INSTITUTO DE INFORMÁTICA UFRGS

INF01202 – ALGORITMOS E PROGRAMAÇÃO
Turmas A/B

Trabalho Prático Final
Semestre 2021/2 – Híbrido
BATTLE INF

1. MOTIVAÇÃO E OBJETIVOS

No decorrer da disciplina de algoritmos e programação são apresentados diversos conceitos e técnicas de programação, visando ensinar aos alunos as principais ferramentas lógicas e práticas para construção de algoritmos e solução de problemas diversos. Como passo fundamental, este trabalho vem consolidar 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 jogo conhecido como *Battle City* chamado *BattleINF* utilizando a biblioteca gráfica *Raylib*. OBS: Alunos interessados em modo texto (*ASCII art*) podem fazer também, mas antes conversar com o professor.

A implementação deve dispor de toda a interação do usuário (jogador) com o cenário (tanques inimigos, blocos e células de energia). Um exemplo do cenário do jogo pode ser visto na Figura 1 usando a *Raylib*. Além disso, pode-se entender melhor o jogo através de um gameplay.

Para os interessados em jogar a versão original, é possível realizar, gratuitamente, o download de um emulador para NES (sigla de Nitendo Entertainment System, o console original do jogo) e o download da rom do jogo original.

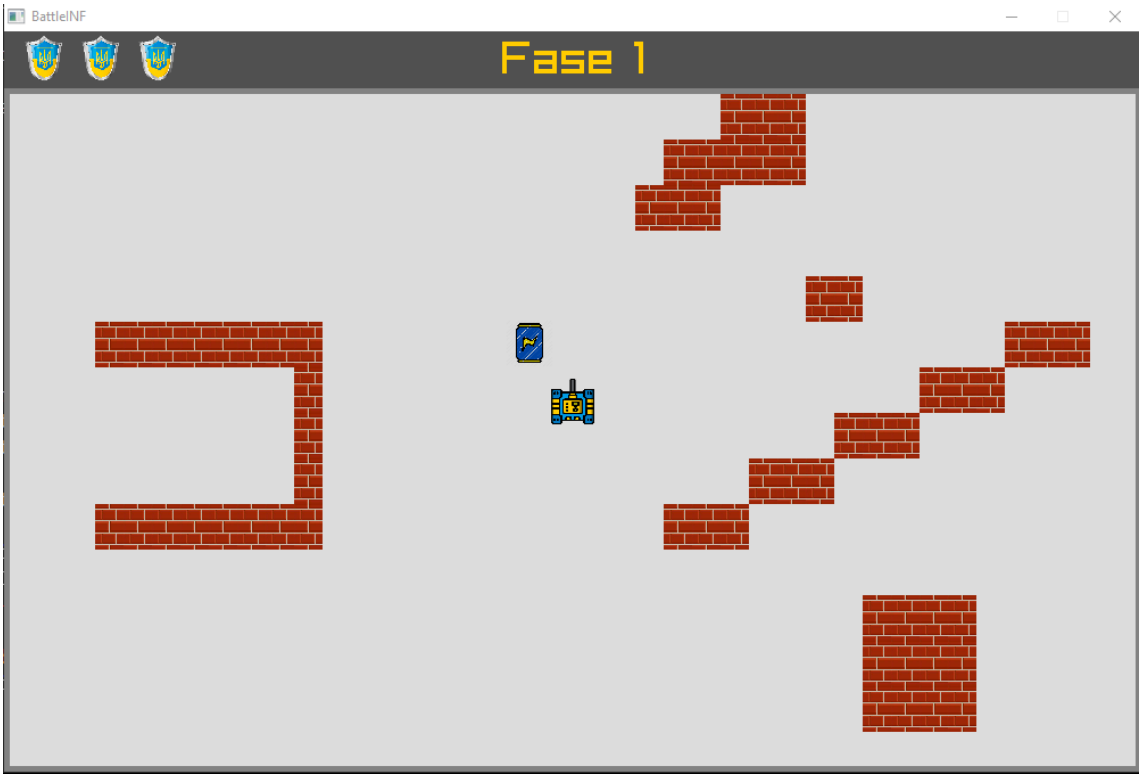


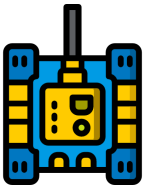
Figura 1: Exemplo de tela do jogo em Raylib

2. VISÃO GERAL DO JOGO

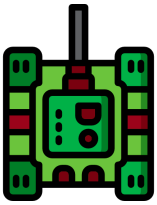
- **O jogador - tanque do jogador - possui as seguintes informações:**
  - Vidas
  - Pontuação
  - Localização (coordenadas x e y)
  - Pode mover-se para a esquerda, direita, para cima e para baixo. Além disto pode atirar
  - Perde uma vida quando atingido por algum **tiro disparado pelo inimigo**
- **Célula de Energia:**
  - Surgem esporadicamente no mapa, em alguma posição randômica.
  - Quando o jogador a coleta, a sua **velocidade** e a de seus projéteis **aumentam 50%**
- **Blocos:**
  - Obstáculos do mapa que podem ser destruídos por um tiro do jogador ou dos inimigos.
  - Nem o jogador nem os tanques inimigos podem transpor os obstáculos
- **Os tanques inimigos possuem as seguintes informações:**
  - Modo: Patrulha vs Perseguição
    - Modo Patrulha: Se deslocam em um sentido aleatório, que é alterado quando se colide com um bloco.
    - Modo Perseguição: Caso fique alinhado verticalmente ou horizontalmente com o tanque do jogador, começa a

perseguir e disparar contra este
    - Sentido de deslocamento (Aleatório no modo patrulha e seguindo o jogador no modo perseguição)
    - Velocidade
    - Localização (coordenadas x e y)
    - Se um tanque inimigo é destruído, o jogador ganha 800 pontos
    - Os tanques inimigos são obstáculos à movimentação do jogador e dos outros tanques inimigos
- **Projéteis:**
    - Um disparo inimigo NÃO destrói um outro tanque inimigo, mesmo que o acerte.
    - Quando um disparo, seja inimigo ou do jogador, acerta um bloco, esse é destruído juntamente com o disparo
    - Se um projétil do jogador atinge um projétil inimigo, ambos são destruídos

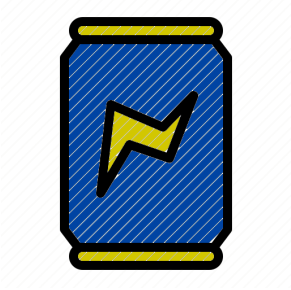
- **O espaço do jogo possui as seguintes informações:**
    - Nível
    - Jogador
    - Conjunto de inimigos
    - Obstáculos
    - Pontuação
    - Vidas
    - Deve ter dimensão 1000x600 pixels no Raylib (sem contar o cabeçalho no topo)
    - Os componentes do jogo são:
        - O jogador:



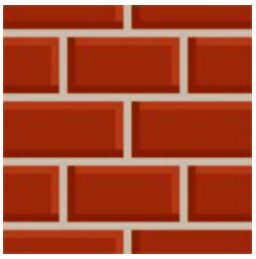
        - Os inimigos:



- Célula Energia ->



- Obstáculos ->



3. REQUISITOS MÍNIMOS (OBRIGATÓRIOS) PARA O PROGRAMA

Os seguintes itens devem estar **obrigatoriamente** implementados:

- O jogo começa com a apresentação do menu como mostrado na Figura 2
- As opções do Menu são:
  1. Novo jogo. Inicia um novo jogo com pontuações zeradas
  2. Continuar. Nessa opção se carrega um jogo salvo previamente
  3. Carregar Mapa. Nessa opção se carrega uma fase única de um arquivo
  4. Apresenta o ranking dos 5 jogadores com maiores pontuações (ver abaixo explicação do arquivo *highscores.bin*)
  5. Sair. Encerra o jogo
- Outros itens no menu principal (Ajuda e Sobre) são opcionais

Figura 2: Tela inicial do jogo em Raylib

- O **jogador** é controlado pelas setas do teclado
- O **jogador** atira com a tecla ‘Espaço’
- Os **inimigos** são gerados esporadicamente em posições aleatórias do mapa, **a cada 5 segundos (? Narciso verifica)**
- A cada ciclo de jogo, um número aleatório no intervalo [0,16] é sorteado. Caso o número seja 0, o inimigo atira
- Um **inimigo** morre se for atingido por um tiro do jogador. Nesse caso o **jogador** recebe 800 pontos
- Se o **jogador** conseguir exterminar todos os **inimigos**, o jogo muda para o próximo nível
- A célula de energia é gerada numa posição aleatória do espaço de jogo, sempre que o resultado de um número aleatório no intervalo [0,16] é zero
- Ao pressionar a tecla ‘S’ em qualquer momento durante o jogo, o programa salva num arquivo texto o estado atual do jogo. Esse jogo pode ser carregado posteriormente pela opção 2 do menu (ver na Figura 2). Você deve pensar uma maneira de salvar o estado corrente do jogo (escore, posições dos elementos, sentido e velocidade de movimento, etc.). Sempre o último jogo salvo deve ser carregado.
  Outra opção seria criar um submenu dentro do jogo que permitisse salvar a partida e voltar ao menu principal.
- Quando o programa iniciar, um arquivo texto chamado *nivel1.txt*, contendo as posições de todos os elementos do jogo, deve ser lido. À

medida que o jogador avança nas fases, essas são representadas por arquivos *nivel2.txt*, *nivel3.txt*, etc. Você deverá implementar pelo menos um nível a mais além do nível inicial. Mais precisamente, o arquivo conterá a posição central de cada elemento do jogo, definidas por caracteres como segue:
    - T - Posição inicial do tanque do Jogador
    - # - Blocos
- Esse arquivo texto tem tamanho de 40 colunas x 15 linhas. Cada coluna representa 25 pixels na horizontal e cada linha representa 40 pixels na vertical.
- Ao iniciar o jogo, deve ser carregado um arquivo binário *highscores.bin,* contendo as 5 maiores pontuações já registradas, juntamente com os nomes dos respectivos jogadores. No início esse arquivo tem 5 nomes fictícios. Quando o jogo for encerrado, o programa deve verificar se a pontuação do atual jogador é uma das cinco melhores. Caso afirmativo, perguntar o nome do jogador (ou no já ter perguntado no início do jogo) e atualizar o arquivo *highscores.bin*.

4. TAREFAS EXTRAS

Se você quer um desafio maior, as seguintes tarefas extras são propostas. Embora opcionais, essas tarefas podem melhorar a avaliação final do seu trabalho (nota máxima 11)

- Exibir efeito de explosão dos tanques
- Exibir efeito de destruição dos blocos
- Emitir avisos sonoros em alguma(s) da(s) situação(ões), por exemplo:
    - quando um inimigo é destruído
    - quando jogador atira
    - quando o jogo acabar
    - quando coletar Célula de Energia
    - etc…
- Seja criativo, implemente suas ideias (mas não esqueça dos requisitos mínimos e converse com o professor antes)!

5. DICAS

- O jogo pode ser implementado em qualquer sistema operacional
- Você pode usar qualquer ambiente para o desenvolvimento, como o CodeBlocks ou o VS Code
- A exibição na tela exige a biblioteca **Raylib**. Para familiarização com essa biblioteca, verifique a prática opcional sobre esse assunto disponível no moodle

6. OUTRAS INFORMAÇÕES

- O trabalho **deverá** ser realizado preferencialmente **em duplas**. Informar os componentes da dupla até o dia **15 de abril** ao professor por e-mail (marcelo.walter@inf.ufrgs.br)
- Até o dia **03 de maio 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
- O trabalho será **obrigatoriamente** apresentado durante a aula do dia **04 de maio de 2022 (4a feira) (excepcionalmente poderemos usar também o dia 05 de maio)**. 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 *nivel1*.txt e *nivel2*.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.