Marcos Samuel Winkel Landi

Henry Bernardo Kochenborger De Avila

Trabalho Final

Inf01202 - Algoritmos e Programação – 2018/2 - UFRGS

Introdução

Este documento descreve o nosso trabalho final da cadeira de Algoritmos e Programação da UFRGS e seu processo de criação. Foi feito um clone do jogo Megamania, originalmente lançado para Atari 2600, utilizando a linguagem C e uma biblioteca gráfica.

A Biblioteca Gráfica

Nós usamos a biblioteca CSFML, um binding de SFML (Simple and Fast Multimedia Library) para C, para fazer os gráficos do jogo. Ela funciona assim: Tudo é desenhado em uma janela (do tipo sfRenderWindow), através de funções da janela para desenhar cada elemento gráfico específico (sfRenderWindow_drawSprite(...), sfRenderWindow_drawRectangleShape(...)).

É configurado o que precisa ser configurado no começo do programa, e as instruções a serem executadas a cada quadro ficam dentro de um loop while, que acaba somente quando a janela é fechada. Dentro deste while, é primeiro testado se o evento de fechar a janela ocorreu, e depois a lógica de cada quadro é descrita. Depois da parte lógica, para desenhar na janela, usa-se a função sfRenderWindow_clear(...) para limpar tudo o que havia anteriormente na tela, após, as funções de desenhar, e no final a função sfRenderWindow_display(...), que exibe tudo na tela.

Todas as imagens do jogo são armazenadas em “sfSprite”s para serem desenhadas na janela. Sprites são elementos que guardam uma textura, com propriedades únicas para serem desenhadas na tela, como a posição, a escala do tamanho, a rotação, a cor... A textura, por sua vez, é a que guarda a imagem em si, que é carregada na textura a partir de um arquivo de imagem. Todo este processo de criação de Sprites foi automatizada através de uma função auxiliar, sfSprite_CreateFromFile(...).

A Lógica do Jogo

Assim que a função main() é executada, são carregadas as sprites, os mapas, e criada a janela. Após, é chamada a função Layout_GameMenu(...), que exibe o menu. No menu, há 3 botões, que levam a outras funções chamadas “layouts”, cada uma responsável pela exibição de um layout gráfico: layoutStage, o layout dos níveis,

layout_Highscores, o layout de ranking e layout_Credits, o layout para mostrar os créditos. Em cada layout há um loop de desenho, e uma condição para fechar este loop. Somente o loop do layout_GameMenu só termina quando a janela é fechada, e em todos os loops é checado se o botão de fechar a janela foi clicada, para fechá-la.

Mapas

Quando o layout de nível é executado, antes devem ser carregados os inimigos, de acordo com o arquivo do nível. Estes arquivos devem estar no seguinte formato: “map_” + numero + “.txt”, onde o número é o número do nível a ser executado. Os níveis vão sendo carregados, a partir do “map_1.txt” até não ter mais números na sequência, ficando mais fácil de adicionar outros níveis.

De acordo com as definições da estrutura dos arquivos de mapas do enunciado do trabalho, podem existir 2 modos de movimento: R (direita) e L (esquerda), que diz para qual lado os inimigos irão se mover. Nós adicionamos 2 modos: B (ambos) e S (senoidal), ambos faz com que seja alternado entre R e L, e senoidal faz um movimento do tipo senoidal verticalmente.

Estruturas

TYPE_BUTTON – tipo que descreve um botão.

- sfText* text – texto do botão;
- sfRectangleShape* base – retângulo clicável onde fica o texto.

TYPE_MENU – tipo que guarda os elementos de menu.

- sfFont* font – fonte de texto;
- sfText* megamaniaLogo – texto, a “logo” do nosso jogo;
- TYPE_BUTTON buttons[3] – array dos 3 botões do menu.

TYPE_FIRE – tipo que descreve todos os tiros dos inimigos.

- int isOnScreen – se está ativo ou não;
- float posX – posição X na tela;
- float posY – posição Y na tela.

TYPE_ENEMIES – Estrutura que descreve um inimigo. Possui as posições X e Y, a posição inicial na tela, uma cor, um TYPE_FIRE e uma flag que diz se está vivo ou não.

- float posX – posição X na tela;
- float posY – posição Y na tela;
- sfVector2f initialPos – posição inicial do inimigo na tela;
- int color – cor do inimigo;

- int isAlive – está vivo ou não;
- TYPE_FIRE fire – tiro do inimigo.

TYPE_PLAYERSHIP – Estrutura que descreve o jogador.

- sfSprite* shipSprite - Sprite para desenhar na tela;
- float posX – posição X na tela;
- float posY – posição Y na tela.

TYPE_GAMEOBJECTS – Estrutura que guarda todas as Sprites, a nave do jogador e os inimigos do jogo.

- Múltiplas sfSprite* - Sprites do jogo;
- TYPE_PLAYERSHIP ship – nave do jogador;
- TYPE_ENEMIES enemies[] – array de inimigos.

TYPE_LEVEL – Estrutura que guarda varias variáveis uteis, relacionadas à lógica de cada nível.

- int levelSpeed – velocidade (dificuldade) do nível;
- char Direction – direção para onde os inimigos andam (‘r’ – direita, ‘l’ – esquerda, ‘b’ – ambos ou ‘s’ - senoidal);
- int numberEnemies – número de inimigos no nível;
- int paused – flag, se está ou não pausado;
- float lastShot – à quantos segundos um inimigo atirou (para controlar a frequência).

Arquivos

- main.c – Onde fica o main(), a função inicial que é executada quando o jogo é iniciado
- enemies.c e .h – Arquivos que descrevem as funções relacionadas aos inimigos, com o prefixo “Enemies_”.
- layout.c e .h – Descrevem as “funções layout” e outras funções relacionadas aos layouts, todas com o prefixo “Layout_”.
- score.c e .h – Descrevem funções relacionadas ao score de cada jogo. São precedidas pelo prefixo “Score_”.
- utility.c e .h – Descrevem funções úteis variadas. São precedidas pelo prefixo “Utility_”

- sprite.c e .h – Descrevem funções relacionadas aos Sprites, tanto para lidar com sprites como funções para desenhar elementos gráficos específicos.
- global.h e .c – Onde são declaradas as estruturas e variáveis globais usadas no programa. Também é onde são definidas constantes e incluídas algumas bibliotecas padrão do C , utilizadas nos outros arquivos.

Instruções

Para executar o jogo a partir da pasta do projeto, basta entrar no diretório /bin/Debug, onde consta o executável do jogo, “megamania.exe”.

Para abrir o projeto no Code::Blocks, deve-se abrir o projeto megamania.cbp, clicar em “rebuild”, e então rodar o programa, caso desejado.

Se for desejado alterar os mapas, deve-se editar os arquivos .txt dos mapas. São eles: map_1.txt, map_2.txt, map_3.txt e assim por diante. Se for desejado adicionar mais um, nomeie-o com o próximo número na sequência, e ele será jogável, assim que você vencer o anterior ele será executado.

Para instalar a biblioteca CSFML manualmente, caso haja problemas para utiliza-la, faça o seguinte:

1. Baixe a biblioteca do site https://www.sfml-dev.org/download/csfml/ (a última versão, Windows 32 bit)
2. No Code::Blocks, na aba Project, clique em “Build Options”
3. À esquerda, clique no projeto megamania, e na aba “Linker settings”
4. Agora, adicione os seguintes arquivos na caixa “Link libraries”, nesta ordem: (eles estão na pasta /lib/gcc/ do diretório do csfml)
    a. libcsfml-graphics.a
    b. libcsfml-window.a
    c. libcsfml-system.a
5. Na aba “Search directories”, na seção “Compiler”, adicione a pasta /include/ do diretório do CSFML.
6. Na seção “Linker”, adicione a pasta /lib/gcc/ do diretório do CSFML.