponível em: <http://www.portugal-a-programar.pt/blogs/>. Acesso em: 24.jun.2015.

[3] MOTA, T. Técnicas de Programação. Disponível em: <http://equipe.nce.ufrj.br/adriano/c/apostila/allegro/docs/tecnicas.html#scrolling>. Acesso em: 27.jun.2015.

[4] FURTADO, D. Allegro 4. Disponível em: <http://www.hardware.com.br/comunidade/>. Acesso em: 29.jun.2015.