

INF01202 – ALGORITMOS E PROGRAMAÇÃO

Turmas G e H

Trabalho Prático Final – Semestre 2020/1

INTRODUÇÃO

No decorrer da disciplina de Algoritmos e Programação, estão sendo apresentados diversos conceitos e técnicas de programação, visando expor aos alunos as principais ferramentas lógicas e práticas para construção de algoritmos e solução de problemas diversos. Como passo fundamental, este trabalho vem colocar à prova o aproveitamento dos alunos frente ao que foi exposto nas aulas teóricas e práticas, na forma de um jogo que deverá ser implementado na linguagem de programação C.



O objetivo é implementar uma versão simplificada do clássico jogo conhecido como *Bow and Arrow* em modo texto (*ASCII art*). O jogo é composto por um arqueiro que tem um objetivo diferente a cada fase do jogo (ver <http://www.youtube.com/watch?v=0sue0jkOKTo> para um exemplo do jogo), e nossa versão de *Bow and Arrow* possuirá duas fases: na primeira, o arqueiro deve atirar para estourar todos os balões sem gastar todas suas flechas; na segunda o arqueiro deve atirar para se defender dos monstros que vem em sua direção.

REQUISITOS

A seguir são listados os requisitos para implementação do arqueiro, das fases do jogo e do cenário.


Arqueiro

Nº	Descrição
1	O arqueiro deve estar posicionado à esquerda do cenário.
2	O arqueiro deve mover-se apenas para cima e para baixo, de acordo com os comandos do jogador dados via teclado pelas teclas W e S.
3	O arqueiro deve efetuar disparos quando o jogador teclar a barra de espaço do teclado.


4	O arqueiro deve iniciar com 15 (quinze) flechas na primeira fase e 30 (trinta) flechas na segunda fase.
5	O arqueiro deve ter tamanho de 4 linhas e 8 colunas, e sua forma será dada através de um arquivo texto <code>arqueiro.txt</code> . Um possível exemplo é 
6	As flechas devem ocupar 3 colunas na mesma linha, e sua forma será dada através de um arquivo texto <code>flecha.txt</code> . Um possível exemplo é: 
7	O arqueiro deve sempre estar sempre com o corpo todo visível na tela.
8	Cada flecha disparada deve ser descontada do contador de flechas do arqueiro.
9	Ao finalizar uma fase, cada flecha não utilizada deve dar ao jogador 50 pontos.

Fase 1: Balões

Nº	Descrição
10	A fase deve iniciar com um grupo de 15 (quinze) balões.
11	A fase deve terminar quando todos os balões forem estourados.
12	O jogo deve ser encerrado se todas as flechas forem utilizadas e ainda houver balões não estourados.
13	Os balões devem estar localizados à frente do arqueiro e alinhados horizontalmente (veja a Figura 2 para um exemplo).
14	Os balões devem mover-se constantemente e autonomamente de baixo para cima até a borda superior da tela onde desaparecem e voltam a reaparecer na borda inferior da tela com a mesma configuração. (Ex.: se apenas 10 balões ainda não foram acertados, apenas esses 10 devem reaparecer embaixo novamente).
15	O balão deve desaparecer quando for atingido pela ponta de uma flecha.
16	Cada balão atingido deve dar 100 (cem) pontos ao jogador.
17	Cada flecha disparada deve viajar em linha reta até a borda direita da tela, atingindo todos os balões que estiverem no seu caminho (ou seja, ela atravessa toda região da tela).
18	Os balões devem ter tamanho de 3 linhas e 3 colunas, e sua forma será dada através de

	um arquivo texto <code>balao.txt</code> . Um possível exemplo é 
--	---

Fase 2: Monstros

Nº	Descrição
19	A fase 2 deve ter 30 (trinta) monstros que surgem um por um a cada 2 (dois) segundos na borda direita da tela.
20	Os monstros movem-se da borda direita até a borda esquerda da tela em linha reta, horizontalmente, surgindo em linhas aleatórias.
21	A fase deve terminar quando todos os monstros atravessarem a tela.
22	O jogo deve ser encerrado se o arqueiro entrar em contato com um monstro.
23	O monstro deve desaparecer quando for atingido pela ponta de uma flecha.
24	Cada monstro atingido deve dar 200 (duzentos) pontos ao jogador.
25	Cada flecha disparada deve viajar em linha reta até a borda direita da tela ou até atingir um monstro (no segundo caso, a flecha tem sua trajetória interrompida).
26	Os monstros devem ter tamanho de 5 linhas e 6 colunas, e sua forma será dada através de um arquivo texto <code>monstro.txt</code> . Um possível exemplo é 

Cenário

Nº	Descrição
27	O cenário deve apresentar a pontuação atual do jogador.
28	O cenário deve apresentar a maior pontuação obtida até o momento no jogo.
29	O cenário deve apresentar o contador de flechas do arqueiro.
30	O cenário total do jogo deve ser representado por uma matriz de <i>char</i> , com dimensão 35 linhas x 80 colunas, sendo que as quatro primeiras linhas são reservadas para o cabeçalho. Esta matriz deve conter todos os componentes do jogo e deve ser atualizada constantemente. Exemplos podem ser vistos nas Figuras 2 e 3.

Gerais

Nº	Descrição
31	O jogo deve iniciar apresentando uma tela de menu com as opções: <ul style="list-style-type: none">• Novo Jogo• Maiores Placares• Sair
32	A fase 2 deve ser iniciada assim que o jogador finalizar a fase 1. Ao finalizar a fase 2, o jogo encerra, e o menu inicial deve ser mostrado novamente.
33	Item bônus (opcional): ao finalizar a fase 2, o jogo volta para a fase 1, mas com velocidade maior (ao invés de encerrar). Nesse caso, o jogo passa a ter 5 (cinco) velocidades, tendo assim um total de 10 (dez) níveis. Os níveis 1, 3, 5, 7 e 9 correspondem à fase 1 e os níveis 2, 4, 6, 8 e 10 correspondem a fase 2. Quanto maior o nível, maior a velocidade de movimento dos balões e monstros. Você também pode adicionar outros níveis inspirados no jogo original (como balões que sobem de maneira não alinhada e com velocidades distintas).
34	Quando o jogo iniciar, deve ser carregado um arquivo binário <i>highscores.bin</i> , contendo as 5 (cinco) maiores pontuações já registradas no jogo, juntamente com os nomes dos respectivos jogadores, e ordenadas em ordem decrescente com relação à pontuação. O arquivo binário é formado por um arranjo de 5 (cinco) elementos do tipo estruturado <i>TIPO_JOGADOR</i> , definido por: <pre>typedef struct tipo_jogador { char nome[40]; int score; } TIPO_JOGADOR;</pre> No início do jogo a forma do arqueiro, balões, flecha e monstros devem ser carregados a partir dos arquivos texto correspondentes, conforme descrito anteriormente.
35	Quando o jogo for encerrado conforme os requisitos 12, 22 ou 32, o jogo deve verificar se a pontuação do jogador é uma das cinco melhores. Caso afirmativo, perguntar o nome do jogador e atualizar o arquivo <i>highscores.bin</i> .
36	Ao selecionar a opção “Maiores placares” no menu, o jogo deve exibir as pontuações salvas no arquivo <i>highscores.bin</i> .
37	Ao selecionar a opção “Sair” no menu, o jogo deve ser fechado.

Abaixo são apresentados exemplos da tela de menu e de telas do jogo.



Figura 1. Tela de menu.

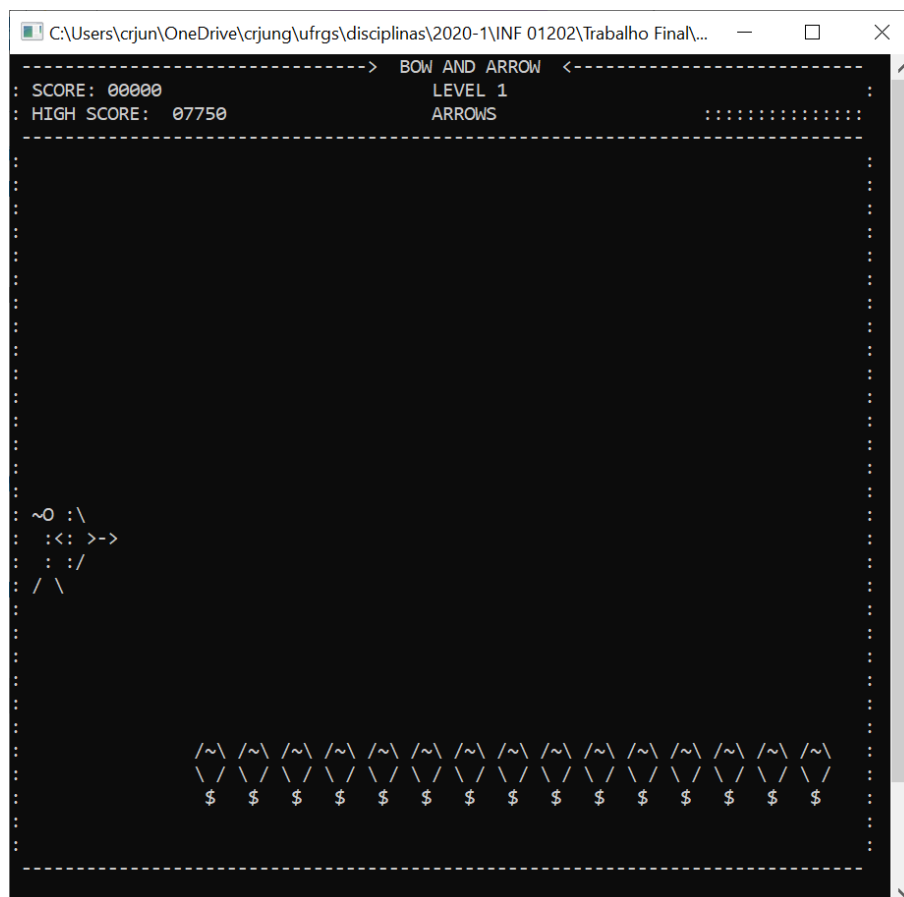


Figura 2. Exemplo de tela da fase 1.



Figura 3. Exemplo de tela da fase 2.

DICAS

Para escrever o jogo na tela é preciso controlar o cursor e definir onde devem ser escritos os dados da matriz. Pode-se utilizar a função `gotoxy()` para definir a posição do cursor na tela.

Para Windows:

```
#include <conio.h>
#include <Windows.h>

void gotoxy (int x, int y)
{
    COORD coord = {0, 0};
    coord.X = x; coord.Y = y; // X and Y coordinates
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
}
```

Para Linux:

```
#include<stdio.h>

// posiciona cursor na posicao i j
void gotoxy(int x,int y)
{
    printf("%c[%d;%df",0x1B,y,x);
}
```

O jogo funcionará em um laço principal e periodicamente deverá ler os comandos do jogador via teclado. Uma maneira de ler o teclado para analisar se o jogador pressionou alguma tecla é utilizar a função `kbhit()`. Para ler a tecla pressionada sem mostrá-la na tela, pode-se utilizar a função `getch()`.

```
#include <conio.h>

int main() {
    char c;

    if( kbhit() ) c = getch(); // lê a entrada do teclado somente quando há algo
    para ser lido

    return 0;
}
```

OUTRAS INFORMAÇÕES

- O trabalho deverá ser realizado em **duplas**. Informar os componentes da dupla até o dia **15 de outubro** aos professores da disciplina, por e-mail.
- Até o dia **18 de novembro à meia-noite**, a dupla deverá **submeter via Moodle** um arquivo zip cujo nome deve conter os nomes dos alunos. O arquivo zip deve conter:
 - Uma **descrição do trabalho** realizado contendo a **especificação completa** das estruturas utilizadas e uma explicação de como usar o programa;
 - Os **códigos-fonte** devidamente organizados e documentados (arquivos .c);
 - O **executável** do programa (**indique explicitamente** se foi feito em Windows ou Linux).
- O **trabalho será obrigatoriamente apresentado** durante a aula prática do dia **20 de novembro**, de forma remota, via Microsoft Teams. Ambos os membros da dupla deverão saber responder perguntas sobre qualquer trecho do código;

- No dia da apresentação será fornecido um novo arquivo `highscores.bin` para testar o programa, assim como os arquivos texto que representam o arqueiro, flecha, balão e monstro (os arquivos fornecidos terão tamanhos compatíveis com o elemento respectivo).
- Os seguintes itens serão considerados na avaliação do trabalho:
 - Estruturação do código em módulos;
 - Documentação geral do código (comentários, indentação);
 - Jogabilidade;
 - Atendimento aos requisitos definidos.
- Se apenas os itens relacionados à fase 1 forem implementados, a nota máxima possível para trabalho é 8. Se as duas fases forem implementadas, a nota máxima possível é 10.
- **Importante:** trabalhos copiados não serão considerados. Saibam que há ferramentas que possibilitam a detecção automática de plágio.