Trabalho Final

Algoritmos e Programação – 2023/1

Prof. Dr. Thiago L. T. da Silveira
tltsilveira@inf.ufrgs.br

Instruções

- Leia este documento **atentamente**. Nele, estão descritos os requisitos mínimos e desejáveis (extras) quanto à implementação do Trabalho Final. Também são apresentadas diretrizes para entrega e critérios de avaliação.

Objetivo

O objetivo principal do Trabalho Final é exercitar conceitos de programação estudados ao longo do semestre através da implementação de um **jogo de computador** utilizando a linguagem C. Objetivos secundários incluem exercitar as habilidades de pesquisa – uma vez que o aluno poderá utilizar bibliotecas e conceitos que não foram trabalhados em aula para implementar certas funcionalidades – e de trabalho em equipe – uma vez que este deve ser realizado **por duplas ou trios**. O Trabalho Final *não* pode ser implementado individualmente.

Temática e Funcionamento Básico

O jogo a ser implementado é inspirado em **The Binding of Isaac** (acesse https://bindingofisaac.fandom.com/wiki/The_Binding_of_Isaac_Wiki para conhecer). Nesse jogo, o personagem **Isaac** (controlado pelo jogador) tem de passar por diversos desafios, se esquivando de fogueiras e derrotando inimigos com seu choro. O jogo é dividido em fases: cada fase tem um portal para a seguinte, com excessão da última – onde o personagem atinge seu objetivo. Todos os inimigos da fase devem ser eliminados para que o personagem possa, ao chegar no portal, mudar de fase. Isaac pode coletar bombas que podem ser usadas para derrotar seus inimigos.

Cada fase tem uma configuração de mapa distinta, indicando uma posição inicial do personagem e a posição do portal. Cada mapa também tem paredes, bombas coletáveis, áreas de trânsito (posições vazias), armadilhas (como espinho ou fogo) e inimigos que dificultarão a chegada do personagem até o portal. O jogador inicia com três vidas e perde uma cada vez que encostar em um inimigo (sem chorar ou lançar uma bomba) ou armadilha. O personagem derrota o inimigo caso o ataque com seu choro poderoso ou bomba. Quando o personagem perde uma vida, caso haja outras, a fase se reinicia. Caso contrário, considera-se *game over* e volta-se ao menu principal.

Requisitos Mínimos

Implementações do Trabalho Final devem respeitar os seguintes requisitos mínimos:

- Os elementos visuais do jogo devem ser implementados e exibidos em modo texto (através de caracteres).
- O jogo não deve ter *delay*, ou seja, ao ser disparada uma ação do jogador, o jogo deve responder imediatamente (exceto quando o *delay* é deliberado). Por exemplo: se o jogador movimentar o personagem para direita, ele deve imediatamente ir para a direita, se o jogador atacar, o personagem deve realizar o ataque, etc.
- O mapa do jogo deverá ser carregado a partir de um arquivo texto nomeado "`mapa<número>.txt`" ("`mapa1.txt`", "`mapa2.txt`", "`mapa3.txt`", etc.), onde o valor numérico indica a fase correspondente ao mapa.
- Cada mapa deve ser estruturado como uma matriz de caracteres de 30 linhas por 60 colunas. O mapa deve ser fechado (isto é, cercado de paredes).
- Os itens do jogo devem ser representados no arquivo do mapa com os seguintes caracteres:

Caractere	Significado
J	Posição inicial do personagem
I	Inimigo
B	Bomba coletável
P	Portal para próxima fase
#	Parede
Espaço em branco	Área de trânsito (posição vazia)
X	Fogueira (armadilha)

- A interação do jogador com o jogo se dá ao pressionar teclas específicas. As ações e teclas são as seguintes:

Tecla	Significado
ESC	Acessa o menu superior e espera o jogador escolher uma opção
Teclas (A/a, D/d, S/s, W/w)	Move o jogador uma posição na direção indicada
Setas (←, →, ↓ e ↑)	Atira com o personagem na direção indicada
E	Posiciona uma bomba no mapa
R	Recomeça o jogo

- Personagem e inimigos não devem atravessar bombas ou paredes.
- O mapa de cada fase terá no máximo 15 inimigos (o número pode variar conforme a fase), podendo ter qualquer número de áreas de trânsito, bombas, fogueiras e paredes.
- Cada mapa deverá ter apenas uma posição inicial do personagem e um portal para a próxima fase (ou fim de jogo).
- Há diferentes tipos de inimigos no jogo (https://bindingofisaac.fandom.com/wiki/The_Binding_of_Isaac_Wiki), cada um com seu próprio padrão de movimento e ataques. Implemente ao menos um deles. **Dica:** escolha monstros mais simples, como o *Gaper* ou o *Pooter*.
- Se o personagem entrar em contato com uma fogueira ou com um inimigo (sem chorar ou lançar uma bomba), e o jogador possuir vidas extras, a quantidade de vidas do jogador é decrementada. Caso o jogador não possua vidas extras, o jogo termina (*game over*) em derrota. Quando um inimigo é atingido por um golpe, ele é eliminado.
- Os inimigos não devem ser capazes de passar pelas paredes, mas podem passar, sem danos, por fogueiras.
- Além do mapa da fase corrente, o jogo deve exibir a fase atual (1, 2, etc.), o tempo decorrido desde início do jogo e a quantidade de vidas do personagem.
- A pontuação do jogador é determinada em função do tempo que ele leva para atingir o fim do jogo (concluir a última fase). A pontuação não fica negativa (ela satura em zero).
- O programa deve ser capaz de funcionar com qualquer número de fases (até o máximo de 99). Por exemplo, se forem incluídos 99 arquivos de mapa disponíveis para o jogo (seguindo a ordem numérica), seu programa deve ser capaz de possibilitar que o jogador avance pelas 99 fases. O jogo deve ser entregue com pelo menos 5 fases.
- O menu (exibido sobre a tela do jogo, no cabeçalho ou rodapé da tela) deve ser navegável com o pressionar de teclas possuir as seguintes opções:

Tecla	Significado
N	Novo Jogo: quando o jogador seleciona esta opção, um novo jogo é iniciado. Todas as configurações do jogo atual são reinicializadas.
C	Carregar jogo: quando o jogador seleciona esta opção, um jogo previamente salvo deve ser carregado. Assuma que existe apenas no máximo um jogo salvo e que pode ser carregado.
S	Salvar jogo: quando o jogador seleciona esta opção, deve-se salvar todas as informações pertinentes do jogo em um arquivo, de modo que ele possa ser carregado e continuado do ponto em que foi salvo. Essas informações incluem posições do personagem e dos inimigos, número de vidas, fase atual, etc.
Q	Sair do jogo: quando o jogador seleciona esta opção, o jogo é encerrado, sem salvar.
V	Voltar: quando o jogador seleciona esta opção, deve-se retomar o do jogo, que continua do ponto em que parou quando o jogador acessou o menu.

Você deverá pesquisar a utilização de bibliotecas para criação de uma interface para o programa. Lembre-se que o requisito mínimo é uma representação visual em modo caractere. A escolha da biblioteca e sua utilização ficarão a cargo dos grupos. Algumas bibliotecas sugeridas são a `conio2` e a `ncurses`, usadas para desenvolver interfaces em modo texto, para Windows e para Linux respectivamente. O aluno é encorajado a desenvolver o jogo utilizando interfaces gráficas. Neste caso, algumas bibliotecas que poderiam ser utilizadas seriam: a `Allegro`, a `raylib` (altamente recomendável) e a `sdl2`. O aluno também pode utilizar bibliotecas que não foram mencionadas aqui.

As Figura 1 apresenta um mapa de exemplo em arquivo texto. Esse mesmo mapa pode ser exibido em um programa em modo texto (terminal) de diferentes formas (veja a Figura 2). A Figura 3 apresenta o mapa de exemplo sendo exibida em um programa que usa a biblioteca `raylib`.

```
############################################################
############################################################
##                                                        ##
##                  J                                     ##
##                                          B          P  ##
##         #######                                        ##
##         #                                              ##
##         #                                              ##
##         #               X          ####################
##         #                          #                   ##
############                          #                   ##
##                                    #                   ##
##                                    #                   ##
##                                    #                   ##
##     B                              #                   ##
##                        B           ######              ##
##         ######                     #                   ##
##         #                          #                   ##
##         #                          #                   ##
##         #        I                 #                   ##
##         #                          #                   ##
##         #                          #                   ##
##     #####                  #########                   ##
##     #                      #                           ##
########                  ######                          ##
##                                                        ##
##                                           B            ##
##                                                        ##
############################################################
############################################################
```

Figura 1: Exemplo de mapa.

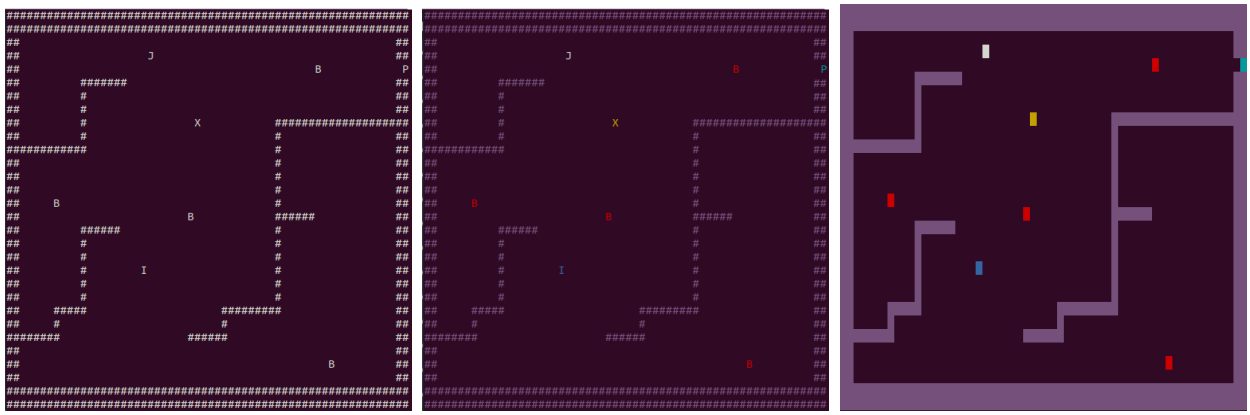

Figura 2: Representações em texto do mapa de exemplo.

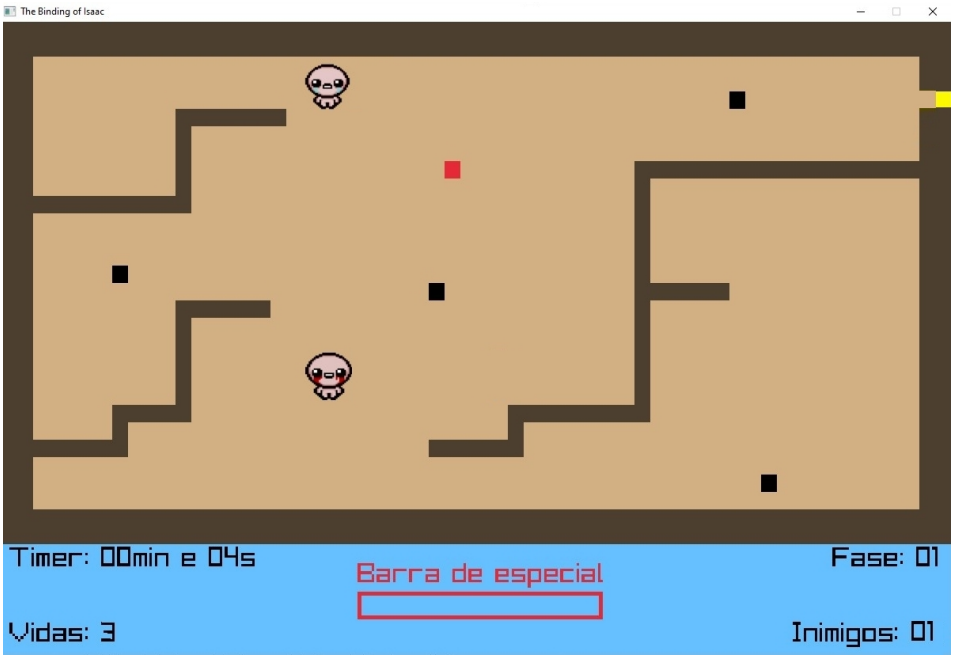

Figura 3: Representação gráfica do mapa de exemplo.

Requisitos Desejáveis (Extras)

Para os alunos motivados, as seguintes tarefas extras são propostas. Embora opcionais, essas tarefas podem melhorar a avaliação final do seu trabalho, caso não tenham conseguido implementar corretamente algum dos requisitos básicos.

- Você pode implementar outros superpoderes, como, por exemplo, habilitar modo "imune" por alguns segundos – o que permitiria passar pelos inimigos e fogueiras sem danos ou "parar o tempo" por alguns segundos – que faria com que os inimigos não se movessem.
- Você deve implementar, ao menos, um tipo de inimigo. Implementar mais de um tipo de inimigo é um bônus – desde que os diferentes tipos ajam de maneira diferente.
- Implemente "*cheats*": por exemplo, ao escrever "MUAHAHA" no teclado (diretamente em tempo de jogo, sem abrir menu adicional), o jogador fica invencível naquela fase.
- Emitir avisos sonoros quando o personagem receber dano de um inimigo, causar dano em um inimigo, se queimar na fogueira, passar de fase, perder ou ganhar o jogo, etc.
- Faça uma versão com representações visuais mais sofisticadas do jogo do que representação em modo texto. A sugestão do professor e monitores é usar a biblioteca *raylib* (`https://www.raylib.com/`). Você pode adicionar texturas para representar cada elemento do jogo.
- Faça com que o programa gerencie um arquivo binário "`highscores.bin`". Esse arquivo deve armazenar as 5 maiores pontuações do jogo. Ao fim de uma partida, se o jogador atingiu uma das 5 maiores pontuações, o programa deve requisitar o nome do jogador. Ao fim da partida, o programa sempre exibe o *ranking* dos 5 jogadores com a maior pontuação, da maior para a menor pontuação. Deve-se incluir uma opção no menu do jogo para visualizar o *ranking*.
- Combine o aumento do número de inimigos e fogueiras e a redução de bombas para gerar fases com dificuldade crescente.
- Sejam criativos, conversem com o professor e monitores e implementem suas ideias!

Entrega

O trabalho será entregue em **3 (três) etapas**:

1. Indicação dos grupos: até dia **21 de Julho de 2023** às 23:59h pelo ambiente virtual Moodle. O trabalho deverá ser realizado em grupos de dois a três alunos. O grupo deve informar os nomes dos integrantes do grupo até o dia 21 de Julho, através da plataforma Moodle. Os alunos que não tiverem definido grupos até esta data terão um grupo atribuído aleatoriamente.
2. Entrega do código: até dia **10 de Setembro de 2023** às 23:59h pelo ambiente virtual Moodle. Até o dia 10 de Setembro, o grupo deverá submeter via Moodle um arquivo `zip` cujo nome deve conter o(s) nome(s) do(s) aluno(s). O arquivo `zip` deve conter:
   - Um relatório contendo a descrição do trabalho realizado, incluindo a especificação de como os elementos do jogo foram representados, como foi implementada a interação dos componentes interativos, bem como as estruturas e funções utilizadas e uma explicação de como usar o programa. O relatório pode ser simples e objetivo, mas deve conter todas essas informações.
   - Os códigos fontes devidamente organizados e documentados (.c).
   - Os códigos executáveis já compilados e prontos para serem executados.
3. Apresentação: dias **11 e 13 e Setembro de 2023** das 08:30 às 10:10h – Turmas G e H, respectivamente.
   - O trabalho será obrigatoriamente apresentado durante as aulas dos dia 11 e 13 de Setembro. Todos os membros do grupo devem saber responder perguntas sobre qualquer trecho do código e estes serão avaliados separadamente.
   - Até o dia 11 de Agosto, os alunos deverão fazer um breve relato a respeito do andamento do trabalho. Esse relato deve ser feito através de um vídeo subido no Youtube (como *unlisted*) O *link* deve ser informado em fórum específico. É esperado que, neste momento, os alunos demonstrem pelo menos: a manipulação das informações do mapa e sua exibição e a movimentação dos personagens. Caso o grupo não apresentar este mínimo neste momento, todos os integrantes perdem 5% da nota final do trabalho.

Avaliação

Os seguintes itens serão considerados na avaliação do trabalho: estruturação do código em módulos, documentação geral do código (comentários, indentação), "jogabilidade" do jogo e atendimento aos requisitos definidos. A divisão da pontuação é a seguinte:

1. Habilidade em estruturar programas pela decomposição da tarefa em subtarefas, utilizando subprogramação para implementá-las (2 pontos).
2. Organização e documentação de programas (indentação, utilização de nomes adequados para variáveis e constantes, abstração dos procedimentos para obter maior clareza, uso de comentários no código) (1 ponto).
3. Domínio na utilização de tipos de dados simples e estruturados (arranjos e estruturas) e passagem de parâmetros por cópia e referência (2 pontos).
4. Atendimento dos requisitos do enunciado do programa: menus, elementos gráficos esperados, interação, movimentação de personagens, funções dos menus, etc. (5 pontos).

Dicas

- Use uma IDE (como o Code::Blocks) para desenvolvimento.
- Alguns alunos relatam problemas com o uso da biblioteca `conio` com as últimas versões do Code::Blocks. Nestes casos, recomenda-se utilizar umas versão anterior o Code::Blocks (como a 17.12).
- Utilizar `structs` para representar os elementos dinâmicos do jogo.
- Alternativas para entrada de dados não bloqueantes (i.e., não esperam o usuário digitar a entrada e pressionar `ENTER`) podem ser encontradas na biblioteca `conio.h`. Veja os exemplos fornecidos na apresentação do enunciado, disponíveis no moodle. Ver funções como `kbhit` e `getch`.
- A biblioteca `windows.h` contém a função `SetConsoleCursorPosition` que pode ser usada para definir a posição do cursor do terminal.
- Além da `conio.h`, existem outras bibliotecas que podem ser usadas, como a `ncurses`, para Linux. Além disso, existem bibliotecas mais avançadas que permitem gerar apresentações visuais mais sofisticadas, como `Allegro`, `raylib` (preferível), `sdl2`, etc., para criação de interfaces gráficas.
- Para o jogo não executar muito rápido, pode-se utilizar a função `sleep`.
- Enquanto ainda não tivermos estudado como manipular arquivos, é possível inicialmente representar o conteúdo do mapa que deveria ser carregado diretamente em código (*hardcoded*), e utilizar essa informação para elaborar a representação visual e movimentação. Depois de estudar a manipulação de arquivos, essa informação do mapa pode ser carregada dos arquivos.
- É comum, à medida que formos estudando novos conteúdos, surgirem ideias de como utilizar esses conteúdos para melhorar o jogo. Isso acaba gerando modificações constantes no jogo. Isso é esperado.
- Verifique o material adicional oferecido no Moodle como suporte para este trabalho.
- Considere o uso e repositórios remotos de código (*Github*, *Gitlab*, *Bitbucket*) com controle e versão para desenvolvimento em equipe.
- Recorra ao monitor para sanar dúvidas gerais sobre instalação de bibliotecas, sobre dúvidas a respeito do uso das biblitotecas adicionais, uso de repositórios (*github*), uso de ferramentas e implementação de funcionalidades associadas ao jogo.

Observações

Este trabalho deve refletir a solução individual do grupo para o problema proposto. Casos de plágio (mais de 75% de similaridade – incluindo entregas de outros semestres) serão tratados com severidade e resultarão em anulação da nota do Trabalho Final (vide programa da disciplina). Para detectar e quantificar o plágio no código, será utilizado o *software* MOSS (`http://theory.stanford.edu/~aiken/moss/`).