Trabalho Prático - Pong

Data de Entrega: Conferir no calendário!

Pedro O.S. Vaz de Melo

March 11, 2014

1 DESCRIÇÃO DO PROBLEMA

O objetivo deste trabalho é fazer com que o aluno utilize as técnicas de programação aprendidas na disciplina para desenvolver um dos mais clássicos dos videogames: o Pong. No jogo, dois jogadores se movimentam verticalmente a fim de rebater a bola quando essa chega até eles. Se um jogador não conseguir rebater a bola, ele é declarado perdedor e o outro vencedor. Neste TP, um dos jogadores deve ser controlado pelo usuário e o outro controlado pelo computador. A Figura 1.1 ilustra um exemplo de como o jogo pode ser criado.
Este trabalho tem um valor total de 20 pontos e será avaliado de acordo com os seguintes critérios:

- **Documentação** curta contendo as seguintes seções: **(1) Manual de Uso**, que descreve como operar o jogo; **(2) Implementação**, que descreve brevemente os trechos de código desenvolvidos por você[1].
- Conhecimento do aluno sobre o código apresentado. Isso será verificado via *prova oral* em laboratório.
- Solução apresentada, isto é, se o aluno foi capaz de desenvolver o jogo proposto.
- Clareza do código.

[1] Exemplo: Da linha 25 à linha 35 há um controle para verificar qual tecla foi digitada pelo usuário. As teclas S e W controlam o movimento do jogador 1.

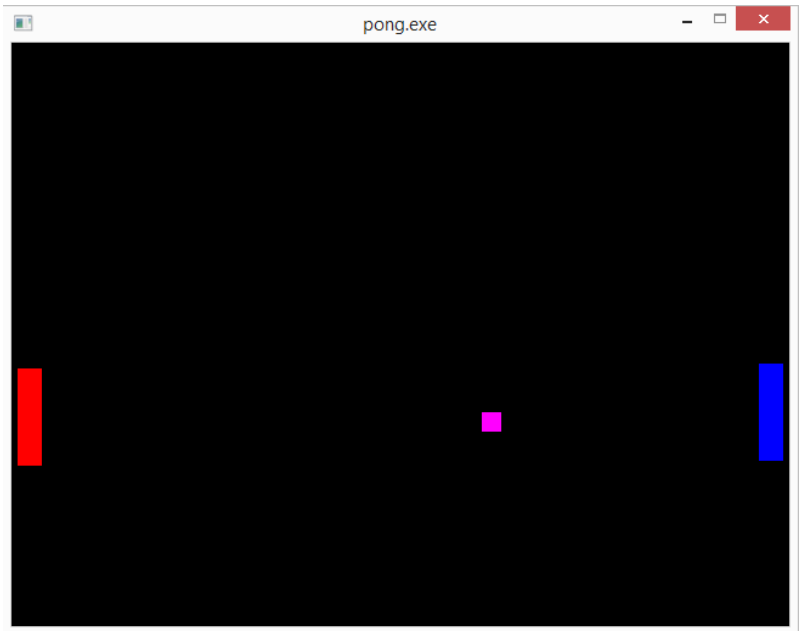

Figure 1.1: Esquema básico do jogo proposto. Cada jogador fica em um canto oposto da tela e pode se movimentar verticalmente para rebater a bola. Neste caso, o jogador à esquerda é controlado pelo usuário e o jogador à direita é controlado pelo computador.

2 O JOGO (12 PONTOS)

O objetivo deste estágio do trabalho é desenvolver um protótipo do jogo *Pong* como ilustrado na Figura 1.1. Para obter os 12 pontos, o aluno só precisa de desenvolver o seguinte:

- Permitir que o usuário controle um dos jogadores, o movimentado verticamente em um dos cantos laterais da tela.
- Fazer com que o outro jogador seja controlado pelo computador, sendo capaz de rebater, pelo menos, a maioria das bolas que chega até ele.
- Ambos os jogadores devem estar afastados claramente das paredes laterais.
- A rebatida da bola com os jogadores deve ser dada de forma clara e sem sobreposição de pixels, ou seja, a bola e o jogador não podem ocupar o mesmo ponto do espaço em nenhum momento.
- A bola deve mudar de comportamente toda vez que for rebatida (ex: ângulo, velocidade, tamanho etc).
- O jogo deve informar no final, pelo menos, quem ganhou e o número de rebatidas totais.

Além dos 20 pontos, o professor pode atribuir **até** 10 pontos extras para o aluno caso esse apresente elementos adicionais ao jogo, tais como:

- Fazer com que o jogo tenha diferentes estágios, com diferentes níveis de dificuldade;
- Possibilitar que os jogadores capturem *power-ups* na tela, que podem dar poderes aos jogadores (ex: aumentar de tamanho), prejudicar o adversário (ex: diminuir de tamanho) ou simplesmente melhorar a pontuação do usuário, caso exista;
- Possibilitar que o usuário escolha diferentes tipos de jogadores, cada qual com características particulares;
- Colocar sons e músicas que tornem o jogo mais divertido;
- Colocar animações que tornem o jogo mais divertido;
- Colocar obstáculos extras na tela;
- Permitir que um segundo usuário também jogue;
- Permitir que o usuário controle a força e direção da bola a partir do teclado;
- Colocar uma rede no centro da tela e simular um jogo de tênis, com a bola ganhando e perdendo altura;
- Etc, etc, etc...

Importante: Geralmente é dado 0.5 ponto extra por item implementado, mas isso pode variar de acordo com o critério do professor. Há itens (o último, por exemplo), que definitivamente valem mais que 0.5 pontos. Itens não descritos na lista acima também poderão ser considerados.

3 GERENCIAMENTO DOS USUÁRIOS (8 PONTOS)

Depois que o jogo for desenvolvido, o aluno deve implementar um módulo de cadastro e login de usuários. Para obter os 8 pontos desta fase do trabalho, o seguintes pontos devem ser implementados:

- Antes do jogo ser iniciado, o seu programa deve pedir o login e a senha de um jogador ou então o cadastro de um novo.
- Para o cadastro, o programa deve pedir do jogador o seu login (ex: polmo), senha, nome e idade. Esses dados devem ser salvos em arquivos pelo programa. Pontos extras serão dados a alunos que conseguirem armazenar e carregar a senha de forma criptografada.
- Para o login, o programa deve abrir o arquivo que contém a informação do jogador e comparar a senha que ele digitou com a senha que está no arquivo. Pontos extras serão dados para os alunos que conseguirem esconder a senha quando digitada na tela.
- Depois que o jogador estiver logado, o programa pode iniciar o jogo ou mostrar a melhor pontuação (ex: menor tempo) que esse jogador obteve no seu histórico de partidas.
- (Ponto extra). Criar uma opção para exibir o ranking dos jogadores com as maiores pontuações.

Para implementar a estrutura que gerencia os jogadores, o aluno deve também definir um novo tipo de dados a partir de uma estrutura, como a seguir:

```
typedef struct Jogador {
   char login[8];
   char senha[8];
   char nome[100];
   int idade;
   float maiorPontuacao;
} Jogador;
```

Os dados dos jogadores podem ser armazenados em um único arquivo ou em vários. Isso deve ficar a cargo do aluno. Além disso, para carregar e salvar as informações sobre um jogador, o aluno **deve criar um módulo específico para isso**, de nome `jogadores.h`. Esse módulo deve conter todas as funções de controle dos dados dos jogadores, como as seguintes:

```
Jogador carregaJogador(char login_jogador[], char senha_jogador[]);
void salvaJogador(Jogador jogador);
void imprimeInformacoesJogador(char login_jogador[]);
void imprimeRankingJogadores(); //ponto extra
```

Assim, o controle do jogo pode ser feito no seu arquivo .c principal (ex: frogger.c), que contém a função `main()` e que usa o módulo `jogadores.h` via `#include "jogadores.h"`. No entanto, o aluno é livre para organizar o seu programa da maneira que desejar, **desde que tenha um módulo específico para tratar o cadastro dos jogadores**.

4 MELHORES TRABALHOS

Os alunos desenvolvedores dos três melhores trabalhos, além dos pontos extras recebidos, também poderão ser premiados pelo professor (prêmio a definir).

5 COMO EU FAÇO?

5.1 INSTALANDO O AMBIENTE DE PROGRAMAÇÃO

Apesar da descrição fazer o trabalho parecer complicado, ele é bastante simples. Tudo que o aluno precisa saber para desenvolver este jogo são os conhecimentos adquiridos na disciplina e um pequeno entendimento de desenvolvimento de aplicações gráficas. Assim como são necessárias bibliotecas novas para a utilização de funções não nativas da linguagem C, como a `Math.h`, uma biblioteca também é necessária para que se utilize funções gráficas. Para este trabalho, pede-se que se utilize a biblioteca Allegro5, que fornece inúmeras funções que podem ajudar no desenvolvimento deste trabalho.
Para iniciar o desenvolvimento do trabalho, pode-se configurar um ambiente de desenvolvimento no Code::Blocks ou via linha de comando, usando o `gcc` e um editor de texto. Para usar o Code:Blocks, basta seguir os passos do tutorial apontado pela URL:

`https://sites.google.com/a/liesenberg.biz/cjogos/home/software/ambiente-code-blocks-allegro-5`

Caso você use Windows e prefira usar `gcc` (**recomendável**)e um editor de texto para desenvolver o seu trabalho, siga os seguintes passos:

1. Para instalar o `gcc`, faça o download da última versão do MingW em:

   `http://sourceforge.net/projects/mingw/files/`

   Instale normalmente, marcando todos os itens da instalação básica.

2. Edite as variáveis de sistema do seu computador e coloque o diretório dos arquivos binários do MingW na variável de ambiente `PATH`. Para isso, abra o Painel de Controle, Sistema e Segurança, Sistema e clique em *Opções Avançadas de Sistema*. Na aba *Avançadas*, clique no botão *Variáveis de Ambiente*. Na lista de variáveis do sistema, procure a variável *Path* e clique em *Editar*. No final do campo *valor da variável*, depois do último ponto e vírgula, coloque o diretório que estão os arquivos binários do MingW (se instalou normalmente, estão em $C:\MinGW\bin$).

3. Faça o download dos arquivos binários da biblioteca Allegro5 para a versão do MingW que você instalou (normalmente é a 4.6.2 e/ou a 4.6.2 funciona) na URL:

`https://www.allegro.cc/files/`

Se a sua versão do MingW é a 4.6.2, você pode baixar o arquivo zip nesta página através do link:

```
http://cdn.allegro.cc/file/library/allegro/5.0.10/allegro-5.0.10-mingw-4.6.2.zip
```

Salve o zip no diretório raiz ($C:\backslash$) do seu computador (pode ser qualquer outro) e extraia o seu conteúdo. Uma pasta de nome `allegro-5.0.10-mingw-4.6.2` deve ter sido criada.

4. Dentro do diretório `bin` do diretório `allegro-5.0.10-mingw-4.6.2` há um arquivo `allegro-5.0.10-monolith-mt.dll`. Copie esse arquivo e o cole no diretório que você vai usar para desenvolver o seu trabalho. Para rodar qualquer programa que use a biblioteca Allegro5, você precisa que esse arquivo esteja no mesmo diretório do programa que desenvolveu.

5. Para compilar qualquer programa Allegro5 a partir da linha de comando, você deve indicar manualmente a localização dos arquivos `.h` da biblioteca Allegro5. Assim, para compilar um programa que está no arquivo `jogo.c`, você deve executar o comando (cuidado para não colocar uma \ depois do `include`):

```
gcc -I C:\allegro-5.0.10-mingw-4.6.2\include -c jogo.c
```

6. Para criar o seu programa executável, é necessário fazer a ligação das bibliotecas allegro5 com o seu programa objeto. Então, após o passo de compilação, é necessário executar o seguinte comando:

```
gcc -o jogo.exe jogo.o C:\allegro-5.0.10-mingw-4.6.2\lib\liballegro-5.0.10-monolith-mt.a
```

7. Se todos os passos foram seguidos corretamente, você pode executar o seu programa `jogo.exe`.

5.2 Alguns Exemplos

Junto com este documento foram disponibilizados quatro exemplos de programas em Allegro:

1. `louco.c`;
2. `teclado.c`;
3. `passaroandante.c`;
4. `bouncer.c`;
5. `frogger.c`;
6. `passaroraivoso.c`;

O conteúdo desses programas é suficiente para desenvolver a versão básica do jogo proposto neste documento. Veja cuidadosamente o código de cada um desses programas (na ordem, pois vão do mais simples ao mais complexo) e os compile (usando os comandos da seção anterior) e os modifique à vontade. Note que há comentários explicativos em praticamente todas as linhas de código desses programas. Apesar disso, sempre que necessário procure pelas referências das funções Allegro no seu manual:

`https://www.allegro.cc/manual/5/`

6 CRIANDO UM MAKEFILE

Para não ficar digitando os longos comandos de compilação descritos neste documento, você pode criar um arquivo que os executa automaticamente para você. Para isso, você deve criar um arquivo "Makefile" (sem extensão) semelhante (ou igual) ao que acompanha este documento. Para executar um arquivo "Makefile", basta digitar o comando "make" na linha de comando. Este comando procura automaticamente por um arquivo "Makefile" e tenta criar o arquivo marcado pela *tag* "all". Se você instalou corretamente o **gcc** a partir deste documento, basta renomear o arquivo `mingw32-make` localizado na pasta `C:\MinGW\bin` para simplesmente `make`.
Vamos olhar para um arquivo "Makefile" em detalhes:

```
all: pong.exe

pong.exe: pong.o
    gcc -o pong.exe pong.o C:\allegro-5.0.10-mingw-4.6.2\lib\liballegro-5.0.10-monolith-mt.a

pong.o: pong.c
    gcc -I C:\allegro-5.0.10-mingw-4.6.2\include -c pong.c

clean:
    del pong.o
    del pong.exe
```

A primeira linha do programa mostra o arquivo que queremos criar, que neste caso, é o arquivo `pong.exe`, o executável do jogo. Sempre que executamos o programa "make" sem parâmetros, ele verifica se existe o(s) arquivo(s) marcados pela tag "all". Caso o arquivo não exista, ele vai executar os comandos necessários para criá-lo. Na linha de baixo, temos os arquivos e comandos necessários para criar o arquivo `pong.exe`. Para a sua criação, é necessário o arquivo `pong.o`. Se esses arquivos existem, o comando

```
gcc -o pong.exe pong.o C:\allegro-5.0.10-mingw-4.6.2\lib\liballegro-5.0.10-monolith-mt.a
```

será executado para criar o arquivo `pong.exe`. Caso o arquivo `pong.o` não exista, o programa irá procurar pela *tag* "pong.o" no arquivo para que o mesmo seja criado. Neste caso, o arquivo `pong.o` só depende do arquivo `pong.c`. Se o arquivo `pong.c` existe, então o comando

```
gcc -I C:\allegro-5.0.10-mingw-4.6.2\include -c pong.c
```

será executado. Para limpar todos os arquivos que o programa **make** cria, basta executar `make clean`, como especificado pela *tag* "clean". **Importante:** você precisa colocar o *TAB* antes de cada comando que deseja executar.