INSTITUTO DE INFORMÁTICA UFRGS

INF01202 - ALGORITMOS E PROGRAMAÇÃO
Profs. Claudio Jung e Dennis Balreira - Turmas C/D
Trabalho Prático Final - Semestre 2024/2
Bomberman

1. Instruções:

Leia com atenção este documento. Estará descrito nele o objetivo, principais instruções, requisitos mínimos e extras para a implementação do Trabalho Final. Além disso, serão apresentadas as diretrizes de entrega.

2. Objetivo

Durante o curso de **Algoritmos e Programação**, são explorados diversos conceitos e técnicas para ensinar aos alunos as principais habilidades lógicas e práticas necessárias para criar algoritmos e resolver uma variedade de problemas de programação. Este **Trabalho Final** busca consolidar o aprendizado em relação ao conteúdo teórico e prático abordado e exercitar a habilidade de trabalho em equipe, consistindo na implementação de **um jogo na linguagem de programação C**.

O jogo a ser implementado é uma versão simplificada do jogo ***BomberMan***, utilizando a biblioteca gráfica **RayLib (https://www.raylib.com/)**. O jogo que deverá ser implementado é inspirado no jogo chamado "Bomberman", e a Fig. 1 ilustra uma tela do jogo original (uma versão do jogo encontra-se em https://www.retrogames.cz/play_085-NES.php#google_vignette.

A mecânica do nosso jogo será similar ao do Bomberman: o personagem terá de usar bombas e chaves para alcançar o objetivo e haverá obstáculos e inimigos no mapa que irão dificultar este processo. O jogo é estruturado em diferentes níveis, onde cada nível é uma sala com uma configuração diferente. A missão do personagem, em cada sala, é coletar cinco chaves para passar para a próxima sala. As chaves estarão espalhadas pelo mapa e escondidas dentro de caixas; porém, nem toda caixa possui uma chave dentro. O jogador possui bombas que são usadas para destruir as caixas, e após a destruição das caixas se pode verificar quais delas possuem chaves.

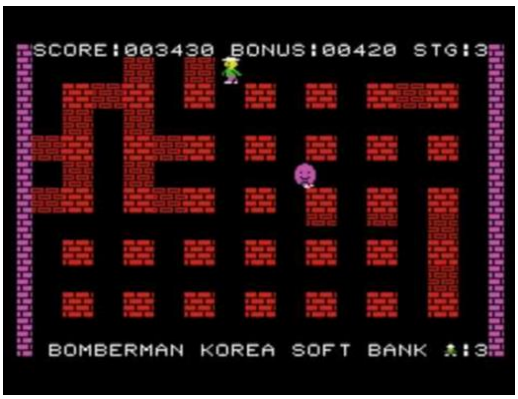

Figura 1: *Screenshot* do jogo Bomberman para MSX (1983)

O jogador também deve tomar cuidado com inimigos que se movem pela tela e que podem causar danos ao jogador. As bombas plantadas pelo jogador podem explodir caixas, matar inimigos, destruir partes destrutíveis das paredes e gerar dano no jogador (elas não afetam as paredes indestrutíveis). Então, tome muito cuidado ao utilizar as bombas.

3. Requisitos mínimos (obrigatórios)

A implementação realizada pelos alunos deverá respeitar os seguintes requisitos mínimos:

- O jogo não deve ter delay, ou seja, ao ser disparada uma ação do jogador, o jogo deve responder imediatamente. Por exemplo, se o jogador apertar a tecla para movimentar o personagem para direita, o personagem deve imediatamente ir para a direita;
- Cada mapa é representado por um arquivo texto com 25 linhas e 60 colunas. Os caracteres no arquivo texto que representam cada elemento do jogo são dados abaixo, que devem ser mapeados para elementos gráficos usando a Raylib (um exemplo de mapa é dado na Figura 2). A janela do jogo deverá ter o tamanho de 600 pixels de altura x 1200 pixels de largura, e cada caraceter representa um bloco de 20 x 20 pixels. A parte inferior da tela (100 pixels) deve ser reservada para impressão das vidas restantes do personagem, pontuação e número de bombas disponíveis no arsenal. Um exemplo de tela de jogo é mostradop na Figura 3.

Caractere no arquivo	**Significado**
(espaço em branco)	Posição vazia
J	Posição inicial do jogador
W	Parede indestrutível
D	Parede destrutível
K	Caixa com chave
B	Caixa sem chave
E	Inimigo

```
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWE WWWWWW     WWWWWW     WWWWWW   WWWWWW  BB   WWWWWWBEWWW
WWW    WW        KWW         WW         WW        KWW D  WWW
WWWWW      WWBWW      WW WW      WW WW      WW WW     DWWWWW
WWW  WW  WW DDD WW  WW     WW  WW     WW  WW     WW  WW  WWW
WWW    WW    W  K WW    W    WW    W    WW    W   KWW    WWW
WWW  WW  WW     WW  WW     WW  WW     WW  WW     WW  WW  WWW
WWWWW      WW WW      WW WWDD   BWW WW      WW WW      WWWWW
WWW               WW         WW         WW               WWW
WWWWW      WW WW      WW WW      WWEWW      WW WW      WWWWW
WWW  WW  WWB    WW  WW     WW  WW     WW  WW     WW  WW KWWW
WWW    WW  DDW    WW    W    WW    W    WW    W    WW    WWW
WWW  WW  WW     WW  WW     WW  WW     WW  WW  B  WW  WW  WWW
WWWWW      WW WW      WW WW  DD  WW WW      WW WW      WWWWW
WWW               WW         WW    D    WW               WWW
WWWWW      WW WW      WW WW      WW WW      WW WW      WWWWW
WWW  WW  WW    BWW  WW     WW  WW     WW  WW     WW  WW  WWW
WWW    WW    WDD  WW    W    WW    W    WW  DDW    WWE   WWW
WWW  WW  WW     WW  WW     WWB WW     WW  WW     WW  WW  WWW
WWWWW    DDWW WW      WW WW  DD  WWEWW      WW WW      WWWWW
WWW    WW         WW         WW         WW         WW    WWW
WWWJ WWWWWW     WWWWWW     WWWWWW     WWWWWW     WWWWWW  WWW
WWW               D                     D                WWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
```

Figura 2: exemplo de mapa do jogo

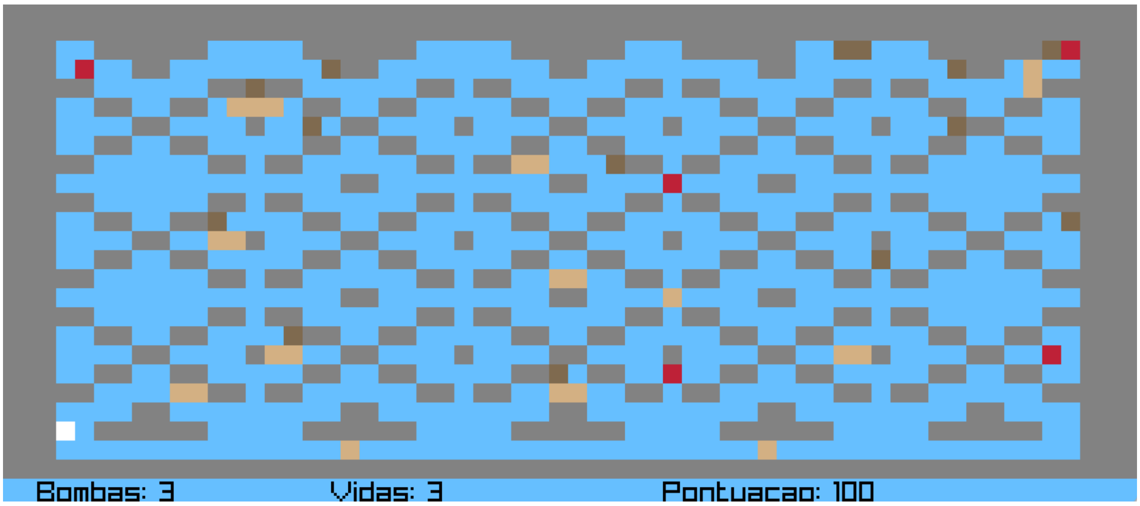

Figura 3: exemplo de tela de jogo

- Caixas com chave e caixas sem chave devem ser exibidas de forma **indistinguível** na tela.
- Quando uma caixa com chave é <u>explodida</u>, uma chave deve ser exibida no seu lugar
- Sobre a física do jogo:
  - Bombas só podem ser plantadas em áreas vazias

    - Bombas, paredes (indestrutíveis ou não) e caixas representam obstáculos. Isto é, o jogador e os inimigos não podem ocupar o mesmo espaço que esses elementos no mapa e durante a interação.
    - Dois inimigos não podem ocupar o mesmo lugar no espaço.
    - Os movimentos do jogador devem respeitar a configuração do mapa, impedindo que ele possa se mover através de paredes ou obstáculos.
- Sobre o estoque e bombas:
    - O personagem possui inicialmente 3 bombas.
    - Quando o personagem planta uma bomba, o estoque de bombas no arsenal é decrementado.
    - Quando uma bomba plantada explode, o estoque de bombas no arsenal é incrementado novamente.
- Sobre a pontuação do jogo:
    - O jogador ganha 20 pontos por inimigo explodido
    - O jogador ganha 10 pontos por caixa ou obstáculo destrutível explodido
    - O jogador perde 100 pontos quando perde vida
    - A pontuação nunca fica negativa
- O personagem perde vida em duas situações:
    - Quando é tocado por um inimigo
    - Quando está no raio de destruição de uma bomba quando ela explode.
- Movimentação dos inimigos:
    - No início do nível, cada inimigo deve seguir em uma direção aleatória.
    - A cada período de tempo (definido pelo programador), o inimigo sorteia aleatoriamente uma nova direção.
    - Sempre que o inimigo não puder se movimentar para a direção corrente (por exemplo, se houver uma parede), deve ser selecionada uma direção aleatória que será utilizada na próxima tentativa de se mover.
- A interação do jogador com o jogo deve ser realizada com as seguintes teclas e suas respectivas ações:
    - TAB: Acessa o menu superior e espera o jogador escolher
    - Setas (←,→,↓,↑), ou teclas (A,D,S,W): movem o jogador uma posição na direção indicada
    - tecla B: Planta bomba
- O menu (que pode ser exibido no cabeçalho ou rodapé da tela) deve possuir as seguintes opções:
    - (N) Novo Jogo: Quando o usuário seleciona esta opção, um novo jogo é iniciado, a partir da primeira fase e toda a pontuação, número de bombas e vidas do jogador também é resetado.

    - (C) Carregar jogo: Quando o usuário 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 usuário 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 das chaves/caixas, obstáculos, paredes, posição do jogador; número de bombas, vidas, fase jogada, etc. Você deve obrigatoriamente usar um **arquivo binário** para salvar o jogo.
    - (Q) Sair do jogo: Quando o usuário seleciona esta opção, o jogo é finalizado, sem salvar.
    - (V) Voltar: Quando o usuário seleciona esta opção, deve-se abandonar o menu e o controle volta para o jogo, que continua do ponto em que parou quando o usuário acessou o menu.
- O jogo deve ter pelo menos dois mapas, mas deve funcionar para qualquer quantidade de mapas incluídos no diretório do programa executável. Cada mapa é definido por um arquivo texto (txt) específico, cujo nome estabelece a sequência de salas do jogo: mapa1.txt, mapa2.txt, mapa3.txt, etc.
- Um mapa deve ter exatamente cinco chaves e cinco inimigos.
- Cada bomba deve levar 3s para explodir após ser plantada (tecla B acionada). Sugere-se que a bomba atue em uma região em "formato de cruz" (direções horizontal e vertical) abrangendo uma região de 100 x 100 pixels centrado na posição da bomba.
- Ao pressionar a tecla B para plantar a bomba, ela deverá aparecer na tela uma posição à frente da do jogador
- A posição do jogador e dos inimigos deve ser obrigatoriamente implementada usando uma **estrutura (struct)** que contém a posição da linha e coluna do elemento.
- Você não pode usar variáveis globais no seu programa.

4. Tarefas extras

Caso sinta-se interessado, há tarefas extras propostas. Embora opcionais, essas tarefas podem melhorar a avaliação final do seu trabalho, como suprir alguma deficiência gerada em um requisito obrigatório.

- Armazenar as 10 pontuações mais altas em um arquivo. Ao fim do jogo, caso o usuário obtenha uma das 10 maiores pontuações, este deve ser alertado, por meio de uma frase como, por exemplo, "Você obteve a segunda maior pontuação! Parabéns!". Desta forma, sempre que o jogo for finalizado, faz-se necessária a verificação deste arquivo, para que o usuário possa ser alertado. Observa-se que enquanto não existir uma pontuação registrada, não é necessária a existência de um arquivo, que pode ser

criado somente no momento em que o primeiro usuário finalizar o jogo. Quando a primeira pontuação for registrada, as demais podem ser indicadas pelo valor zero.

- Emitir avisos sonoros em alguns eventos do jogo (plantar bomba, bomba explodir, etc.)
- Movimentação inteligente para os inimigos. Em vez os inimigos se movimentarem aleatoriamente, eles perseguem o jogador.
- **Seja criativo, implemente suas ideias,** mas não esqueça dos requisitos mínimos e converse com o professor antes!

5. Dicas

- Considere utilizar diversas *structs* para representar os elementos do jogo. Lembre-se que pode ser utilizado uma struct dentro da outra, como uma struct *posição* com x e y dentro de outra, como *jogador* ou *monstros*.
- Considere ir **desenvolvendo o jogo com antecedência,** modularizando-o ao implementar item por item para checar possíveis erros antes que se propaguem. Caso ainda não tenhamos aprendido a manipular arquivos, utilize o mapa diretamente no código (*hardcoded*) para ir implementando os demais recursos, e então quando souber carregar os arquivos altere esse recurso para atingir esse critério obrigatório.
- A exibição na tela exige a biblioteca Raylib. Para familiarização com essa biblioteca, verifique a prática opcional sobre esse assunto disponível no moodle.
- **Cuidado com o uso de ferramentas de Inteligência Artificial, pois serão desconsiderados códigos que não estejam alinhados com o aprendizado da sala de aula. Fica evidente quando esses recursos foram empregados.** Vocês têm capacidade para implementar esse trabalho com base nos seus próprios conhecimentos!

6. Outras informações

- O trabalho **deverá** ser realizado **em duplas**. Informar os componentes da dupla até o dia **13/12/2024** ao professor por e-mail (crjung@inf.ufrgs.br)
- Até o dia **08/01/2025 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 sucinta do trabalho e uma explicação de como usar o programa
    - os códigos-fonte devidamente organizados e documentados (arquivos .c)
    - o executável do programa
- O trabalho será **obrigatoriamente** apresentado durante as aulas práticas do dia **09/01/2025**. Ambos os membros da dupla deverão saber responder perguntas sobre **qualquer** trecho do código. Detalhes sobre essa apresentação serão fornecidos posteriormente

- No dia da apresentação serão fornecidos novos arquivos de mapas *Mapa01.txt*, *Mapa02.txt*, *Mapa03.txt* para testar o programa
- Os seguintes itens serão considerados na avaliação do trabalho:
    - estruturação do código em módulos
    - documentação geral do código (comentários, indentação)
    - "jogabilidade" do jogo (deve jogar em tempo-real)
    - atendimento aos requisitos definidos
- **Importante:** trabalhos copiados não serão considerados. Temos ferramentas que possibilitam a detecção automática de plágio.