

INF 01202 – ALGORITMOS E PROGRAMAÇÃO

Turmas G e H

Trabalho Prático Final – Semestre 2015/1

1 MOTIVAÇÃO E OBJETIVOS

No decorrer da disciplina de algoritmos e programação são apresentados diversos conceitos e técnicas de programação, visando expor os alunos às principais ferramentas lógicas e práticas para construção de algoritmos e solução de problemas diversos. Como passo fundamental, este trabalho vem colocar a 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, **em duplas de alunos**.

O objetivo é implementar uma versão simplificada do jogo Frogger em modo texto (ASCII art). A implementação deve conter toda a interação do usuário (jogador) com o cenário.

2 ESPECIFICAÇÃO

2.1 OBJETIVO

O objetivo do jogo é fazer com que os sapos cheguem às suas “casas”. Para isso, eles devem atravessar uma estrada movimentada, sem serem atingidos pelos veículos que trafegam, e em seguida cruzar um rio, subindo nos troncos e tartarugas que passam. A pontuação é dada pelo tempo: quanto mais rápido o jogador conseguir efetuar a travessia, maior é sua pontuação.

Toda vez que o sapo for atingido por um veículo, ou se afogar no rio, ele perde uma vida. No total, um jogador possui 3 vidas, e o jogo termina quando elas se esgotarem. A Figura 1 mostra o layout de uma versão do jogo encontrada na Internet.

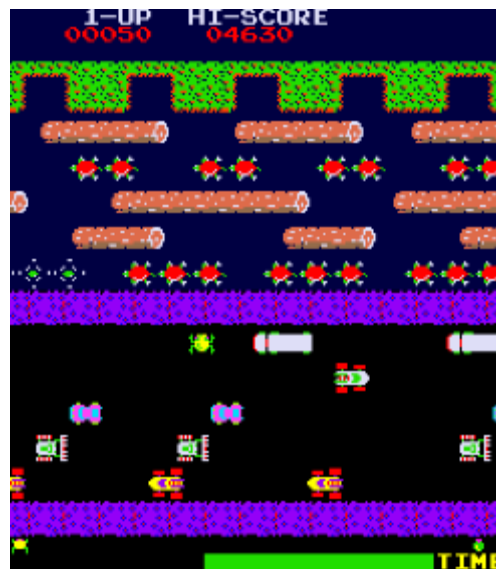


Figura 1: Jogo *Frogger* encontrado em <http://www.happyhopper.org/>

2.2 LAYOUT

O mapa é composto de objetos fixos e móveis. Os objetos fixos que compõem a paisagem do mapa são:

- Posição inicial de onde saem os sapos;
- Cinco pistas por onde passam os carros;
- Espaço livre entre as pistas e o rio;
- Cinco espaços de rio por onde passam troncos e tartarugas;
- Cinco “casas de sapo” lado a lado.

Além disso, nas duas primeiras linhas superiores, há a exibição do número de vidas extras e pontuação (atual e máxima). E na última linha é mostrado o tempo restante ao jogador. Sugere-se que o aluno observe e teste o funcionamento do jogo através do site.

Neste trabalho é pedido que o aluno desenvolva uma versão ligeiramente mais simplificada deste jogo. Onde o mapa e seus objetos são dados por caracteres ASCII, como mostra o exemplo na Figura 2.

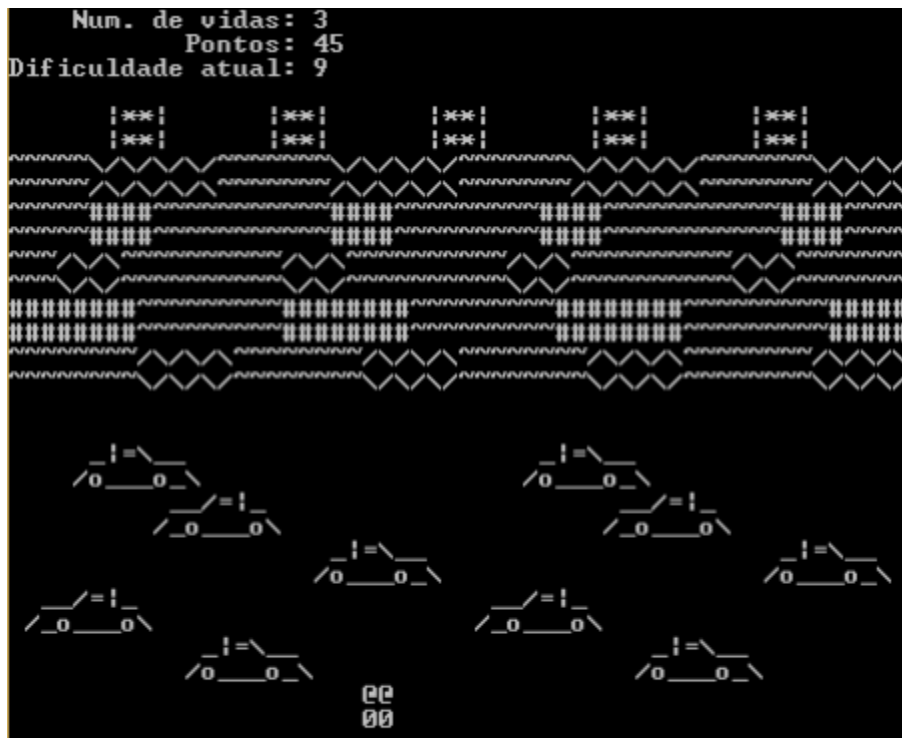
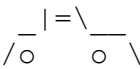
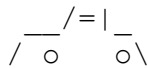





Figura 2: Exemplo de tela em ASCII.

Cada unidade do mapa é representada por uma matriz 2x2 de caracteres, e o mapa é dado por uma matriz de 13x28 de unidades. Sendo assim, o mapa do jogo ocupará um tamanho total de 26x56 caracteres na tela (além do cabeçalho de três linhas, que informa o número de

vidas, a pontuação e a dificuldade atual). Note que na versão do jogo mostrada na Figura 1, o mapa tem o tamanho 13x14, mas neste trabalho pede-se que o aluno implemente com tamanho 13x28, como é mostrado na Figura 2.

Cada objeto no mapa é constituído de um número inteiro de unidades, acopladas horizontalmente. Por exemplo, um objeto de 3 unidades deve ocupar o tamanho de uma matriz 2x6 de caracteres. Os objetos a serem utilizados no jogo são dados na tabela a seguir:

Sapo (2x2)	Casas (2x4)	Água (2x2)	Carro dir. (2x8)	Carro esq. (2x8)
@@ 00	** **	~~ ~~		
Objetos de tamanho variável:		Tartaruga (2x(2*N))	Tartaruga submergindo (2x(2*N))	Tronco (2x(2*N))
(nos exemplos a seguir, N = 3)				

(*espaços livres são dados por matrizes 2x2 de espaços “ ”).

2.3 CARACTERÍSTICAS DE CADA OBJETO

Cada um dos objetos mostrados possuem características individuais:

- **Sapo:** Pode movimentar-se lateralmente e verticalmente, pode subir sobre troncos e tartarugas (para fazer a travessia do rio). Não pode ultrapassar os limites do mapa, não pode estar na mesma unidade que um veículo ou água (condições onde o jogador perde uma vida). Quando estiver sobre umas das casas finais, a pontuação obtida na travessia é somada a pontuação total, e o sapo deve voltar a posição inicial (ao centro da linha de início);
- **Espaço vazio e água:** são componentes fixos do cenário, portanto não se movimentam. Sapo e carros movimentam-se sobre o espaço vazio. Troncos e tartarugas movimentam-se sobre a água;
- **Carros:** Movimentam-se horizontalmente apenas a direita ou a esquerda, nas linhas que representam a estrada. Carros na mesma linha possuem a mesma distância entre si e velocidade fixa;
- **Troncos e tartarugas:** Movimentam-se horizontalmente apenas para a direita ou para a esquerda, nas linhas que representam o rio. É permitido que o sapo movimente-se sobre eles. Quando o sapo subir em um tronco ou tartaruga, ele deve movimentar-se junto com este objeto. As tartarugas podem periodicamente submergir, fazendo com que o sapo se afogue caso ele esteja sobre uma que submergiu. Pouco antes de

submergir completamente, as tartarugas devem “avisar” o jogador que vão submergir, através da mudança de layout para o objeto “tartaruga submergindo”, e depois devem desaparecer por um pequeno período de tempo. Troncos e tartarugas na mesma linha possuem a mesma distância entre si e velocidade fixa.

Quando um objeto móvel (veículo, tronco ou tartaruga) atingir a borda do mapa, este deve imediatamente reiniciar sua trajetória, ou seja, retornar ao início da linha no sentido em que estava percorrendo-a. A velocidade destes objetos móveis é fixa por linha, de maneira que dois ou mais objetos em uma mesma linha possuam a mesma velocidade. Entretanto, linhas diferentes podem ter velocidades diferentes, e estas velocidades podem aumentar a medida que o jogo avança, caracterizando um aumento no nível de dificuldade.

2.4 MENU INICIAL

O menu inicial deverá conter as seguintes opções :

- **Novo Jogo:** inicia uma nova partida;
- **Continuar:** continua a última partida salva;
- **Sair:** fecha o programa.

2.5 OPERAÇÕES DE LEITURA E ESCRITA DE ARQUIVOS

2.5.1 INICIALIZAÇÃO DO JOGO

A definição sobre os objetos na inicialização da fase é dada em um arquivo binário (**fase.bin**). Esse arquivo possui diversas entradas do tipo TIPO_FASE, dado abaixo, com as informações necessárias sobre cada objeto.

```
typedef struct tipo_fase
{
    char objeto;
    int tamanho;
    int espacamento;
    float velocidade;
    int linha_inicial;
    int coluna_inicial;
} TIPO_FASE;
```

O significado de cada campo é dado na tabela a seguir:

Campo	Significado
objeto	Caractere que especifica o tipo de objeto, podendo ser: S: sapo T: tronco

	R: tartaruga C: carro
tamanho	Número de unidades que o objeto terá. Para troncos, indica o tamanho de cada tronco. Para tartarugas, indica o número de tartarugas adjacentes.
espacamento	Número de unidades que separam o final de um objeto ao início do outro objeto do mesmo tipo ao longo de uma linha.
velocidade	Valor real entre -1 e 1. O sinal indica o sentido (positivo para a direita, negativo para a esquerda). O módulo indica o quão rápido o objeto se move (valores próximos de 1 indicam velocidade máxima)
linha_inicial	índice da linha onde o caractere inferior esquerdo do objeto será inicializado. Será sempre um valor par, sendo o valor zero a primeira linha abaixo do cabeçalho.
coluna_inicial	índice da coluna onde o caractere inferior esquerdo do objeto será inicializado. Será sempre um valor par, sendo o valor zero a primeira coluna abaixo do cabeçalho.

2.5.2 CARREGAR/SALVAR JOGO

Deve ser possível, a qualquer momento no jogo, salvar o estado atual da partida, de modo que o usuário possa continuar o jogo em um momento futuro. Você tem liberdade para definir como será o formato do arquivo, mas também deverá ser salvo uma “captura da tela” no momento em que o jogo é salvo. Tal captura será um arquivo texto (use o nome “tela_salva.txt”).

2.6 PONTUAÇÃO E FIM DE JOGO

A pontuação do jogador é dada da seguinte forma:

- O jogador inicia a partida com 100 pontos;
- A cada timestep ou movimentação, é subtraído 1 ponto do jogador;
- Quando este alcançar a casinha, ganha +100 pontos;
- Se a pontuação chegar a 0, o jogo se encerra.

Além da pontuação em zero, existem outras duas condições de encerramento:

- o jogador perder todas as suas 3 vidas;
- o jogador pressionar a tecla ‘Q’.

2.7 NÍVEL DE DIFICULDADE

O nível de dificuldade deve aumentar a medida que o jogo avança. Nesta implementação, a dificuldade é caracterizada pelo aumento da velocidade dos objetos móveis, da seguinte forma: sempre que o sapo atingir a casinha, deve-se aumentar ligeiramente a velocidade destes objetos. A maneira como isso será feito fica a critério do aluno.

3 REQUISITOS MÍNIMOS

O que se espera do programa:

- O programa deve implementar todas as regras descritas até aqui;
- O código fonte deve compilar sem apresentar erros de compilação;
- O executável deve rodar sem mostrar erros de execução;
- Manipulação dos arquivos:
 - fase.bin;
 - tela_salva.txt;
- Uso de *structs*;
- Aumento gradual do nível de dificuldade;
- Salvar e continuar um jogo;
- Identação e documentação (comentários) do código fonte;
- Modularização do código (uso de funções e passagem de parâmetros).

4 FUNCIONALIDADES OPCIONAIS

Pontos extras serão atribuídos a quem desenvolver soluções para os desafios propostos. Entretanto, o aluno pode receber nota máxima se implementar de forma correta todos os itens obrigatórios. Outros desafios poderão ser considerados conforme sugestão dos alunos e avaliação pelo professor:

- Uso de cores diferentes para elementos do cenário;
- Implementar mais objetos móveis no cenário, inspirados no jogo original (como outros tipos de carros);
- Implementar novos desafios que garantam uma pontuação extra ao jogador, como por exemplo:
 - pegar algum objeto solto no mapa;
 - chegar a uma das casinhas movimentando-se apenas na direção vertical;

5 RODADA DE TESTE NA APRESENTAÇÃO

Na apresentação do trabalho, além da demonstração do programa funcionando com os arquivos auxiliares do próprio aluno, será fornecido um novo arquivo **fase.bin** para teste.

6 DATAS IMPORTANTES

- Até o dia 22 de maio, os alunos devem enviar por e-mail (crjung@inf.ufrgs.br) os nomes completos dos componentes das duplas. Podem ser formadas duplas entre alunos das turmas G e H, mas ambos devem estar disponíveis para apresentar às 08:30 ou 10:30 no dia 24 de junho.
- Até o dia 24 de junho, 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 o sistema operacional)
- O trabalho será obrigatoriamente apresentado durante a aula prática do dia 26 de junho. Ambos os membros da dupla deverão saber responder perguntas sobre qualquer trecho do código;
- Os seguintes itens serão considerados na avaliação do trabalho:
 - estruturação do código em módulos;
 - documentação geral do código (comentários, indentação);
 - “jogabilidade” do jogo;
 - atendimento aos requisitos definidos;
- Importante: trabalhos copiados não serão considerados. Saibam que há ferramentas que possibilitam a detecção automática de plágio.