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 básicos de jogabilidade.

Versão: 2.3

Conteúdo

Capítulo I

Prefácio

Ser um desenvolvedor é ótimo para criar seu próprio jogo.

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

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

De qualquer forma, tente respeitar o trabalho dos outros.

Capítulo II

Objetivos

É hora de você criar um projeto básico de computação gráfica!

solong 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.

Os outros objetivos são semelhantes a todos os outros objetivos deste primeiro ano: ser rigoroso, subir de nível na programação C, usar algoritmos básicos, fazer alguma pesquisa de informações e assim por diante.

Capítulo III

Instruções Comuns

- Seu projeto deve ser escrito em C.
- Seu projeto deve ser redigido de acordo com a Norma. Se você tiver arquivos/funções de bônus, eles serão incluídos na verificação da norma e você receberá um 0 se houver um erro de norma dentro.
- Suas funções não devem ser encerradas inesperadamente (falha de segmentação, erro de barramento, double free, etc.) além de comportamentos indefinidos. Se isso acontecer, 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 de origem para a saída necessária com os sinalizadores -Wall, -Wextra e -Werror, use cc e seu Makefile não deve revincular.
- Seu Makefile deve conter pelo menos as regras $(NAME), all, clean, fclean e ré.
- Para ativar bônus em seu projeto, você deve incluir uma regra de bônus em seu Makefile, que adicionará todos os vários 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 bônus é feita separadamente.
- Se seu projeto permite que você use sua libft, você deve copiar suas 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, em seguida, compilar o projeto.
- Incentivamos você a criar programas de teste para o seu projeto, mesmo que este trabalho **não precise ser enviado e não seja avaliado.** Isso lhe dará a chance de testar facilmente seu trabalho e o de seus colegas. Você achará esses testes especialmente úteis durante sua defesa. De fato, durante a defesa, você é livre para usar seus testes e/ou os testes do par que está avaliando.
- Envie seu trabalho para o repositório git atribuído. Somente o trabalho no repositório git será avaliado. Se o Deepthought for designado para avaliar seu trabalho, isso será feito

depois de suas avaliações de 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, maps, textures NAME, all, clean,
Argumentos	fclean, re Um mapa no formato *.ber
Makefile	
Funções externas.	• abrir, fechar, ler, escrever, malloc, liberar, perror, strerror, sair • Todas as funções da matemática biblioteca (opção de compilador -lm, man man 3 math) • Todas as funções do MiniLibX • ft_printf e qualquer equivalente VOCÊ codificou
Liberação autorizada	Sim
Descrição	Você deve criar um jogo 2D básico no qual um golfinho escapa da Terra depois de 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 as seguintes regras:

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

IV.1 Jogo

- O objetivo do jogador é coletar todos os presentes colecionáveis no mapa e 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 nessas **4 direções:** para cima, para baixo, para a esquerda, para a direita.
- O jogador não deve ser capaz de se mover nas paredes.
- A cada movimento, o **número atual de movimentos** deve ser exibido no shell.
- Você deve usar uma **visão 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 quiser querer.

Se preferir, você pode usar o ZQSD ou as setas 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 suave (mudando para outra vitória diminuir, minimizar 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, colecionáveis** e **livre 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 válido simples:

```
11111111111111
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. Se não for, o programa deve retornar um erro.

- Você deve verificar se há 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:

```
1111111111111111111111111111111111111
1E0000000000000C00000C000000000001
1010010100100000101001000000010101
1010010010101010001001000000010101
1P0000000C00C00000000000000000000C1
1111111111111111111111111111111111111
```

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

Capítulo V

Parte bônus

Normalmente, você seria encorajado a desenvolver seus próprios recursos extras originais. No entanto, haverá projetos gráficos muito mais interessantes posteriormente. Eles estão à tua espera!!
Não perca muito tempo nesta tarefa!

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

Você receberá pontos extras se:

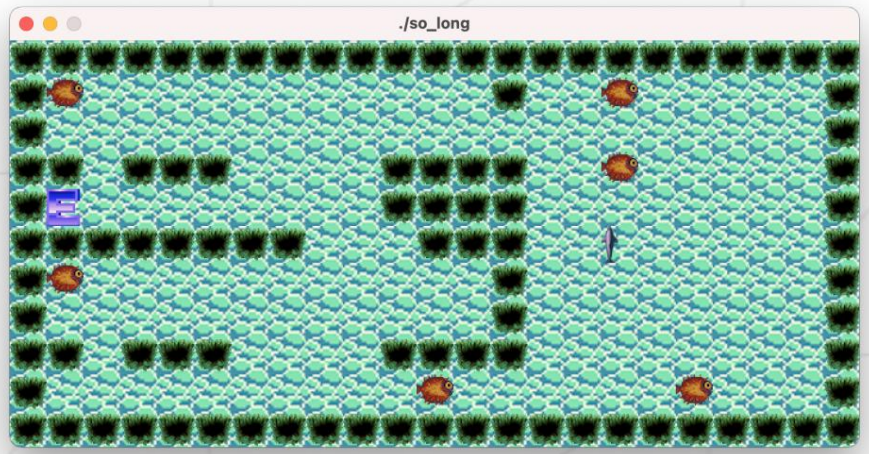
- Fazer o jogador perder ao tocar em uma patrulha inimiga.
- Adicione alguma animação sprite.
- Exibir a contagem de movimento diretamente na tela em vez de escrevê-la no shell.

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

A parte bônus só será avaliada se a parte obrigatória for PERFEITA. Perfeito significa que a peça obrigatória foi executada integralmente e funciona sem avarias. Se você não passou em TODOS os requisitos obrigatórios, sua parte de bônus não será avaliada.

Capítulo VI

Exemplos



so_long exemplos mostrando péssimo gosto em design gráfico
(quase valendo alguns pontos de bônus)!

Capítulo VII

Submissão e avaliação por pares

Entregue sua atribuição em seu repositório Git como de costume. Apenas o trabalho dentro do seu repositório será avaliado durante a defesa. Não hesite em verificar novamente os nomes de seus arquivos para garantir que estejam corretos.

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