Documentação

Jogo das canoas

Índice:

Bibliotecas (Arquivos .h)

LIST:

RIVER:

Arquivos .c

Bibliotecas allegro.h

Cria as funções que são responsáveis pela imagem gerada em interface gráfica como as que desenham retângulos e triângulos de água e de terra manipula a imagem que representa a canoa. Além de chamar a biblioteca allegro, abrir e fechar a janela e criar os eventos do teclado.

Bibliotecas getopt.h

Cria a função char getopt(int argc, char **argv, char *optstring); que controla o menu de opções. Através dele, recebemos as opções que o usuário entra como parâmetro e e a utilizamos. Também cria variáveis auxiliares à respectiva função.

Bibliotecas player.h

Controla a quantidade de vidas do jogador.

Bibliotecas statistical.h

Cria as funções estatísticas void stat_set_seed(int seed);, float stat_gen_uniform_float(float init, float end); e int stat_gen_uniform_int(int init, int end); no módulo statistical.c que auxiliam na geração aleatória do rio e suas ilhas.

Bibliotecas strip.h

Responsável pelas funções que manipulam as faixas de terra do terreno com as funções void tstrip_seed(int);, void tstrip_free(TStrip);, void tstrip_island(float prob, int freq); e TStrip tstrip_generate(int size, int zone, float normalization, TStrip base, TStrip nova); que estão localizadas no módulo strip.c.

Bibliotecas terrain.h

Responsável pelas funções que controlam a configuração do terreno como a velocidade e tipo, no caso terra ou água.

Bibliotecas test.h

É responsável pelas funções que realizam os testes com velocidades, margens e fluxos de acordo com o número de frames/iterações que o usuário escolher. Serão passados os valores de acordo com o n-ésimo frame.

Bibliotecas utils.h

Responsável pela função de alocação de memória de uma forma que o programa será finalizado caso haja erro de alocação.

Bibliotecas LIST list.h

Cria as funções de manipulação de listas ligadas circulares com cabeça duplamente encadeadas. Tais como remoção, inicialização, inserção, seleção, entre outros.

Bibliotecas LIST list-internal.h

É responsável pela configuração da struct que manipulamos em list.h

Bibliotecas LIST list-item.h

Define um vetor LItem(*LItem) como um terrain.

Bibliotecas RIVER river.h

Controla as funções que irão manipular a configuração do rio.

Bibliotecas RIVER river-internal.h

É responsável pela struct de configuração do rio, controlando o fluxo, altura, largura, zona, probabilidade, frequência e impressão da faixa de terra.

Arquivos .c allegro.c

Manipula as funções que são responsáveis pela imagem gerada na tela e pela criação e finalização da janela e de eventos através das funções:

void gui_init(void) Inicializa a biblioteca allegro e atribui null a fila de eventos.

void gui_window_clear(void) Pinta a janela inteira com a cor preta (0, 0, 0).

void gui_window_delay(float t) Segura a execução por t segundos.

void gui_window_update(void) Carrega a janela, atualiza as configurações que foram feitas e chama a função que desenha os corações na tela conforme o número de vidas restantes do usuário.

void gui_window_create(int length, int height) Cria uma janela de altura height e altura length.

int gui_window_destroy(void) Fecha a janela e finaliza o programa quando o usuário clica em 'X'.

void gui_river_water(float x1, float y1, float x2, float y2) Desenha retangulos na cor da água nas coordenas (x1, y1, x2, y2).

void gui_river_land(float x1, float y1) Desenha retangulos na cor de terra nas coordenadas (x1, y1, x2, y2).

void gui_river_smooth_water(float x1, float y1, float x2, float y2, float x3, float y3) Desenha triangulos na cor de água que são resultado da margem polarizada.

void gui_river_smooth_land(float x1, float y1, float x2, float y2, float x3, float y3) Desenha triangulos na cor de terra que são resultado da margem polarizada.

void gui_river_heart(int lifes) Desenha n corações no canto superior direito da tela, sendo n o número de vidas restantes do usuário.

void gui_boat_draw(int *x, int *y, int proportion) Imprime a imagem do barco na tela e ajusta o seu angulo de acordo com o seu movimento (se ele foi virado para a esquerda, será inclinado para a esquerda) e de acordo com a posição em que o usuário atribuiu a ele durante o jogo.

int gui_keyboard_init(void) Inicializa teclado e o associa a uma fila de eventos.

int gui_event_get(void) Recebe os eventos que estão na fila de eventos para poder manipular o jogo. Como por exemplo os eventos fecha janela, pressionar a tecla "seta"que controlará os movimentos da canoa e soltar a tecla que fará com que o jogador possa segurar uma tecla e a canoa ande até a tecla ser soltada.

Arquivos .c getopt.c

Este módulo possui a função getopt recebe todos os parâmetros passados para o main pelo usuário, a quantidade de palavras recebidas(arg c) e também, um vetor de chars(optstring) que possui todas as opções de letras possíveis, sendo que este é apontado por um inteiro que controlará os valores já verificados... Como o L (de Largura), o H(de altura), entre outros. Essa função compara, a cada iteração, os valores recebidos pelo main e os valores de optstring... Caso seja encontrado um valor semelhante e ele tenha o -"antes, esse char é devolvido pela função, ele é movido para o início pela função move_to_init(char **argv, int initial_pos); (função que se localiza neste mesmo módulo) e o inteiro que aponta para aoptstring passa a apontar para o primeiro valor ainda não verificado.

Arquivos .c list.c

Este módulo irá controlar as listas ligadas circulares com cabeça duplamente encadeadas de uma forma generalizada, permitindo que tipos diferentes de estruturas possam utilizar esse meio de manipulação de lista. Nele estão localizadas várias funções como...

List list_init(int N) inicializa a lista, criando uma lista e uma cabeça para ela... E como é uma lista circular duplamente encadeada, atribuímos a cabeça como o próximo e o anterior dela mesma.

void list_free(List list) libera a lista inteira, inclusive a cabeça. Essa função é de suma importância, pois desocupa os espaços que serão mais utilizados na memória.

int list_empty(List list) retorna 1 caso a lista esteja vazia e 0 caso possua algum elemento. Observe que uma lista vazia é aquela que possui apenas a cabeça.

LItem list_remove(List list, Link node) irá remover uma determinada célula da lista e irá retornar o valor(item) que ela possuia. Quando essa célula é removido, a célula posterior passará a ser adjacente à anterior e a célula removida passará a ser excluída da memória usando free.

void list_insert(List list, LItem item) adiciona uma célula à lista de forma que esta seja adicionada após a cabeça. Ou seja, cabeça-¿próximo = cálula.

Link list_head(List list) devolve o endereço da cabeça da lista, passada como parâmetro.

Link list_next(Link node) devolve o endereço da célula seguinte à célula que foi passada como parâmetro.

Link list_prev(Link node) devolve o endereço da célula anterior à que foi passada como parâmetro.

LItem list_item(Link node) retorna o item que está contido na célula que foi passada como parâmetro.

void list_select(List list, int direction, void (*visit) (LItem)) Percorrerá a lista de frente para trás caso direction seja igual a HEAD, caso contrário, percorrerá a lista de trás para frente.

Link list_new_node(LItem item) cria uma nova célula preenchida por um item passado como parâmetro. Essa nova célula criada não se liga a nenhuma outra, é uma célula inicialmente isolada.

Arquivos .c river.c

Esse módulo irá controlar as características do rio, como o fluxo, a velocidade de cada ponto, entre outros com as seguintes funções:

void river_config_flux(float flux) recebe como parâmetro um valor e o passa para o atribui ao fluxo do rio.

void river_config_island (float prob_island, int freq_island) atribui a probabilidade de surgimento de uma ilha e a distância mínima que ela deve estar de uma outra ilha ao rio, a partir dos parâmetros recebidos.

void river_config_size(int length, int height) atribui altura e largura ao rio a partir dos parâmetros recebidos.

void river_config_margins(int zone) atribui uma zona de conforto ao rio que é a mínima distância que uma margem pode ter da outra, impedindo que o rio fique inavegável.

void river_animation_generate(int seed) é responsável por gerar o rio, ela irá chamar outras funções que atribuirão estatisticamente valores para iniciar o rio e que preencherão o rio com h linhas que contêm informações de ser água ou terra e velocidade. Sendo h a altura do rio atribuída pelo usuário.

void river_animation_iterate() é responsável pela imagem que o usuário verá na tela, pois pegará o valor da altura e juntamente com a função void river_animation_generate(int seed), gerará um frame do rio, juntando as h faixas de terreno do repectivo frame.

void river_animation_init() Inicializa o rio, atribuindo padrões como vida = 10 e canoa centralizada e criação da janela.

void strip_print(TStrip strip) irá imprimir uma determinada linha do rio em interface gráfica chamando as funções de allegro.h. Nessa função será verificado se determinado pixel é terra, água ou margem para assim poder usar a biblioteca allegro através de outas funções.

void river_animation_finish() é responsável por finalizar o rio, desalocando as células utilizadas por ele.

Arquivos .c statistical.c

Esse módulo é responsável por gerar números aleatórios de acordo com determinadas distribuições que são determinadas pelas seguintes funções:

void stat_set_seed(int seed) inicializa a semente dos números randomicos.

float stat_gen_uniform_float(float min, float max) gera números aleatórios do tipo float entre min e max de acordo com a distribuição uniforme.

int stat_gen_uniform_int(int min, int max) gera números aletórios do tipo inteiros entre min e max (incluindo min e max) de acordo com a distribuição uniforme

Arquivos .c strip.c

Manipula as faixas de terra, que são as linhas da imagem, com as seguintes funções:

void tstrip_seed(int) irá inicializar a seed através de uma função no módulo estístico.

void tstrip_island(float prob, int freq) irá atribuir valores recebidos por parâmetros, à variáveis internas de frequência e probabilidade de surgimento de ilha.

void tstrip_free(TStrip strip) irá remover a faixa de terra utilizando a função de list.c, desse modo, esse espaço será desalocado da memória.

TStrip tstrip_generate(int size, int zone, float normalization, TStrip base) gera uma faixa de terra. Se ela for a primeira linha, esta será gerada unicamente com funções estísticas, porém, se ela não for a primeira faixa, irá gerar a linha estatisticamente de acordo com a linha anterior... Também atribuirá velocidades aos pontos de acordo com funções estatísticas, sempre tomando como base a faixa de terra anterior.

Arquivos .c test.c

void analyse_program(int seed, int iterations) fará com que o programa gere um rio com n iterações para que na n-ésima iteração, possa-se retirar os valores para teste.

void analyse_river(int seed) é responsável por imprimir na tela os resultados do teste de acordo com o n-ésimo frame. Imprime características como fluxo do rio, altura da tela, largura da tela, zona de conforto do rio, entre outras características que são passados pelo usuário via parâmetro.

static void analyse_lines(TStrip strip) calcula e faz um relatório de determinada linha do rio, mostrando valores que não são passados pelo usuário, mas são gerados probabilisticamente, como a quantidade de água e terra, fluxo, e posição das margens em uma linha.

Arquivos .c utils.c

Aloca memória com tamanho size_t de forma que, caso haja um erro de alocação o programa é finalizado.