Contanto

E obrigado por todos os peixes!

Resumo:
Este projeto é um jogo 2D muito pequeno.
Seu objetivo é fazer você trabalhar com texturas, sprites e alguns outros elementos de jogo bem básicos.

Versão: 3

Conteúdo

Capítulo I

Prefácio

Ser um desenvolvedor é ótimo para criar seu próprio jogo.

Mas um bom jogo precisa de bons recursos. Para criar jogos 2D, você terá que procurar peças, conjuntos de peças, sprites e folhas de sprites.

Felizmente, alguns artistas talentosos estão dispostos a compartilhar seus trabalhos em plataformas como: itch.io

De qualquer forma, procure respeitar o trabalho dos outros.

Capítulo II

Objetivos

Chegou a hora de você criar um projeto básico de computação gráfica!

tanto tempo irá ajudá-lo a melhorar suas habilidades nas seguintes áreas: gerenciamento de janelas, manipulação de eventos, cores, texturas e assim por diante.

Você vai usar a biblioteca gráfica da escola: a MiniLibX! Esta biblioteca foi desenvolvida internamente e inclui ferramentas básicas necessárias para abrir uma janela, criar imagens e lidar com eventos de teclado e mouse.

As outras metas são semelhantes a todas as outras metas deste primeiro ano: ser rigoroso, subir de nível em programação C, usar algoritmos básicos, fazer pesquisas de informações e assim por diante.

Capítulo III

Instruções Comuns

- Seu projeto deve ser escrito em C.
- Seu projeto deverá ser escrito de acordo com a Norma. Se você tiver arquivos/funções bônus, eles serão incluídos na verificação de norma e você receberá 0 se houver um erro de norma dentro.
- Suas funções não devem encerrar inesperadamente (falha de segmentação, erro de barramento, liberação dupla, etc.) além de comportamentos indefinidos. Caso isso aconteça, seu projeto será considerado não funcional e receberá nota 0 na avaliação.
- Todo o espaço de memória alocado no heap deve ser liberado adequadamente quando necessário. Sem vazamentos será tolerado.
- Se o assunto exigir, você deve enviar um Makefile que irá compilar seus arquivos fonte para a saída necessária com as flags -Wall, -Wextra e -Werror, use cc, e seu Makefile não deve relinkar.
- Seu Makefile deve conter pelo menos as regras $(NAME), all, clean, fclean e ré.
- Para entregar bônus ao seu projeto, você deve incluir uma regra bônus em seu Makefile, que adicionará todos os diversos cabeçalhos, bibliotecas ou funções que são proibidas na parte principal do projeto. Os bônus devem estar em um arquivo diferente _bonus.{c/h} se o assunto não especificar mais nada. A avaliação da parte obrigatória e da parte bônus é feita separadamente.
- Se o seu projeto permitir que você use sua libft, você deve copiar seus fontes e seu Makefile associado em uma pasta libft com seu Makefile associado. O Makefile do seu projeto deve compilar a biblioteca usando seu Makefile e depois compilar o projeto.
- Nós encorajamos você a criar programas de teste para o seu projeto, mesmo que este trabalho **não precise ser enviado e não receba notas.** Isso lhe dará a chance de testar facilmente seu trabalho e o de seus colegas. Você achará esses testes especialmente úteis durante sua defesa. Na verdade, durante a defesa, você é livre para usar seus testes e/ou os testes do colega que está avaliando.
- Envie seu trabalho para o repositório git designado. Somente o trabalho no repositório git será avaliado. Se Deepthought for designado para avaliar seu trabalho, isso será feito

após suas avaliações por pares. Se ocorrer um erro em qualquer seção do seu trabalho durante a avaliação do Deepthought, a avaliação será interrompida.

Capítulo IV

Parte obrigatória

Nome do programa	so_long
Entregar arquivos	Makefile, *.h, *.c, mapas, texturas NAME, all, clean,
Argumentos	fclean, re Um mapa no formato *.ber
Makefile	
Funções externas.	• abrir, fechar, ler, escrever, malloc, free, perror, strerror, exit • Todas as funções da matemática biblioteca (opção do compilador -lm, man man 3 math) • Todas as funções do MiniLibX • ft_printf e qualquer equivalente VOCÊ codificou
Libft autorizado	Sim
Descrição	Você deve criar um jogo 2D básico no qual um golfinho escapa da Terra após comer alguns peixes. Em vez de um golfinho, um peixe e a Terra, você pode usar qualquer personagem, qualquer colecionável e qualquer lugar que desejar.

Seu projeto deve obedecer às seguintes regras:

- Você **deve** usar o MiniLibX. Tanto a versão disponível nas máquinas escolares, ou instalá-lo usando suas fontes.
- Você terá que entregar um Makefile que irá compilar seus arquivos fonte. Não deve vincular novamente.
- Seu programa deve tomar como parâmetro um arquivo de descrição de mapa terminando com .ber extensão.

IV.1 Jogo

- O objetivo do jogador é coletar todos os presentes colecionáveis no mapa e depois escapar escolhendo o caminho mais curto possível.
- As teclas W, A, S e D devem ser usadas para mover o personagem principal.
- O jogador deve ser capaz de se mover nestas **4 direções:** para cima, para baixo, para a esquerda, para a direita.
- O jogador não deve ser capaz de se mover contra paredes.
- A cada movimento, o **número atual de movimentos** deve ser exibido no shell.
- Você deve usar uma **visualização 2D** (de cima para baixo ou de perfil).
- O jogo não precisa ser em tempo real.
- Embora os exemplos dados mostrem um tema de golfinho, você pode criar o mundo que deseja querer.

Se preferir, você pode usar ZQSD ou as teclas de seta do teclado para mover seu personagem principal.

IV.2 Gestão gráfica

- Seu programa deve exibir a imagem em uma janela.
- A gestão da sua janela deve permanecer tranquila (mudança para outra janela dow, minimização e assim por diante).
- Pressionar ESC deve fechar a janela e sair do programa de forma limpa.
- Clicar na cruz na moldura da janela deve fechar a janela e sair do programa de forma limpa.
- O uso das imagens do MiniLibX é obrigatório.

IV.3 Mapa

- O mapa deve ser construído com 3 componentes: **paredes, itens colecionáveis** e **itens gratuitos espaço.**

- O mapa pode ser composto apenas por estes 5 caracteres: **0** para um espaço vazio, **1** para uma parede, **C** para um colecionável, **E** para uma saída do mapa, **P** para a posição inicial do jogador.

  Aqui está um mapa simples e válido:

```
1111111111111
10010000000C1
1000011111001
1P0011E000001
1111111111111
```

- O mapa deve conter **1 saída,** pelo menos **1 colecionável** e **1 posição inicial** para ser válido.

Se o mapa contiver caracteres duplicados (saída/início), você deverá exibir uma mensagem de erro.

- O mapa deve ser retangular.

- O mapa deve ser fechado/cercado por paredes. Caso contrário, o programa deve retornar um erro.

- Você deve verificar se existe um caminho válido no mapa.

- Você deve ser capaz de analisar qualquer tipo de mapa, desde que respeite as regras acima.

- Outro exemplo de mapa .ber mínimo:

```
1111111111111111111111111111111111
1E000000000000C00000C000000000001
1010010100100000101001000000010101
1010010010101010001001000000010101
1P0000000C00C0000000000000000000C1
1111111111111111111111111111111111
```

- Se qualquer tipo de configuração incorreta for encontrada no arquivo, o programa deverá sair de forma limpa e retornar "Erro\n" seguido de uma mensagem de erro explícita de sua escolha.

Capítulo V

Parte bônus

Normalmente, você seria incentivado a desenvolver seus próprios recursos extras originais. Porém, haverá projetos gráficos muito mais interessantes posteriormente. Eles estão à tua espera!!
Não perca muito tempo nesta tarefa!

Você tem permissão para usar outras funções para completar a parte bônus, desde que seu uso seja **justificado** durante sua avaliação. Seja esperto!

Você ganhará pontos extras se:

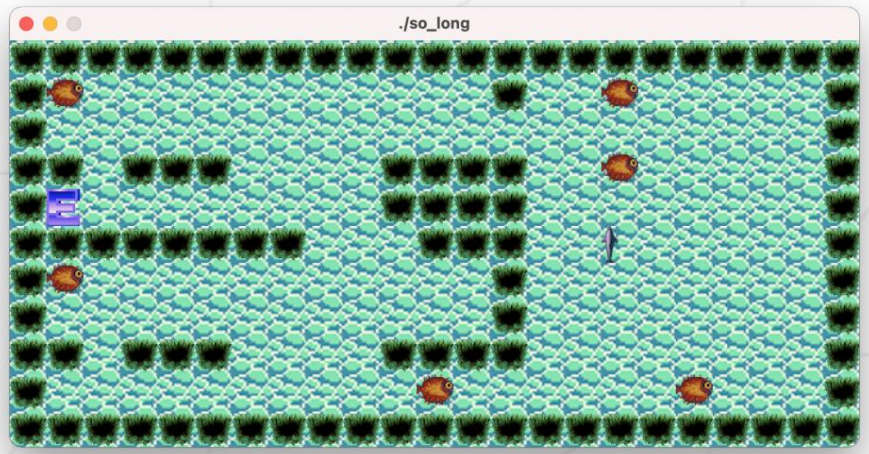
- Faça o jogador perder ao tocar em uma patrulha inimiga.
- Adicione alguma animação de sprite.
- Exiba a contagem de movimentos diretamente na tela em vez de escrevê-la no shell.

Você pode adicionar arquivos/pastas com base nos bônus, conforme necessário.

A parte bônus só será avaliada se a parte obrigatória for PERFEITA. Perfeito significa que a parte obrigatória foi feita integralmente e funciona sem mau funcionamento. Se você não passou em TODOS os requisitos obrigatórios, sua parte do bônus não será avaliada de forma alguma.

Capítulo VI

Exemplos



exemplos so_long mostrando péssimo gosto em design gráfico
(quase valem alguns pontos extras)!

Capítulo VII

Envio e avaliação por pares

Entregue sua tarefa em seu repositório Git normalmente. Somente os trabalhos dentro do seu repositório serão avaliados durante a defesa. Não hesite em verificar os nomes dos seus arquivos para garantir que estão corretos.

Como essas atribuições não são verificadas por um programa, fique à vontade para organizar seus arquivos como desejar, desde que entregue os arquivos obrigatórios e cumpra os requisitos.

arquivo.bfe:VAA0DAYFf07ym3ROeASsmsgnY0o0sDMJev7zFHhwQ
S8mvM8V5xQQpLc6cDCFXDWTiFzZ2H9skYkiJ/DpQtnM/uZ0