TP Prático I

Douglas Rodrigues de Almeida

Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG) – Belo O Horizonte, MG – Brasil

douglasralmeida@live.com

1. Introdução

O TP consiste na criação de um jogo eletrônico em 2D para Microsoft Windows construído na linguagem C com o auxílio da biblioteca Allegro.

O jogo possui um personagem, visto de cima, que deve atirar em todos os inimigos para concluir a fase.

2. Requisitos para compilação

Para compilar o aplicativo, é necessário as seguintes ferramentas:

- GCC 5.2 ou superior.
- MingW 4.0.3 ou superior.
- InnoSetup 5.5.6 ou superior, para geração do instalador.
- Biblioteca Allegro 5.1.9 ou superior.
- Biblioteca Libvorbis.
- Biblioteca OpenAL.
- Biblioteca FreeType.
- Biblioteca LibJPEG Turbo.
- Biblioteca LibPNG.

2. Requisitos para a execução

Para executar o aplicativo compilado, é necessário:

- Sistema Operacional Windosw 7 64 bits ou superior.
- GDCL Mpeg-4 Demultiplexor, já incluso.

Uma versão do instalador para teste do jogo esta disponivel para download no endereco:

https://www.dropbox.com/s/xl0q19258olyhsx/tp_douglas_jogo1.exe?dl=0

3. Organização dos arquivos e pastas

O jogo está organizado em pastas conforme descrito a seguir:

\bin: Armazena o jogo compilado e todos os arquivos necessários para execução do jogo.

\doc: Contém o relatório sobre a produção do jogo.

\include: Contém os cabeçalhos necessários para a compilação.

\obj: Contém os objetos dos arquivos compilados.

\res: Contém recursos gráficos para compilação e geração do instalador.

\setup: Script para geração do instalador usado a ferramenta InnoSetup.

\source: Código-fonte do jogo.

\test: Implementação de testes unitários (incompleto).

\tools: Ferramentas para manipulação do código-fonte.

4. Sobre o Makefile

Na raiz do projeto do jogo há um arquivo makefile, utilizado para manipulação do código-fonte. Os seguintes comandos são aceitos pelo makefile.

MAKE – Limpa os objetos compilados, o executável e faz uma compilação completa do jogo.

MAKE CLEAN – Exclui os objetos compilados e o executável.

MAKE DEBUG – Execute o jogo, após compilado, no modo de depuração sob a supervisão do GDB.

MAKE HELP – Exibe um menu de comandos.

MAKE INSTALL – Registra a biblioteca mp4demux.dll, necessária para executação do jogo. Execute este comando com privilégios de administrador.

MAKE README – Exibe o arquivo leiame.

MAKE RUN – Executa o jogo, após compilado. Antes de executar o jogo pela primeira vez, execute o comando MAKE INSTALL.

MAKE SYSINFO – Exibe informações de versão das ferramentas de desenvolvimento instalados no seu computador.

MAKE TEST – Compila os arquivos para teste unitário.

5. Estrutra do jogo

A estrutura do jogo está na pasta \bin\recursos\. Os dados jogo estão no arquivo *jogo.dados*, que fornece informações ao aplicativo executável. A pasta contém as seguintes subpastas:

\cursores: Fornece imagens para cursores do mouse.

\fases: Fornece os arquivos das fases do jogo.

\fontes: Fornece as fontes para geração de textos na tela do jogo.

\graficos: Fornece os gráficos necessários para geração de objetos do jogo, como o mundo, personagens, projétis.

\imagens: Fornece as imagens para exibição durante o jogo.

\mapas: Fornece os arquivos dos mapas do jogo.

\melodias: Fornece as músicas do ambiente do jogo.

\sons: Fornece os arquivos sonoros das interações do jogo, como passos, tiros de projétil e interações do menu.

\videos: Fornece arquivos de vídeos usados durante o jogo.

Os arquivos que fornecem dados ao jogo são divididos por seções, indicados pelo símbolo dois-pontos (:). Cada seção possui chaves que fornecem as informações para o jogo Cada arquivo finaliza com a seção FIM.

O analisador de arquivos diferencia letras maiúsculas de letras minusculas.

5.1. O arquivo jogo.data

Situado na pasta \recursos\, este é o principal arquivo do jogo, cuja função é fornecer dados para alocação de memória e os dados das fases do jogo. Descrição do arquivo:

Seção ARQUIVOJOGO:

Chave Versao: Contem a versao do arquivo. Deve ser marcado como 1.

Seção DEFINICOES:

Chave CursorMouse: Contém o arquivo do cursor do mouse exibido no jogo.

Chave MaxParticulas: Contem o tamanho do vetor do detector de partículas do jogo. Se o jogo tiver um número de objetos interativos maior do que suportado pelo vetor, ele exibirá uma mensagem de erro.

Chave MaxProjetis: Contem o tamanho do vetor de manipulação de projeteis do jogo. É a quantidade máxima de projéteis simultâneos que o jogo suporta.

Chave QuantidadeFases: O número de fases do jogo.

Seção FASES: Contém, um por linha, o nome dos arquivos de todas fases do jogo.

*5.2 O arquivo *.fase*

Situado na pasta \recursos\fases\, possui os dados de cada fase do jogo. Descrição do arquivo:

Seção ARQUIVOFASE

Chave Versao: Contem a versao do arquivo. Deve ser marcado como 1.

Seção DEFINICOES

Prologo: Imagem a ser exibida antes do início da fase.

ArquivoMapa: Arquivo com dados do mapa da fase.

MusicaAmbiente: Áudio que será reproduzido durante a execução da fase.

PosJogadorInicial: Localização, no mapa, da posição inicial do jogador

PosJogadorFinal: Localização, no mapa, da posição final do jogador. O jogador deve se dirigir a essa posição após cumprir todos os pré-requisitos para passar de fase.

Seção JOGADOR

ArquivoPele: Bitmap do jogador.

Seção CHEFAO

ArquivoPele: Bitmap do chefão da fase.

PosChefao: Posição inicial, no mapa, da posição inicial do chefão da fase.

Seção INIMIGOS

QuantInimigos: Quantidade de inimigos na fase.

ArquivoPele: Bitmap dos inimigos na fase.

Seção POSINIMIGOS

PosInimigo: Posição inicial, no mapa, de cada inimigo na fase. Essa chave deve repetir para cada inimigo da fase.

*5.2 O arquivo *.mapa*

Situado na pasta \recursos\mapa\, possui os dados do mapa de uma fase do jogo. Descrição do arquivo:

Seção ARQUIVOMAPA

Chave Versao: Contem a versao do arquivo. Deve ser marcado como 1.

Seção DEFINICOES

AlturaMapa: Quantidade de quadrantes na linha vertical do mapa.

AudiosPassadas: Quantidade de aúdio das passadas dos personagens a serem alocados.

CursorArquivo: Cursor do mouse utilizado no mapa.

CursorFoco: Qual ponto do arquivo do cursor será usado como foco do mouse.

LarguraMapa: Quantidade de quadrantes na linha horizontal do mapa.

NumeroMascaras: Quantidade de máscaras para quadrantes do mapa a serem alocados pelo jogo.

QuadrantesInformados: Quantidade de quadrantes diferentes do quadrante padrão alocados pelo jogo.

QuadranteFinal: Qual, dos quadrantes alocados, simbolizará o local onde o jogador deverá ir para finalizar a fase.

Seção MASCARAS: Bitmaps, um por linha, que serão alocados para mascarar os quadrantes do mapa.

Seção PASSADAS: Áudios, um por linha, que serão reproduzidos durante uma caminhada de um personagem.

Seção BORDAS

Chave BordaS: Qual das mascarás alocadas, será utilizada para mascarar a borda superior do mapa.

Chave BordaI: Qual das mascarás alocadas, será utilizada para mascarar a borda inferior do mapa.

Chave BordaE: Qual das mascarás alocadas, será utilizada para mascarar a borda esquerda do mapa.

Chave BordaD: Qual das mascarás alocadas, será utilizada para mascarar a borda direita do mapa.

Chave BordaSE: Qual das mascarás alocadas, será utilizada para mascarar o canto superior esquerdo do mapa.

Chave BordaSD: Qual das mascarás alocadas, será utilizada para mascarar o canto superior direto do mapa.

Chave BordaID: Qual das mascarás alocadas, será utilizada para mascarar o canto inferior direito do mapa.

Chave BordaIE: Qual das mascarás alocadas, será utilizada para mascarar o canto inferior esquerdo do mapa.

Seção QUADRANTEPADRAO: Dados do quadrante padrão do mapa. Quadrante padrão é aquele quadrante que não foi especificado pelo arquivo de mapa. Não deve conter a chave Posicao.

Seção QUADRANTE: Dados de um quadrante específico.

Posicao: Posição com linha e coluna do quadrante.

Colisao: Zero para permitir passagem de personagens. Um para bloquear a passagem.

Mascara: ID do bitmap de máscara alocado.

TemSangue: Não implementado.

Passada: ID do áudio que será reproduzido quando um personagem caminhar pelo quadrante.

6. Estrutura do código-fonte

O código-fonte do jogo está localizado na pasta \source\ e seus respectivos cabeçalhos estão localizados na pasta \include\.

O jogo é todo modularizado, ou seja, cada arquivo trabalha com um recurso específico do jogo.

- arquivo.c: Responsável pela manipulação de arquivos.

- audio.c: Responsável pelo áudio do jogo.

-colisoes.c: Responsável por tratar as colisões do jogo.

-fase.c: Responsável por tratar e manipular as fases do jogo.

-fisica.c: Responsável pela física do jogo.

-fonte.c: Responsável por tratar as fontes utilizadas para exibir textos na tela.

-funcoes.c: Possui funções básicas do jogo e de recursos sistema.

-graficos.c: Responsável por tratar a parte gráfica do jogo.

-janela.c: Responsável por tratar uma janela do jogo.

-janelas.c: Responsável por tratar cada janela específica do jogo.

-jogo.c: Responsável por tratar o ambiente do jogo.

-mapa.c: Responsável por tratar o mapa do jogo.

-matematica.c: Responsável pelos cálculos matemáticos.

-menu.c: Responsável por tratar os menus do jogo.

-mouse.c: Responsável por tratar as ações do mouse.

-painelstatus.c: Responsável por tratar o Painel de Status do Jogador.

-personagem.c: Responsável por tratar os personagens do jogo.

-principal.c: Contém a função WinMain e é a porta de entrada para o jogo.

-projetil.c: Responsável por tratar os projeteis do jogo.

-teclado.c: Responsável por tratar as ações do teclado do jogo.

-vetor.c: Responsável por tratar vetores e matrizes alocados durante o jogo.

-video.c: Responsável pela execução de vídeos durante o jogo.

-recursos.rc: Contém recursos gráficos, como o ícone do jogo.h.

-tipos.h: Contém tipos genéricos usados pelo jogo.

7. Funcionamento básico do jogo

O personagem é uma estrutura que caminha pela tela quando o jogador pressiona as teclas de direção. A velocidade do movimento é um vetor, com norma que indica a taxa de variação da posição, com ângulo que indica a direção do movimento e o sentido. O ângulo é calculando como sendo o menor ângulo entre a trajetória e uma reta paralela ao eixo horizontal. O mouse é utilizado para alterá-lo.

Para verificar se existe colisão com outro objeto, o jogo utiliza a técnica de caixas delimitadoras. Cada objeto interativo recebe uma máscara, um paralelepípedo que delimita este objeto. Ao efetuar um movimento é verificado se este paralelepípedo sobrepõe algum outro paralelepípedo do jogo. Caso exista sobreposição, o movimento é cancelado.

Os projéteis são representados por segmentos de retas. Após um movimento de projétil, é verificado se o segmento de reta cujo início é o ponto onde foi efetuado o disparo e o final é a posição atual do projetis intercede algum dos segmentos de reta formados pela caixas delimitadoras dos objetos interativos.

7.1 Os buffers

Para melhorar a eficiência do jogo, antes de seu início, todos os gráficos, imagens, fontes e áudios utilizados são registrados na memória. Isso evita que, a cada interação, seja efetuada uma leitura no disco rígido para leitura dos recursos o que ocasiona grande perda de desempenho do jogo.

7.2 Suporte a vídeo

Para executar vídeos, o jogo utiliza a API DircectShow do Microsoft Windows. O DirectShow é uma arquitetura para execução de mídias com suporte a tecnologias recentes para reprodução de vídeos de alta qualidade. Para obter maiores informações sobre o DirectShow, consulte sua página no MSDN.

O DirectShow é baseado em tecnologia COM. O arquivo video.c provê acesso básico aos objetos COM do gerenciador do grafo de filtros para a reprodução do vídeo. Como o Windows não possui filtros para manipulação de arquivos de vídeo MPEG-4, foi incluida a biblioteca mp4demux.dll para a execução dessa tarefa. Ela deve ser registrada no sistema através do comando regsvr32. Este comando só é executado como sucesso caso o usuário possua privilégios de administrador.

7.3 Estrutura TApplication

O jogo conta com uma estrutura global chamada Tapplications. Ele é responsável por auxiliar o tratamento de recursos durante toda a execução do aplicativo. Declarada no arquivo *funcoes.h*, sua estrutura é a seguinte:

int ClosingAll: Valor booleano que indica que o aplicativo deve ser fechado.

TConfig Confi: Estrutra que guarda as configurações do jogo. Se deve haver reprodução de áudio, por exemplo.

int DebugMode: Valor booleano que indica que o aplicativo está no modo de depuração.

HWND MainWindowHandle: Identificador da janela do jogo.

WCHAR FilePath[MAX_PATH]: String que guarda o caminho do jogo.

ALLEGRO_DISPLAY* Screen: Guarda os dados da tela do jogo para o Allegro.

int ScreenWidth: Guarda a largura da tela do jogo.

int ScreenHeight: Guarda a altura da tela do jogo.

7.4 Diretivas de compilação

Alguns recursos do jogo somente estão disponíveis conforme as diretivas usadas na compilação.

-D_UNICODE -DUNICODE: Ativa o suporte Unicode. Nas aplicações Unicode os tipos de dados char e char* são preteridos pelos tipos WCHAR e *WCHAR. A API do Directshow está disponível apenas para aplicações Unicode.

-D_WIN32_WINNT=0x0600: O aplicativo requer Windows Vista ou superior, pois utilizará recursos que estão disponíveis apenas nesses sistemas operacionais.

-D_WIN32_IE=0x0800: O aplicativo requer o Intenet Explorer 8.0 ou superior. Algumas APIs do Windows somente estão disponíveis com esta diretiva, como por exemplo, os estilos visuais das versões modernas do Windows.