Introdução a Computação Gráfica
Projeto final: aMaze Story

Luiz Fernando Gomes de Oliveira
Gustavo Jaruga Cruz
Guilherme Fay Vergara

Resumo

Apresentação do aMaze Story. Como foram tomadas as decisões e o que ele pode oferecer. Uma descrição breve sobre seus objetos e compilação.

1 Introdução

Este programa , aMaze Story, trás não apenas as lições ensinadas em sala de aula, mas também alguns conhecimentos adquiridos no decorrer do curso de engenharia que serão compartilhados neste documento.

1.1 Objetivos

No inicio do projeto, tínhamos os seguintes desafios:

- Criar um programa que faça de uso das ferramentas do OpenGL.
- Aperfeiçoar o conhecimento da linguagem *C* para viabilizar a construção de um programa com grande volume de dados de forma pratica e passível de modulação.

Devido ao OpenGL ser uma ferramenta bastante conhecida, é extremamente fácil encontrar na internet exemplos e modelos utilizando a ferramenta, porém com o decorrer do projeto, o grupo tratou de incluir alguns novos itens como desafios para o projeto, a fim de melhorar a qualidade do produto final. Estes foram os pontos incluídos:

- **Uso da linguagem C++**, no intuito de aproveitar o conceito de orientação de objetos para expandir o projeto para um jogo mais próximo de algo com formato profissional.
- **Caracterização dos módulos**, dividindo assim o programa em vários arquivos fontes menores, facilitando assim a localização de *bugs* e permitindo também a possibilidade de que varias pessoas editem o código simultaneamente.
- **Uso de ferramentas VCS/SVN**, permitindo vários backups e facilitando a construção de varias partes do código em múltiplos computadores.
- **Portabilidade**. O conhecimento de que o OpenGL não se restringia apenas a plataforma *Windows* acabou gerando o desejo de produzir um código que pudesse ser compilado em qualquer computador, seja *Windows*, *Mac* ou *Linux*.

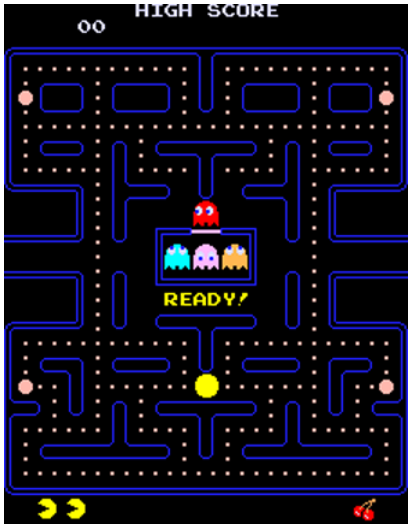

Figura 1: Pac-Man.
O clássico dos anos 80 só foi ter um score perfeito - máximo de pontos, sem falhas ou mortes - em 1999, quando *Billy Mitchell* consegui a incrível marca de $3,333,360$ pontos, após vencer os consecutivos 256 leveis do jogo.

1.2 A Historia

Toru Iwatani, criador do jogo PAC-MAN, se inspirou em uma história infantil sobre uma criatura que protegia as crianças dos monstros por comê-los. Um dos métodos de design Iwatani incluído a palavras-chave associadas com uma história para auxiliar no desenvolvimento de suas idéias. O kanji da palavra taberu ("comer"), tornou-se a premissa para o jogo. A palavra kuchi ("boca") tem um formato quadrado para seu símbolo kanji e forneceu a inspiração para o jogo da principal lenda personagem-o mais conhecido de Iwatani receber sua inspiração de uma pizza com uma fatia faltando foi, por sua própria admissão, não inteiramente correta:

> *"Bem, é uma meia verdade. Em caráter do japonês para boca (Kuchi) tem uma forma quadrada. Não é circular como a pizza, mas eu decidi arredonda-lo. Havia a tentação de fazer a forma de Pac-Man menos simples. Enquanto eu estava projetando este jogo, alguém sugeriu adicionar os olhos. Mas nós finalmente descartamos essa idéia, porque uma vez que nós adicionacemos olhos, nós gostariamos de adicionar óculos e talvez um bigode. Não teria fim. O alimento é a outra parte do conceito básico. Na minha concepção inicial, eu tinha colocado o jogador em meio a comida por toda a tela. Entao eu pensei sobre isso, percebi que o jogador não saberia exatamente o que fazer: o objetivo do jogo seria obscuro. Então, eu criei um labirinto e coloquei a comida nele. Assim, quem jogasse o jogo teria alguma estrutura ao se mover através do labirinto. Os japoneses têm uma gíria - paku-paku - eles usam para descrever o movimento da boca abrindo e fechando, enquanto se come. O nome Puck-Man veio essa palavra. "*
> - Toru Iwatani

Os monstros da história das crianças foram incluídos como quatro fantasmas que perseguem o jogador através do labirinto, proporcionando um elemento de tensão. Ataques contra o jogador foram projetados para vir em ondas (semelhante ao **Space Invaders**), em oposição a um ataque sem fim, e cada fantasma foi dada uma personalidade única e caráter. A história das crianças também incluiu o conceito de kokoro ("espírito") ou uma força de vida utilizada pela criatura que lhe permitia comer os monstros. Toru incorporou este aspecto da história de quatro pastilhas de energia comestíveis no labirinto para virar a mesa contra os fantasmas, tornando-os vulneráveis a ser comido pelo jogador.

A aparência de Puck-Man continuou a evoluir por mais de um ano. Uma grande quantidade de tempo e esforço foi feito para desenvolver os fantasmas padrões de movimentos únicos através do labirinto e aprimorando as variáveis do jogo de dificuldade, como placas foram apuradas. Símbolos de bônus (incluindo o carro-chefe Galaxian) foram adicionados à mistura, em algum momento, e os fantasmas foram finalmente nomeados: Akabei, Pinky, Aosuke, e Guzuta. Efeitos sonoros e música foram alguns dos toques finais adicionados, com o desenvolvimento se aproximando do fim, eram feitos ajustes constante do comportamento dos fantasmas.

Midway era uma distribuidora de jogos que funcionam com moedas nos EUA. Estavam sempre procurando o próximo grande sucesso do Japão para licenciar e trazer para a América. Eles optaram por tanto Puck-Man e Galaxian, modificando os armários e obras de arte para torná-los mais fáceis de fabricar, bem como proporcionar um olhar mais americano.

Puck-Man passou por grandes mudanças: o gabinete foi ligeiramente modificado, mudando a cor de branco para um amarelo brilhante para fazê-lo sobressair no arcade. O detalhado gabinete multi-colorido foi substituído com mais barato, para produzir em três cores de arte que ilustra uma representação icônica de Puck-Man (agora desenhado com olhos e pés) e um fantasma azul. Nomes ingleses foram dadas para os fantasmas (Blinky, Pinky, Inky e Clyde), e o título foi mudado da Namco para a Midway. A mudança mais significativa para Puck-Man foi o nome. A Midway temia que seria muito fácil para vândalos desagradável de espírito para mudar o P em Puck-Man para um F, criando um epíteto desagradável. Não querendo seu produto associado a esta palavra, a Midway renomeou o jogo para Pac-Man antes de liberá-lo para os arcades americanos em outubro de 1980. [1]

1.3 Entradas e Saídas

Inicialmente, o grupo precisava de uma sala complexa, com varias paredes e corredores. Assim poderíamos levantar estruturas de colisões, movimentação, iluminação e texturas. De inicio, foi utilizado um algoritmo chamado e *"Growing Tree"*, utilizado para a criação de labirintos. Inicialmente foram escolhidos dois programas base para a criação de um labirinto randômico e posteriormente a exportação do labirinto para o programa.

Com a evolução do programa e as ferramentas feitas, foi adotado um labirinto fixo, que tivesse as características dos jogos clássicos de PAC-MAN, que pode ser observado na figura 1.

O programa ainda continua fazendo leituras do teclado e do mouse para a movimentação do usuário, apresentando apenas como saída o *framebuffer* na tela do usuário.

2 Desenvolvimento

2.1 Estruturas

2.1.1 Arquitetura

No intuito de manter o jogo compatível com qualquer sistema operacional, foi decidido centralizar as inclusões de bibliotecas em um único arquivo. Para essa função foi criado o arquivo *"defines.h"*, que é responsável por reconhecer o sistema em que esta sendo compilado e incluir os devidos *headers*.

defines.h

```
#if defined (__APPLE__) || defined (MACOSX) /*MAC OS*/
```

```
    #include <GLUT/glut.h>
#else
    #ifdef _WIN32                              /* Windows */
        #define WIN32_LEAN_AND_MEAN
        #include <glee.h>
        #include <gl/gl.h>
        #include <gl/glut.h>
        #include <windows.h>
        #define sleep(x) Sleep(x)
    #else                                      /*Linux*/
        #include <cstdarg>
        #include <unistd.h>
        #include <GL/gl.h>
        #include <GL/glut.h>
        #include <GL/glu.h>
        #define Sleep(x) usleep(x<1000000?10000+300*x:x)
    #endif
#endif
```

No trecho mostrado acima, podemos ver como o programa reconhece em qual sistema esta sendo compilado e em qual endereço irá procurar pelas bibliotecas. A decisão é tomada de forma bem simples e objetiva, buscando apenas saber se as definições **MACOSX** ou **__WIN32** existem. Com estas duas definições é suficiente para dividir entre os três sistemas operacionais que o programa se propõe a dar suporte.

Porém este não é o único problema enfrentado quando se trata de um programa multiplataforma, mas também existem as dificuldades com a própria compilação.

Visando isso, foi feito um arquivo *makefile* que procede com teste semelhante ao feito no *defines.h* para verificar em que sistema se encontra e assim efetuar os links corretamente. Um trecho do *makefile* pode ser observado a seguir:

Makefile

```
UNAME = $(shell uname)
ifeq ($(UNAME),Linux) # Linux OS
    GLFLAGS = -lglut -lglui -lGLU -lGL -lalut -lopenal
    else
    ifeq ($(UNAME),Darwin) # MAC OS X
        GLFLAGS=-framework OpenGL -framework GLUT
    else #Windows
        GLFLAGS=-lopengl32 -lglu32 -lglut32 -lalut
    endif
endif
```

É valido aproveitar a oportunidade para frisar no trecho mostrado acima do *makefile* a inclusão das flags *-lalut -lopenal* para inclusão de áudio no programa.

2.1.2 Execução

2.1.2.1 **Windows**: O programa foi desenvolvido com auxilio da IDE *CodeBlocks*[1]. Assim, para gerar o executável na plataforma, basta abrir o arquivo *Projeto - Labirinto.cbp* no *CodeBlocks* e mandar compilar/construir o projeto. Na própria IDE haverá meios de executar o arquivo de saída, porém na pasta do projeto será possível localizar também o arquivo **.exe*.

2.1.2.2 **Linux**: Para se construir o programa na plataforma Linux, é necessário ter algumas bibliotecas instaladas no sistema. Dentre elas é valido destacar as do OpenGL e de áudio (*Alut* e *Openal*). Na pasta onde se encontra os arquivos fontes, é possível localizar o arquivo *makefile*. No terminal, basta executar o comando **make run** no diretorio contendo o arquivo *makefile* para compilar os arquivos e inicializar o programa corretamente. Caso alguma das bibliotecas necessárias não estejam instaladas, será observado a lista de *warnings/errors*, orientando qual biblioteca deve de ser instalada. É valido lembrar que para instalar as bibliotecas para este fim na plataforma Linux, deve-se buscar pelos nomes com o sufixo *-dev*, garantindo assim que serão instalados os arquivos necessários. A compilação será feita de forma silenciosa e se não tiver problemas, apresentará uma saída semelhante a:

Saída do terminal - Linux

```
$ make run
System: Linux OS
compiling...ok
Running...
```

2.1.2.3 **Mac OS**: Semelhante aos passos no sistema Linux, o usuário terá que executar o comando **make run** no diretorio contendo o arquivo *makefile* para compilar os arquivos e inicializar o programa corretamente. Se a compilação ocorrer corretamente, a saída deverá ser semelhante a:

Saída do terminal - Mac OS

```
$ make run
System: Darwin
compiling...ok
Running...
```

2.1.2.4 **Valgrind/Callgrind**: No intuito de melhor observar como o programa se comportava durante sua execução, utilizamos da ferramenta do *Valgrind* para visualizar a sequencia de chamadas efetuadas no programa. Para isso foi incorporado no *Makefile* a chamada para o Valgrind, onde uma nova compilação ocorre sem as chamadas de otimização e verificação de erros seguida da chamada do Valgrind para a geração de um arquivo *Callgrind.out*. Este arquivo pode ser utilizado para gerar um gráfico com as chamadas realizadas pelo programa *KCachegrind* semelhante ao gerado na imagem 2. É valido lembrar que o Valgrind roda com memoria limitada. Por este motivo, ele não permite realizar o monitoramento do programa por períodos muito extensos. O gráfico apresentado na figura 2 foi gerado disponibilizando apenas 16MB para captura de dados no Valgrind [2], como pode ser observado no trecho do manual:

> *By default, Valgrind uses the current "ulimit" value for the stack size, or 16 MB, whichever is lower. In many cases this gives a stack size in the range 8 to 16 MB, which almost never overflows for most applications.* [2]

Normalmente, faríamos da seguinte forma para usar o Valgrind:

Gerando arquivo callgrind.out

```
$ make valgrind
    g++ -g *.cpp -o prog -lglut -lglui -lGLU -lGL -lalut
-lopenal
    valgrind --tool=callgrind --dsymutil=yes
        --trace-jump=yes ./prog
```

1. Acesse http://www.codeblocks.org/ para maiores informações sobre a IDE.

Porém algumas opções foram incluídas para ter uma resposta mais apropriada. A primeira alteração trata-se da forma de compilação. Ao invés de compilar todos os arquivos diretamente, foi criado uma biblioteca dinâmica, para que o executável final carregue apenas as funções que realmente foram usadas - já que nosso código ainda carrega algumas funções para debug.

Gerando uma biblioteca dinâmica

```
    g++ -g -c button.cpp  defines.cpp   eventos.cpp
minimap.cpp  soundAL.cpp  textureloader.cpp camera.cpp
entidade.cpp  framerate.cpp  map.cpp          player.cpp
text.cpp     tile.cpp
    ar rc libAmaze.a *.o
```

Em seguida, utilizamos a biblioteca dinâmica para compilar o arquivo principal do jogo. Essa atitude permite que o binário carregue menos informações, o que implica em uma quantidade de memoria menor reservada no Valgrind.

Compilando com a biblioteca dinâmica

```
    g++ -g gamemanager.cpp -o ToGring -lglut -lglui -lGLU
-lGL -lalut  -lopenal -L./ -lAmaze
```

Assim, temos um novo binario - ToGrind - contendo apenas as funções realmente utilizadas no programa. Por fim, chamamos o Valgrind, passando algumas opções a mais:

Chamada personalizada do Valgrind

```
    valgrind --tool=callgrind --dsymutil=yes ./ToGring -q
--fullpath-after=string  --show-possibly-lost=yes
--trace-children=yes -v --main-stacksize=512MB
```

Seguem a lista de alterações passadas para o Valgrind:

1) **fullpath-after:** Essa opção é importante para programas que contenham muitos arquivos em distintos diretórios.
2) **show-possibly-lost:** Mostra possíveis blocos de memoria perdidos.
3) **trace-children:** Caso o programa produza processos filhos, eles serão acompanhados também.
4) **main-stacksize:** Altera o tamanho de memoria reservado para captura de dados.

2.1.3 Artefatos

2.1.3.1 **Arquivos**: Arquivos utilizados na construção do programa[2]:

- button.cpp
- button.h
- camera.cpp
- camera.h
- defines.cpp
- defines.h
- entidade.cpp
- entidade.h
- eventos.cpp
- eventos.h
- framerate.cpp
- framerate.h
- gamemanager.cpp
- gamemanager.h
- map.cpp
- map.h
- minimap.cpp
- minimap.h
- player.cpp
- player.h
- soundAL.cpp
- soundAL.h
- text.cpp
- text.h
- textureloader.cpp
- textureloader.h
- tile.cpp
- tile.h
- vetor3d.h
- vetor.h

2.1.3.2 **README**: O arquivo README pode ser localizado dentre os arquivos fontes, em B.2.12. Nele há algumas informações sobre como o programa foi desenvolvido e uma breve instrução de como construir o jogo a partir do código fonte.

2.1.4 Problemas Técnicos

2.1.4.1 **Inconsistencias entre sistemas operacionais:**: Ao apertar Shift para correr após já estar se movendo em windows o SO windows não envia o evento e portanto não realiza a corrida. Ao passo que no SO linux o evento é enviado e o jogador começa a corrida, como deveria.

2.1.4.2 **Frame rate:**: O sistema utiliza-se de um frame cap de 60 FPS. Porém ocorre certas divergencias devido aos sleep's do windows e do linux serem um pouco diferentes entre si.

3 Caso de Teste

Foram feitos três estudos de casos referentes ao programa em sí.

3.1 Sistema de derrota

Para o primeiro caso é estudado a situação de derrota. É a situação onde o jogador colide com um fantasma.

2. Atualizado em 10 de Setembro de 2012

Pré-condições	Ter iniciado o programa
Procedimentos	1. Usando as teclas WSAD e o mouse, andar na direção de um inimigo.
	2. Colidir com o inimigo.
Resultado Esperado	Musica de derrota é tocada. Jogador perde uma vida e retorna para a tela principal com uma mensagem de derrota.
Pós-condições	Câmera do jogador parada olhando para o muro.

3.2 Sistema de movimento

Neste segundo estudo é verificado a condição primaria do programa, ou seja, iniciar o jogo e movimentar-se pelo cenário.

Pré-condições	Ter iniciado o programa
Procedimentos	1. Apertar sobre o botão quadrado no centro para iniciar o jogo.
	2. Usar as teclas WSAD para se movimentar.
	3. Segurar o botão esquerdo do mouse e movimenta-lo para mover a direção da câmera.
Resultado Esperado	A musica é alterada. Mostra a câmera do jogador e permite move-la com WSAD. Permite mover a direção da câmera com o mouse ao apertá-lo.
Pós-condições	É alterada a posição do jogador no ambiente. Ao fazer de uso do mouse, é alterado a direção de visão do jogador.

3.3 Sistema de colisão

Neste terceiro estudo é verificado a condição de colisão com objetos. Para tal é verificado a colisão com a parede.

Pré-condições	Ter iniciado o programa
Procedimentos	1. Usar o mouse para apontar a câmera para a direção de um muro.
	2. Usar a tecla W para seguir em frente e tentar atravessar o muro.
Resultado esperado	O programa não deixo a câmera do jogador ultrapassar o muro e para o seu movimento.
Pós-condições	Câmera do jogador parada olhando para o muro.

4 Conclusão

4.1 Dificuldades encontradas

- Dificuldades em discubrir o modo com que o glut atrubui as funções e gerencia os eventos.
- Dificuldades em tornar o jogo jogável por multiplataformas; especificamente no tratamento de sons.
- Dificuldade em imprimir objetos 2d por cima do cenario 3d (minimap)

4.2 Sugestões

- Multiplayer para 2 jogadores Alternados.
- Registro de nome para usuarios que concluirem um nível com sua respectiva potuação.

Referências

[1] J. Pittman. the pac-man dossier. [Online]. Available: http://home.comcast.net/~jpittman2/pacman/pacmandossier.html

[2] T. V. developers and A. Roldan. Valgrind - manual. [Online]. Available: http://www.valgrind.org/docs/manual/index.html

Luiz Fernando Gomes de Oliveira
Matricula: 10/46969
E-mail: ziulolioveira@gmail.com

Gustavo Jaruga Cruz
Matricula: 09/0066634
E-mail: darkssha-des@hotmail.com

Guilherme Fay Vergara
Matricula: 10/45547
E-mail: guifayvergara@hotmail.com

APÊNDICE A
FIGURAS

A.1 Valgrind

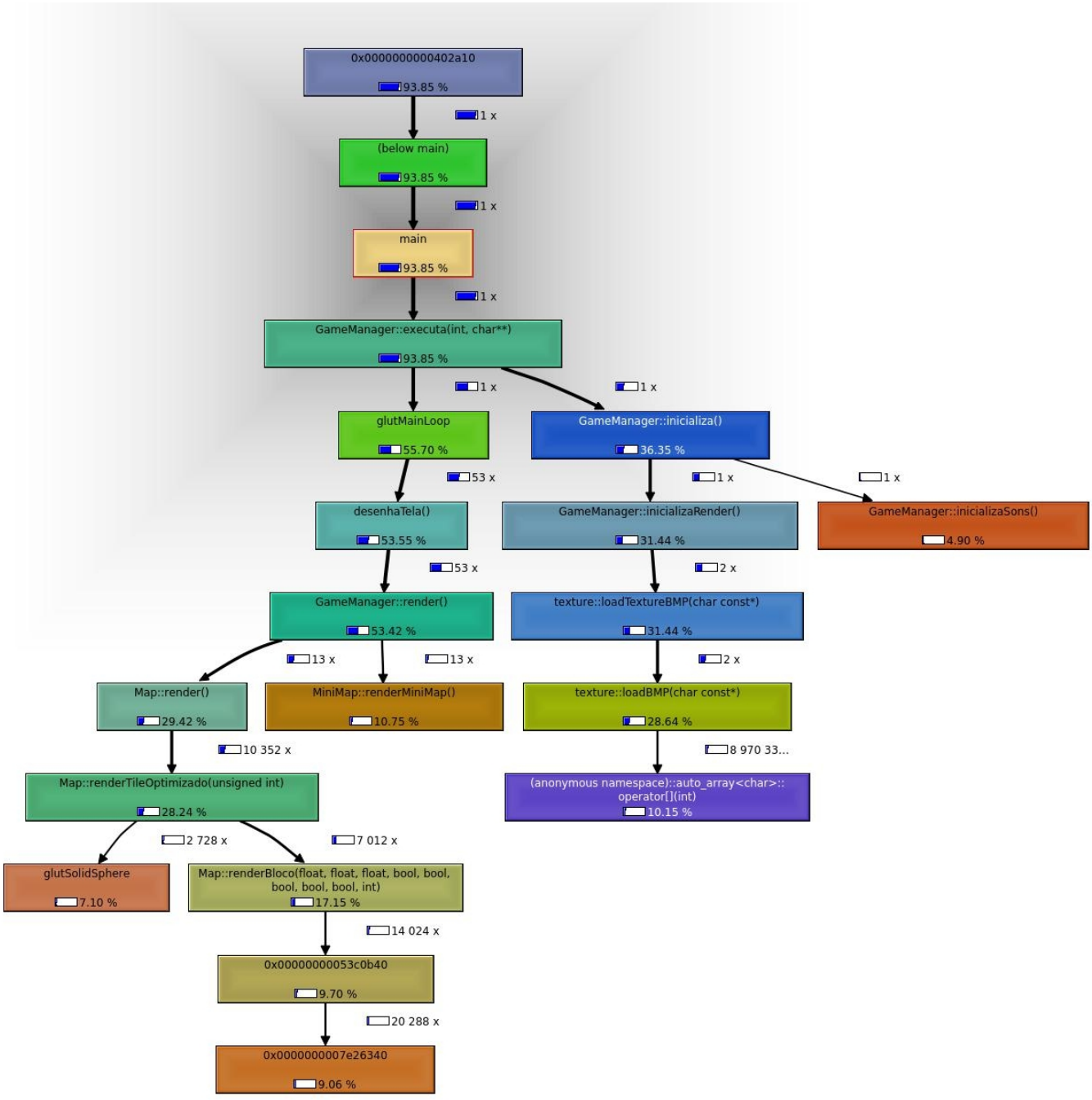

Figura 2: Saída gerada pelo Valgrind

A.2 Diagrama de Classes

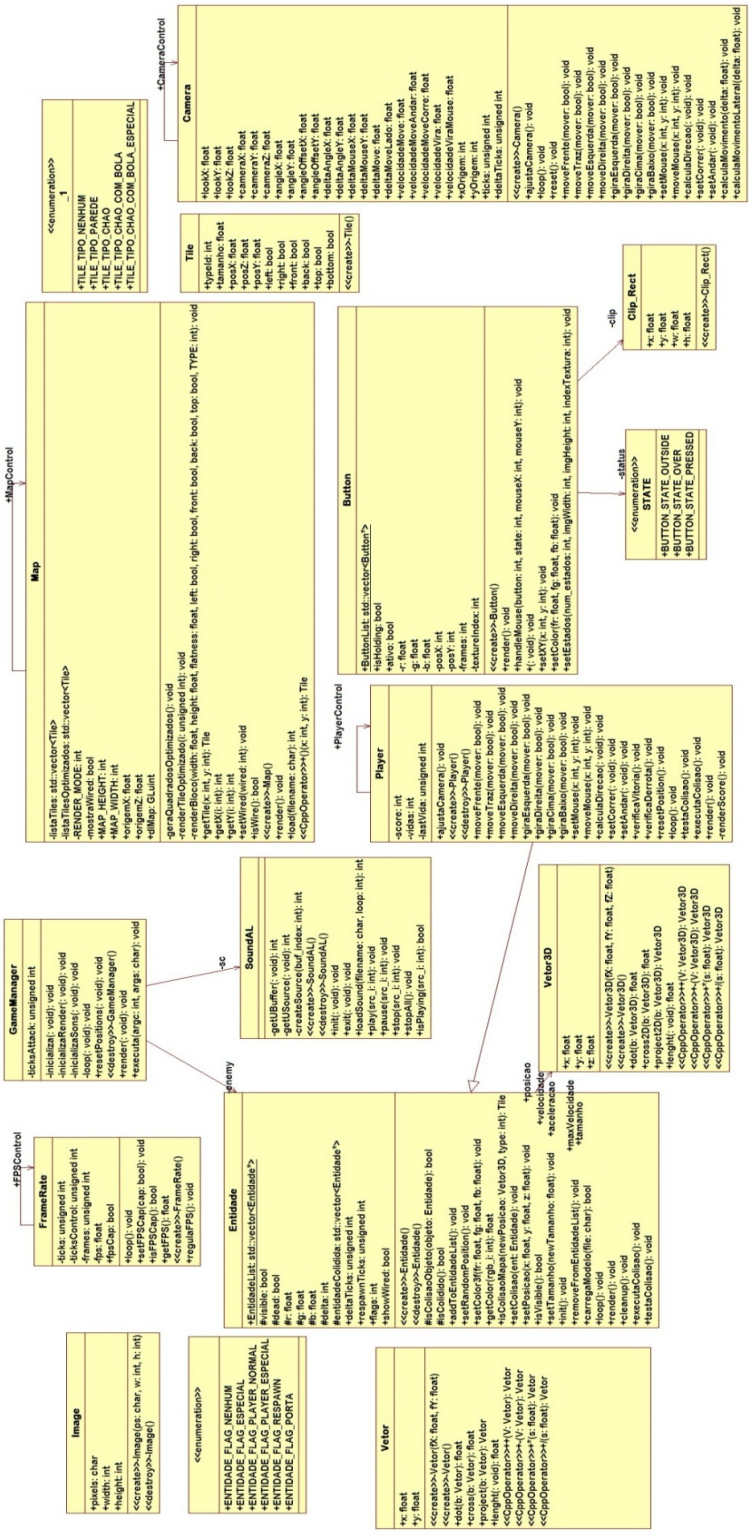

Figura 3: Diagrama de classes

APÊNDICE B
CÓDIGOS FONTES

B.1 Headers

B.1.1 Camera

```
#ifndef _CAMERAS_H_
#define _CAMERAS_H_

#include "defines.h"


#define CAMERA_ANDA 20
#define CAMERA_CORRE 40

class Camera
{
    public:
        float lookX, lookY, lookZ;
        float cameraX, cameraY, cameraZ;

        float angleX, angleY;
        float angleOffsetX, angleOffsetY;

        float deltaAngleX, deltaAngleY;
        float deltaMouseX, deltaMouseY;
        float deltaMove, deltaMoveLado;

        float velocidadeMove;
        float velocidadeMoveAndar;
        float velocidadeMoveCorre;
        float velocidadeVira;
        float velocidadeViraMouse;

        int xOrigem, yOrigem;
        unsigned int ticks;
        unsigned int deltaTicks;
    public:
        Camera();
        static Camera CameraControl;

        void ajustaCamera(); //Set position and direction of the camera
        void loop();  //set timer
        void reset();

        void moveFrente(bool mover);
        void moveTraz(bool mover);
        void moveEsquerda(bool mover);
        void moveDireita(bool mover);

        void giraEsquerda(bool mover);
        void giraDireita(bool mover);
        void giraCima(bool mover);
        void giraBaixo(bool mover);

        void setMouse(int x, int y);
        void moveMouse(int x, int y);
        //temp as public
        void calculaDirecao(void);

        //Turns run
        void setCorrer(void);
        void setAndar(void);


        void calculaMovimento(float delta);
        void calculaMovimentoLateral(float delta);

};
#endif
```

B.1.2 Entidade

```

#ifndef __ENTIDADE_H_
#define __ENTIDADE_H_

#include <vector>
#include "vetor3d.h"
#include "defines.h"
#include "map.h"
#include "camera.h"
#include "soundAL.h"
#include "model_obj.h"
```

```
12
13enum
14{
15    ENTIDADE_FLAG_NENHUM            =   0,
16    ENTIDADE_FLAG_ESPECIAL          =   0x00000001,
17    ENTIDADE_FLAG_PLAYER_NORMAL     =   0x00000002,
18    ENTIDADE_FLAG_PLAYER_ESPECIAL   =   0x00000004,
19    ENTIDADE_FLAG_RESPAWN           =   0x00000008,
20        //not used
21    ENTIDADE_FLAG_PORTA             =   0x00000016
22};
23
24
25class Entidade
26{
27    public:
28        static std::vector<Entidade*> EntidadeList;
29        Entidade();
30        virtual ~Entidade();
31    protected:
32        bool isColisaoObjeto(Entidade* objeto);
33        bool isColidido();
34        bool visible;
35        bool dead;
36
37        float r,g,b;
38
39        int delta;
40        std::vector<Entidade*> entidadeColidida;
41
42
43
44
45
46
47    public:
48        Model_OBJ obj;
49        void createModel(char* filename){obj.Load(filename);}
50        void addToEntidadeList();
51        void setRandomPosition();
52        void setColor3f(float fr, float fg, float fb);
53        float getColor(int rgb_i);
54        Tile* isColisaoMapa(Vetor3D newPosicao, int type = TILE_TIPO_PAREDE);
55        void setColisao(Entidade* ent);
56        void setPosicao(float x, float y, float z);
57        //Ex: int delta = getTicks() - deltaTicks;
58        //Ex: posicao = posicao + (velocidade * (delta/1000.f ) );
59        unsigned int deltaTicks; //quantos ms desde a ultima vez
60        unsigned int respawnTicks;// o tempo em q ele morreu
61        Vetor3D posicao;
62        Vetor3D velocidade;
63        Vetor3D aceleracao;
64        Vetor3D maxVelocidade;
65        Vetor3D tamanho;
66        int flags;
67        bool showWired;
68    public:
69        bool isVisible();
70        void setTamanho(float newTamanho);
71    public:
72        void init();
73        void removeFromEntidadeList();
74
75
76        virtual bool carregaModelo(char* file);
77        virtual void loop();
78        virtual void render();
79        virtual void cleanup();
80        virtual void executaColisao();
81        virtual void testaColisao();
82
83
84};
85
86
87#endif
```

B.1.3 Framerate

```
1#ifndef __FRAMERATE_H_
2#define __FRAMERATE_H_
3
4#include "defines.h"
5
6
7class FrameRate
```

```
{
    private:
        unsigned int ticks;
        unsigned int ticksControl;
        unsigned int frames;
        float fps;
    public:
        void loop();

        bool fpsCap;

        void setFPSCap(bool cap);
        bool isFPSCap();
        float getFPS();
        FrameRate();

        void regulaFPS();

        static FrameRate FPSControl;
};


#endif
```

B.1.4 Map

```
#ifndef _MAPS_H_
#define _MAPS_H_

#include "defines.h"
#include "tile.h"
#include "camera.h"
#include "text.h"
#include <vector>
#include <stdio.h>
#include <math.h>
#include "model_obj.h"


class Map
{
    private:
        std::vector<Tile> listaTiles;
        std::vector<Tile> listaTilesOptimizados;
        void geraQuadradosOptimizados();

        int RENDER_MODE;


        //void renderTile(unsigned int i);
        void renderTileOptimizado(unsigned int i);
        void renderBloco(float width, float height, float flatness, bool left,
                         bool right, bool front, bool back, bool top, int TYPE);


        bool mostraWired;
    public:
        Tile* getTile(int x, int y);
        inline int getX(int i);
        inline int getY(int i);

        void setWired(int wired);
        bool isWire();

        Map();

        //void render();
        void render();
        int load(char* filename);

        //Used to others classes to get info about the map
        static Map MapControl;
        //Operator overload
        inline Tile* operator ()  (const int x, const int y)
        {
            return this->getTile(x,y);
        }
        //Propriedades publicas
    public:
        int MAP_HEIGHT;
        int MAP_WIDTH;

        float origemX; // Where the map start to render
        float origemZ; //Tile 0,0, growls on right-down

        GLuint dlMap;
```

```
61
62          Model_OBJ coin;
63          Model_OBJ bigCoin;
64
65          float coinRotate;
66          float coinVelocidade;
67          //Usa pra calcular rotate
68          unsigned int deltaTicks;
69
70
71};
72
73
74#endif
```

B.1.5 Texture Loader

```
 1#ifndef _TEXTURELOADER_H_
 2#define _TEXTURELOADER_H_
 3
 4#include "defines.h"
 5
 6//Represents an image
 7class Image {
 8    public:
 9        Image(char* ps, int w, int h);
10        ~Image();
11
12        /* An array of the form (R1, G1, B1, R2, G2, B2, ...) indicating the
13         * color of each pixel in image.  Color components range from 0 to 255.
14         * The array starts the bottom-left pixel, then moves right to the end
15         * of the row, then moves up to the next column, and so on.  This is the
16         * format in which OpenGL likes images.
17         */
18         //Array de pixels no formato R,G,B, R1,G1,B1
19         //Comeca de baixo-esquerda, formato do openGL nativo
20        char* pixels;
21        int width;
22        int height;
23};
24
25#endif
26
27namespace texture
28{
29    //Le uma imagem BMP do arquivo
30    extern GLuint loadTextureBMP(const char* filename);
31    extern Image* loadBMP(const char* filename);
32}
```

B.1.6 Defines

```
 1#ifndef __DEFINESS__H_
 2#define __DEFINESS__H_
 3
 4
 5#if defined (__APPLE__) || defined (MACOSX) /*MAC OS*/
 6    #include <GLUT/glut.h>
 7    #include <OpenAL/alut.h>
 8    #include <OpenAL/al.h>
 9    #include <OpenAL/alc.h>
10
11#else
12    #ifdef _WIN32                           /* Windows */
13        #define WIN32_LEAN_AND_MEAN
14        #include <glee.h>
15        #include <gl/gl.h>
16        #include <gl/glut.h>
17        #include <windows.h>
18        #include <AL/al.h>
19        #include <AL/alc.h>
20        #include <AL/alut.h>
21
22        #define sleep(x) Sleep(x)
23    #else                                   /*Linux*/
24        #include <cstdarg>
25        #include <unistd.h>
26        #include <GL/gl.h>
27        #include <GL/glut.h>
28        #include <GL/glu.h>
29        #include <AL/al.h>
30        #include <AL/alc.h>
31        #include <AL/alut.h>
32
33        #define Sleep(x) usleep(x<1000000?10000+300*x:x)
34    #endif
```

```
35#endif
36
37#include <stdio.h>
38#include <stdlib.h>
39
40
41#define SCREEN_WIDTH            800
42#define SCREEN_HEIGHT           600
43
44#define FRAMES_PER_SECOND       60.0f
45
46#define TAMANHO_BLOCO           12
47#define COR_PAREDE              1.0f, 1.0f, 1.0f
48#define COR_CHAO                1.0f, 1.0f, 1.0f
49#define COR_COIN                1.0f, 1.0f, 1.0f
50#define COR_BIG_COIN            0.6f, 0.9f, 0.5f
51#define GAME_FOV                28
52
53#define PONTOS_BOLA             10
54#define PONTOS_BOLA_ESPECIAL    50
55
56#define TAMANHO_INIMIGO         5
57
58
59
60//Size of the current screen
61extern float wScreen;
62extern float hScreen;
63//textures
64extern GLuint wallTexture;
65extern GLuint floorTexture;
66//Menu
67extern bool menuPrincipal;
68extern int status;
69
70//Sounds
71extern int SOUND_main;
72extern int SOUND_inter1;
73extern int SOUND_inter2;
74extern int SOUND_inter3;
75extern int SOUND_attack;
76extern int SFX_die;
77extern int SFX_eat;
78extern int SFX_eat2;
79extern int SFX_alert;
80//Global from gameplay
81extern int attack_mode;
82
83#define STATUS_NORMAL 0
84#define STATUS_VITORIA 1
85#define STATUS_DERROTA 2
86
87
88
89#endif
```

B.1.7 Eventos

```
 1#ifndef EVENTOS_H_
 2#define EVENTOS_H_
 3
 4#define GLUT_KEY_ESC             27
 5#define GLUT_KEY_TAB              9
 6#define GLUT_KEY_RETURN          13
 7
 8extern void teclasNormais(unsigned char key, int x, int y);
 9extern void teclasNormaisUp(unsigned char key, int x, int y);
10extern void teclasEspeciais(int key, int x, int y );
11extern void teclasEspeciaisSoltar(int key, int x, int y);
12extern void mouseButton(int button, int state, int x, int y);
13extern void moveMouse(int x, int y);
14
15#endif
```

B.1.8 Game Maneger

```
 1//============================================================================
 2/*
 3    Classe que contera o metodo main e gerenciara o jogo.
 4    Class that will have the main method and care the game
 5*/
 6//============================================================================
 7#ifndef _GAME_MANAGER_H_
 8#define _GAME_MANAGER_H_
 9#include <cstdlib>
10#include "defines.h"
```

```
11#include "camera.h"
12#include "framerate.h"
13#include "map.h"
14#include "text.h"
15#include "entidade.h"
16#include "player.h"
17#include "minimap.h"
18#include "button.h"
19#include "soundAL.h"
20#include "textureloader.h"
21
22#define MAX_ENEMY 8
23
24///Note: the cleanup .cpp is called by atExit() in stdlib
25class GameManager
26{
27    private:
28        void inicializa(void);
29        void inicializaRender(void);
30        void inicializaSons(void);
31        void loop(void);
32
33        Entidade* enemy[MAX_ENEMY];
34        Model_OBJ coin;
35
36        //SoundController ... Controls sound
37        SoundAL sc;
38
39        unsigned int ticksAttack;
40    public:
41        void resetPositions(void);
42        ~GameManager();
43        void render(void);
44        void executa(int argc, char* args[]);
45        void Testes();
46};
47
48
49#endif
```

B.1.9 Text

```
1#ifndef __TEXTT__H_
2#define __TEXTT__H_
3
4#include "defines.h"
5#include <stdio.h>
6
7namespace txt
8{
9    extern void renderBitmapString(
10            float x,
11            float y,
12            int spacing,
13            void *font,
14            char *string) ;
15
16
17
18    ///ARRUMA PROJECOES
19    extern void setProjecaoOrto();
20    extern void restauraProjecaoPerspectiva();
21
22    extern void renderText2dOrtho(float x, float y, int spacing, const char*pStr, ...);
23
24}
25
26
27
28#endif
```

B.2 Sources

B.2.1 Camera

```
1#include "camera.h"
2
3#include <math.h>
4Camera Camera::CameraControl;
5Camera::Camera()
6{
7    angleX = 90.0f;
8    angleY = 0.0f;
9    angleOffsetX = angleOffsetY = 0;
10
11    lookX = 0.5f;
```

```
12      lookY = 0.0f;
13      lookZ = -1.0f;
14
15      cameraX = (TAMANHO_BLOCO*1) + TAMANHO_BLOCO/2;
16      cameraY = 5.0f;
17      cameraZ = (TAMANHO_BLOCO*1) + TAMANHO_BLOCO/2;
18      //tests
19
20      //tests
21      deltaAngleX = deltaAngleY = 0.0f; //Angle of rotation of the horizontal and vertical direction
22
23      deltaMouseX = deltaMouseY = 0.0f;
24
25      deltaMove = deltaMoveLado = 0.0f;
26
27
28      velocidadeMoveAndar = CAMERA_ANDA;
29      velocidadeMoveCorre = CAMERA_CORRE;
30      velocidadeMove = velocidadeMoveAndar;
31      velocidadeVira = 45.f;
32      velocidadeViraMouse = 0.1f;
33
34      xOrigem = -1;
35      yOrigem = -1;
36      ticks = 0;
37
38      calculaDirecao();
39  }
40
41  void Camera::reset()
42  {
43      angleX = 90.0f;
44      angleY = 0.0f;
45      angleOffsetX = angleOffsetY = 0;
46
47      lookX = 0.5f;
48      lookY = 0.0f;
49      lookZ = -1.0f;
50
51      cameraX = (TAMANHO_BLOCO*1) + TAMANHO_BLOCO/2;
52      cameraY = 5.0f;
53      cameraZ = (TAMANHO_BLOCO*1) + TAMANHO_BLOCO/2;
54      //tests
55
56      //tests
57      deltaAngleX = deltaAngleY = 0.0f; //Angle of rotation of the horizontal and vertical direction
58
59      deltaMouseX = deltaMouseY = 0.0f;
60
61      deltaMove = deltaMoveLado = 0.0f;
62
63
64      velocidadeMoveAndar = CAMERA_ANDA;
65      velocidadeMoveCorre = CAMERA_CORRE;
66      velocidadeMove = velocidadeMoveAndar;
67      velocidadeVira = 45.f;
68      velocidadeViraMouse = 0.1f;
69
70      xOrigem = -1;
71      yOrigem = -1;
72      ticks = 0;
73
74      calculaDirecao();
75      ticks = glutGet(GLUT_ELAPSED_TIME);
76  }
77
78
79  //Called internally by Player.
80  void Camera::ajustaCamera()
81  {
82
83      if (deltaAngleX || deltaAngleY)
84          calculaDirecao();
85
86      gluLookAt(  cameraX      , cameraY      , cameraZ,
87                  cameraX+lookX, cameraY+lookY, cameraZ+lookZ,
88                  0.0f   , 1.0f,    0.0f);
89
90      ticks = glutGet(GLUT_ELAPSED_TIME);
91  }
92
93  void Camera::loop()
94  {
95      deltaTicks = glutGet(GLUT_ELAPSED_TIME) - ticks;
96  }
97
```

```
 98 void Camera::calculaDirecao(void)
 99 {
100     float fator = deltaTicks/1000.f;
101     angleX += deltaAngleX*fator;
102     angleY += deltaAngleY*fator;
103 
104     //correct angle
105     if ( angleX+angleOffsetX >= 360 )
106         angleX -= 360;
107     if ( angleX+angleOffsetX < 0)
108         angleX += 360;
109 
110     //Only allows to rotate 180 degrees Y
111     if ( angleY+angleOffsetY >= 90 )
112         angleY = 90-angleOffsetY;
113     if ( angleY+angleOffsetY <= -90)
114         angleY = -(90+angleOffsetY);
115 
116 
117     lookX = sin( (angleX+angleOffsetX)*M_PI/180);
118     lookZ = cos( (angleX+angleOffsetX)*M_PI/180);
119 
120     lookY = sin( (angleY+angleOffsetY)*M_PI/180);
121 }
122 void Camera::calculaMovimento(float delta)
123 {
124     //Add the movement
125     float fator = deltaTicks/1000.f;
126 
127     //Factor delta times direction. 0.1f to adjust speed.
128     cameraX += (delta*fator) * lookX;
129     cameraZ += (delta*fator) * lookZ;
130 }
131 void Camera::calculaMovimentoLateral(float delta)
132 {
133     float fator = deltaTicks/1000.f;
134 
135     float lateralX = sin( (angleX-90)*M_PI/180);
136     float lateralZ = cos( (angleX-90)*M_PI/180);
137     //Add the movement
138     //Factor delta times direction. 0.1f to adjust speed.
139     cameraX += (delta*fator) * (lateralX);
140     cameraZ += (delta*fator) * (lateralZ);
141 }
142 
143 
144 void Camera::moveFrente(bool mover)
145 {
146     if(mover)
147         deltaMove = velocidadeMove;
148     else
149         deltaMove = 0.0f;
150 }
151 void Camera::moveTraz(bool mover)
152 {
153     if(mover)
154         deltaMove = -velocidadeMove;
155     else
156         deltaMove = 0.0f;
157 
158 }
159 void Camera::moveEsquerda(bool mover)
160 {
161     if(mover)
162         deltaMoveLado = -velocidadeMove;
163     else
164         deltaMoveLado = 0.0f;
165 }
166 void Camera::moveDireita(bool mover)
167 {
168     if(mover)
169         deltaMoveLado = velocidadeMove;
170     else
171         deltaMoveLado = 0.0f;
172 }
173 
174 void Camera::giraEsquerda(bool mover)
175 {
176     if(mover)
177         deltaAngleX = velocidadeVira;
178     else
179         deltaAngleX = 0.0f;
180 }
181 void Camera::giraDireita(bool mover)
182 {
183     if(mover)
```

```
184         deltaAngleX = -velocidadeVira;
185     else
186         deltaAngleX = 0.0f;
187 }
188 void Camera::giraCima(bool mover)
189 {
190     if(mover)
191         deltaAngleY = velocidadeVira;
192     else
193         deltaAngleY = 0.0f;
194 }
195 void Camera::giraBaixo(bool mover)
196 {
197     if(mover)
198         deltaAngleY = -velocidadeVira;
199     else
200         deltaAngleY = 0.0f;
201 }
202
203 void Camera::setMouse(int x, int y)
204 {
205     xOrigem = x;
206     yOrigem = y;
207
208     if (xOrigem == -1) //Both will be necessarily -1
209     {
210         angleX +=angleOffsetX;
211         angleY +=angleOffsetY;
212         angleOffsetX = 0;
213         angleOffsetY = 0;
214     }
215 }
216 void Camera::moveMouse(int x, int y)
217 {
218     deltaMouseX = deltaMouseY = 0;
219     //If there was displacement
220     if (xOrigem>0)
221     {
222         angleOffsetX = (xOrigem-x) * 0.1f;
223     }
224     if (yOrigem>0)
225     {
226         angleOffsetY = (yOrigem-y) * 0.1f;
227     }
228     calculaDirecao();
229 }
230
231 void Camera::setCorrer(void)
232 {
233     velocidadeMove = velocidadeMoveCorre;
234 }
235 void Camera::setAndar(void)
236 {
237     velocidadeMove = velocidadeMoveAndar;
238 }
```

B.2.2 Entidade

```
1 #include "entidade.h"
2
3 #include <stdlib.h>
4
5
6
7
8 //============================================================================
9 // static variables
10 //============================================================================
11 std::vector<Entidade*> Entidade::EntidadeList;
12
13 //============================================================================
14 // constructors
15 //============================================================================
16 Entidade::Entidade()
17 {
18     flags = ENTIDADE_FLAG_NENHUM;
19     entidadeColidida.clear();
20     deltaTicks = 9999999;
21     deltaTicks = 0;
22     tamanho.x = tamanho.y = tamanho.z = 10;
23     visible = true;
24     dead = false;
25     showWired = false;
26
27     r = 1.0f;
28     g = b = 0.0f;
```

```
29
30      maxVelocidade.x = maxVelocidade.y = maxVelocidade.z = 50.f;
31      entidadeColidida.clear();
32
33  }
34
35  void Entidade::init()
36  {
37      deltaTicks = glutGet(GLUT_ELAPSED_TIME);
38  }
39  Entidade::~Entidade()
40  {
41
42  }
43  void Entidade::cleanup()
44  {
45  }
46  bool Entidade::isColisaoObjeto(Entidade* objeto)
47  {
48      //Note: The point marks position 0 .... ex: position 0 beginning of the block end of the block in the x, y, z
49      //Such that y lower = y ; y highest = y+tamanhoY
50      int baixo1 = this->posicao.y;
51      int cima1 = this->posicao.y + this->tamanho.y;
52      int esquerda1 = this->posicao.x;
53      int direita1 = this->posicao.x + this->tamanho.x;
54      int frente1 = this->posicao.z;
55      int traz1 = this->posicao.z + this->tamanho.z;
56
57      int baixo2 = objeto->posicao.y;
58      int esquerda2 = objeto->posicao.x;
59      int frente2 = objeto->posicao.z;
60      int direita2 = objeto->posicao.x + objeto->tamanho.x;
61      int cima2 = objeto->posicao.y + objeto->tamanho.y;
62      int traz2 = objeto->posicao.z + objeto->tamanho.z;
63
64      if (
65          !(baixo1 > cima2) &&
66          !(cima1  < baixo2) &&
67          !(esquerda1 > direita2) &&
68          !(direita1 < esquerda2) &&
69          !(frente1 > traz2) &&
70          !(traz1 < frente2)
71          )
72          {
73              return true;
74          }
75
76      return false;
77
78  }
79  //=============================================================================
80  // Returns true if colliding with the map
81  //=============================================================================
82  Tile* Entidade::isColisaoMapa(Vetor3D newPosicao, int type)
83  {
84      //Calculates Id tile to be tested
85      //Ex: X = 5 Such that startX = 0,41 = 0 endX = 1,3 = 1
86      int startX = (newPosicao.x) / TAMANHO_BLOCO;
87      int startZ = (newPosicao.z) / TAMANHO_BLOCO;
88      int endX = (newPosicao.x + (tamanho.x)) / TAMANHO_BLOCO;
89      int endZ = (newPosicao.z + (tamanho.z)) / TAMANHO_BLOCO;
90
91      //Check collisions with tiles
92      for(int iZ = startZ; iZ <= endZ; iZ++) {
93          for(int iX = startX; iX <= endX; iX++) {
94              Tile* bloco = Map::MapControl(iX, iZ);
95
96              if(
97                 (bloco->typeId == type) &&
98                 (posicao.y < (bloco->posY+bloco->tamanho) ) &&
99                 ((posicao.y+tamanho.y) > bloco->posY)
100                )
101                 return bloco;
102             }
103     }
104     return 0;
105 }
106
107 void Entidade::removeFromEntidadeList()
108 {
109     for(unsigned int i = 0; i < EntidadeList.size(); i++)
110     {
111         if (EntidadeList[i] == this)
112             EntidadeList.erase(EntidadeList.begin()+i);
113     }
114 }
```

```
115 void Entidade::addToEntidadeList()
116 {
117
118
119     for(unsigned int i = 0; i < EntidadeList.size(); i++)
120     {
121         if (EntidadeList[i] == this)
122             return; //Se ja estiver na lista, retorna
123     }
124
125     EntidadeList.push_back(this);
126 }
127
128 bool Entidade::carregaModelo(char* file){return true;}
129 //=========================================================================
130 // Performs actions of the loop, acceleration, speed.
131 //=========================================================================
132 void Entidade::loop()
133 {
134     //3 seconds has the spawn
135     if ( (flags == ENTIDADE_FLAG_RESPAWN) && ( (glutGet(GLUT_ELAPSED_TIME) - respawnTicks) > 3000) )
136     {
137         dead = false;
138         visible = true;
139         setRandomPosition();
140         flags = ENTIDADE_FLAG_NENHUM;
141     }
142
143     if(dead) return;
144     //deltaTicks reset the surrender
145     delta = glutGet(GLUT_ELAPSED_TIME) - deltaTicks;
146     float fator = delta/1000.f;
147
148     //calculates accelerations
149     if ( velocidade.x + aceleracao.x <= maxVelocidade.x)
150         velocidade.x += (aceleracao.x * fator);
151     if ( velocidade.y + aceleracao.y <= maxVelocidade.y)
152         velocidade.y += (aceleracao.y * fator);
153     if ( velocidade.z + aceleracao.z <= maxVelocidade.z)
154         velocidade.z += (aceleracao.z * fator);
155
156     Vetor3D newPosicao = posicao + (velocidade * fator );
157
158     if (isColisaoMapa(newPosicao) == false)
159         posicao = newPosicao;
160     else
161     {
162         velocidade.x = 0;
163         velocidade.z = 0;
164         aceleracao.x = 0;
165         aceleracao.z = 0;
166         int pos = (int)(rand() % 4);
167         switch(pos)
168         {
169             case 0:
170                 aceleracao.x = 20;break;
171             case 1:
172                 aceleracao.x = -20;break;
173             case 2:
174                 aceleracao.z = 20;break;
175             case 3:
176                 aceleracao.z = -20;break;
177             default:;
178         }
179
180     }
181
182     deltaTicks = glutGet(GLUT_ELAPSED_TIME);
183 }
184 void Entidade::render()
185 {
186     if (!isVisible())
187         return;
188
189     int tamanhoCubo = tamanho.x; //Temp while using glutCube
190     glPushMatrix();
191     //Centers due to GLUT
192     if (flags == ENTIDADE_FLAG_ESPECIAL)
193         glColor3f( getColor(1), getColor(2), getColor(3) );
194     else
195         glColor3f(r,g,b);
196     glTranslated(posicao.x+tamanho.x/2,
197                  posicao.y+tamanho.y/2,
198                  posicao.z+tamanho.z/2);
199     if (showWired)
200         glutWireCube(tamanhoCubo);
```

```
201     else
202     {
203         glScaled(2.0f, 2.0f, 2.0f);
204         obj.Draw();
205 
206     }
207 
208     glPopMatrix();
209 
210 
211 }
212 void Entidade::testaColisao()
213 {
214     if(dead) return;
215 
216     unsigned int thisID = -1;
217     for (unsigned int i = 0; i < EntidadeList.size(); i++)
218         if (EntidadeList[i] == this)
219         {
220             thisID = i;
221             break;
222         }
223     //Tests with all the entities of this forward.
224     //Ex:    lista: 1 2 3 4
225     // thisID =1,  tests with 2, 3 , 4
226     // thisID = 2  tests with 3, 4      this way, thisID = 2 no collisions with 1 as has already been tested previously.
227     for (unsigned int i = thisID+1; i < EntidadeList.size(); i++)
228     {
229         if (EntidadeList[i] != this && !EntidadeList[i]->dead)
230         {
231             if(isColisaoObjeto(EntidadeList[i]) )
232             {   //adds this element collisions so as tested in
233                 setColisao(EntidadeList[i]);
234                 EntidadeList[i]->setColisao(this);
235             }
236         }
237     }
238 }
239 //Set collision through the public method
240 void Entidade::setColisao(Entidade* ent)
241 {
242     entidadeColidida.push_back(ent);
243 }
244 bool Entidade::isColidido()
245 {
246     if (entidadeColidida.size() == 0)
247         return false;
248     else
249         return true;
250 }
251 void Entidade::executaColisao()
252 {
253     if ( !isColidido() )
254         return; // no collisions
255 
256 
257 /*
258 
259     //Back what had moved.
260     float fator = delta/1000.f;
261     posicao = posicao - (velocidade * fator );
262     //For, and go in the opposite direction
263     velocidade.x = 0;
264     velocidade.z = 0;
265     aceleracao.x = -aceleracao.x;
266     aceleracao.z = -aceleracao.z;
267 */
268     if ( (flags == ENTIDADE_FLAG_ESPECIAL) && (entidadeColidida[0]->flags == ENTIDADE_FLAG_PLAYER_ESPECIAL) )
269     {
270         flags = ENTIDADE_FLAG_RESPAWN;
271         respawnTicks = glutGet(GLUT_ELAPSED_TIME);
272         dead = true;
273         visible = false;
274         SoundAL sc;
275         sc.play(SFX_eat2);
276     }
277 
278     entidadeColidida.clear();
279 }
280 
281 void Entidade::setRandomPosition()
282 {
283     bool isOK = false;
284         while(!isOK) {
285             int posX = rand() % Map::MapControl.MAP_WIDTH;
286             int posZ = rand() % Map::MapControl.MAP_HEIGHT;
```

```
287
288             //If the position is different from the wall, then ground .... put cube
289             if (Map::MapControl.getTile(posX, posZ)->typeId != TILE_TIPO_PAREDE) {
290                 //Note: (TAMANHO_BLOCO/2 - tamanho.x/2)  is used to find the center of the floor
291                 posicao.x = (TAMANHO_BLOCO/2 - tamanho.x/2) + TAMANHO_BLOCO*posX;
292                 posicao.y = 0;
293                 posicao.z = (TAMANHO_BLOCO/2 - tamanho.z/2) + TAMANHO_BLOCO*posZ;
294                 //1 to 10
295                 aceleracao.x = 1 + rand() % 10;
296                 aceleracao.z = 1 + rand() % 10;
297                 init();
298                 isOK = true;
299                 ///Possible to add verification that the entity was not in the same place using
300                 //isColisao and clear() from list of collisions
301             }
302         }
303}
304
305bool Entidade::isVisible()
306{
307     return visible;
308}
309void Entidade::setTamanho(float newTamanho)
310{
311     tamanho.x = tamanho.y = tamanho.z = newTamanho;
312}
313void Entidade::setPosicao(float x, float y, float z)
314{
315     posicao.x = x;
316     posicao.y = y;
317     posicao.z = z;
318}
319void Entidade::setColor3f(float fr, float fg, float fb)
320{
321     r = fr;
322     g = fg;
323     b = fb;
324}
325float Entidade::getColor(int rgb_i)
326{
327     float color = 0.0f;
328     switch(rgb_i)
329     {
330         case 1:
331             color = r;
332             if (flags == ENTIDADE_FLAG_ESPECIAL)
333                 color -= 0.55f;
334             break;
335         case 2:
336             color = g;
337             if (flags == ENTIDADE_FLAG_ESPECIAL)
338                 color += 1;
339             break;
340         case 3:
341             color = b;
342             if (flags == ENTIDADE_FLAG_ESPECIAL)
343                 color += 0.95f;
344             break;
345     }
346     return color;
347}
```

B.2.3 Framerate

```
 1#include "framerate.h"
 2
 3
 4FrameRate FrameRate::FPSControl;
 5
 6
 7
 8float FrameRate::getFPS()
 9{
10     return fps;
11}
12void FrameRate::setFPSCap(bool cap)
13{
14     fpsCap = cap;
15}
16bool FrameRate::isFPSCap()
17{
18     return fpsCap;
19}
20FrameRate::FrameRate()
21{
22     ticks = glutGet(GLUT_ELAPSED_TIME);
```

```
23     ticksControl = glutGet(GLUT_ELAPSED_TIME);
24     frames = 0;
25     fps = 0;
26     fpsCap = false;
27 }
28
29 void FrameRate::regulaFPS()
30 {
31     unsigned int step = 1000.0f/FRAMES_PER_SECOND;
32     unsigned int decorrido = glutGet(GLUT_ELAPSED_TIME) - ticksControl;
33     if(decorrido < step )
34         Sleep( step - decorrido);
35
36     ticksControl = glutGet(GLUT_ELAPSED_TIME);
37 }
38
39 void FrameRate::loop()
40 {
41     unsigned int decorrido = glutGet(GLUT_ELAPSED_TIME) - ticks;
42     frames++;
43     if (decorrido > 1000)
44     {
45         fps = ((float)frames*1000.0f/(float)decorrido);
46
47         frames = 0;
48         ticks = glutGet(GLUT_ELAPSED_TIME);
49     }
50
51     if (fpsCap)
52         regulaFPS();
53
54 }
```

B.2.4 Map

```
 1 #include "map.h"
 2
 3 //Used by others classes to get info about the map
 4 Map Map::MapControl;
 5
 6 //Take the Title in position x,y of the map
 7 //Ex: Map 1 2 3   vector sera 1 2 3 4 5 6
 8 //        4 5 6
 9 Tile* Map::getTile(int x, int y)
10 {
11     unsigned int ID = 0;
12
13     ID = (y * MAP_WIDTH) + x;
14
15     return &listaTilesOptimizados[ID];
16 }
17 inline int Map::getX(int i)
18 {
19     return i % MAP_WIDTH;
20 }
21 inline int Map::getY(int i)
22 {
23     return (int) i/MAP_WIDTH;
24 }
25
26 Map::Map()
27 {
28     origemX = -TAMANHO_BLOCO;
29     origemZ = -TAMANHO_BLOCO;
30     mostraWired = false;
31     RENDER_MODE = 0x0007; //GL_QUADS
32     coinRotate = 0;
33     coinVelocidade = 180;
34 }
35
36 void Map::renderBloco(float width, float height, float flatness, bool left,
37         bool right, bool front, bool back, bool top, int TYPE = GL_QUADS)
38 {
39     float w = width/2;
40     float h = height/2;
41     float f = flatness/2;
42
43     float xTexNumber = width/TAMANHO_BLOCO;
44
45     glEnable(GL_TEXTURE_2D);
46     glBindTexture(GL_TEXTURE_2D, wallTexture);
47     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
48     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
49
50
51     glBegin(TYPE);
```

```
52      //Front
53      if(front)
54      {
55           glNormal3f(0.0f, 0.0f, 1.0f);
56              //glNormal3f(-1.0f, 0.0f, 1.0f);
57              glTexCoord2f(0.0f, 0.0f);
58          glVertex3f(-w, -h, f);
59              //glNormal3f(1.0f, 0.0f, 1.0f);
60              glTexCoord2f(xTexNumber, 0.0f);
61          glVertex3f(w, -h, f);
62              //glNormal3f(1.0f, 0.0f, 1.0f);
63              glTexCoord2f(xTexNumber, 1.0f);
64          glVertex3f(w, h, f);
65              //glNormal3f(-1.0f, 0.0f, 1.0f);
66              glTexCoord2f(0.0f, 1.0f);
67          glVertex3f(-w, h, f);
68      }
69
70      //Right
71      if(right)
72      {
73           glNormal3f(1.0f, 0.0f, 0.0f);
74              //glNormal3f(1.0f, 0.0f, -1.0f);
75              glTexCoord2f(0.0f, 0.0f);
76          glVertex3f(w, -h, -f);
77              //glNormal3f(1.0f, 0.0f, -1.0f);
78              glTexCoord2f(0.0f, 1.0f);
79          glVertex3f(w, h, -f);
80              glTexCoord2f(1.0f, 1.0f);
81              //glNormal3f(1.0f, 0.0f, 1.0f);
82          glVertex3f(w, h, f);
83              glTexCoord2f(1.f, 0.0f);
84              //glNormal3f(1.0f, 0.0f, 1.0f);
85          glVertex3f(w, -h, f);
86      }
87
88      //Back
89      if(back)
90      {
91              glNormal3f(0.0f, 0.0f, -1.0f);
92              //glNormal3f(-1.0f, 0.0f, -1.0f);
93              glTexCoord2f(0.0f, 0.0f);
94          glVertex3f(-w, -h, -f);
95              //glNormal3f(-1.0f, 0.0f, -1.0f);
96              glTexCoord2f(0.0f, 1.0f);
97          glVertex3f(-w, h, -f);
98              //glNormal3f(1.0f, 0.0f, -1.0f);
99              glTexCoord2f(xTexNumber, 1.0f);
100         glVertex3f(w, h, -f);
101             //glNormal3f(1.0f, 0.0f, -1.0f);
102             glTexCoord2f(xTexNumber, 0.0f);
103         glVertex3f(w, -h, -f);
104     }
105
106
107     //Left
108     if(left)
109     {
110         glNormal3f(-1.0f, 0.0f, 0.0f);
111             //glNormal3f(-1.0f, 0.0f, -1.0f);
112             glTexCoord2f(0.0f, 0.0f);
113         glVertex3f(-w, -h, -f);
114             //glNormal3f(-1.0f, 0.0f, 1.0f);
115             glTexCoord2f(1.0f, 0.0f);
116         glVertex3f(-w, -h, f);
117             //glNormal3f(-1.0f, 0.0f, 1.0f);
118             glTexCoord2f(1.0f, 1.0f);
119         glVertex3f(-w, h, f);
120             //glNormal3f(-1.0f, 0.0f, -1.0f);
121             glTexCoord2f(0.0f, 1.0f);
122         glVertex3f(-w, h, -f);
123     }
124     glEnd();
125 glDisable(GL_TEXTURE_2D);
126     glBegin(TYPE);
127     //Top
128     if(top)
129     {
130         glNormal3f(0.0f, 1.0f, 0.0f);
131         //glNormal3f(-1.0f, 1.0f, -1.0f);
132         glVertex3f(-w, h, -f);
133             //glNormal3f(-1.0f, 1.0f, 1.0f);
134         glVertex3f(-w, h, f);
135             //glNormal3f(1.0f, 1.0f, 1.0f);
136         glVertex3f(w, h, f);
137             //glNormal3f(1.0f, 1.0f, -1.0f);
```

```
138         glVertex3f(w, h, -f);
139     }
140 
141     // Don't need background
142     /*
143     //Bottom
144     glNormal3f(0.0f, -1.0f, 0.0f);
145         //glNormal3f(-1.0f, -1.0f, -1.0f);
146     glVertex3f(-w, -h, -f);
147         //glNormal3f(-1.0f, -1.0f, 1.0f);
148     glVertex3f(-w, -h, f);
149         //glNormal3f(1.0f, -1.0f, 1.0f);
150     glVertex3f(w, -h, f);
151         //glNormal3f(1.0f, -1.0f, -1.0f);
152     glVertex3f(w, -h, -f);
153     */
154     glEnd();
155 }
156 
157 void Map::render()
158 {
159     glPushMatrix();
160     float offset = (float)TAMANHO_BLOCO/2.0f;
161 
162     // Glut start printing starting from the center
163     glTranslated(offset, offset, offset);
164     glColor3f(COR_PAREDE);
165 
166     int indexX = (Camera::CameraControl.cameraX / TAMANHO_BLOCO);
167     int indexY = (Camera::CameraControl.cameraZ / TAMANHO_BLOCO);
168 
169     int beginX = indexX - GAME_FOV;
170     int beginY = indexY - GAME_FOV;
171     int endX = indexX + GAME_FOV;
172     int endY = indexY + GAME_FOV;
173     if(endX > MAP_WIDTH)
174             endX = MAP_WIDTH;
175     if(endY > MAP_HEIGHT)
176         endY = MAP_HEIGHT;
177     if(beginX < 0)
178         beginX = 0;
179     if(beginY < 0)
180         beginY = 0;
181 
182 
183     for(int i = beginY; i < endY; i++)
184     {
185         for(int j = beginX; j < endX; j++)
186         {
187             glPushMatrix();
188                 renderTileOptimizado(j+i*MAP_WIDTH);
189             glPopMatrix();
190         }
191     }
192 
193     //Desenha chao
194     glPopMatrix();
195 
196 }
197 void Map::renderTileOptimizado(unsigned int i)
198 {
199     //Gera fator para a rotacao da moeda
200     int delta = glutGet(GLUT_ELAPSED_TIME) - deltaTicks;
201     float fator = delta/1000.f;
202     deltaTicks = glutGet(GLUT_ELAPSED_TIME);
203 
204     coinRotate += coinVelocidade*fator;
205 
206     if ( coinRotate > 360 )
207         coinRotate-=360;
208 
209 
210     //Camera centra em 0,0,0
211     glTranslated(listaTilesOptimizados[i].posX * TAMANHO_BLOCO,
212                  listaTilesOptimizados[i].posY * TAMANHO_BLOCO,
213                  listaTilesOptimizados[i].posZ * TAMANHO_BLOCO);
214 
215 
216     if(listaTilesOptimizados[i].typeId == TILE_TIPO_PAREDE )
217     {
218         glColor3f(COR_PAREDE);
219         renderBloco(listaTilesOptimizados[i].tamanho, listaTilesOptimizados[i].tamanho, listaTilesOptimizados[i].tamanho,
220                     listaTilesOptimizados[i].left,listaTilesOptimizados[i].right,listaTilesOptimizados[i].front,
221                     listaTilesOptimizados[i].back,listaTilesOptimizados[i].top,
222                     RENDER_MODE);
223 
```

```
224     }
225     else //Print ground
226     {
227         glEnable(GL_TEXTURE_2D);
228         glBindTexture(GL_TEXTURE_2D, floorTexture);
229         glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
230         glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
231 
232         float offset = (float)TAMANHO_BLOCO/2.0f;
233         glColor3f(COR_CHAO);
234         glBegin(RENDER_MODE);
235             glNormal3f(0.0f, 1.0f, 0.0f);
236                 glTexCoord2f(0.0f, 0.0f);
237             glVertex3f(-offset, -offset, -offset);
238                 glTexCoord2f(0.0f, 1.0f);
239             glVertex3f(-offset, -offset, offset);
240                 glTexCoord2f(1.0f, 1.0f);
241             glVertex3f(offset, -offset, offset);
242                 glTexCoord2f(1.0f, 0.0f);
243             glVertex3f(offset, -offset, -offset);
244         glEnd();
245 
246         glDisable(GL_TEXTURE_2D);
247         if (listaTilesOptimizados[i].typeId == TILE_TIPO_CHAO_COM_BOLA)
248         {
249             glTranslated(0,-2,0);
250             //glutSolidSphere(1,8,8);
251             glRotatef(coinRotate, 0, 1, 0);
252             glColor3f(COR_COIN);
253             coin.Draw();
254         }
255         else
256         if (listaTilesOptimizados[i].typeId == TILE_TIPO_CHAO_COM_BOLA_ESPECIAL)
257         {
258             glTranslated(0,-2,0);
259             //glutSolidSphere(3,8,8);
260             glRotatef(coinRotate, 0, 1, 0);
261             glColor3f(COR_BIG_COIN);
262             bigCoin.Draw();
263         }
264 
265     }
266 }
267 
268 
269 int Map::load(char* filename)
270 {
271     listaTiles.clear();
272 
273     FILE* file = fopen(filename, "r");
274 
275     if(file == NULL)
276         return -1;
277 
278     MAP_HEIGHT = MAP_WIDTH = 0;
279 
280     // Take the map size (blocks)
281     int error = fscanf(file, "%d-%d\n", &MAP_WIDTH, &MAP_HEIGHT);
282 
283     for (int y = 0; y < MAP_HEIGHT; y++)
284     {
285         for (int x = 0; x < MAP_WIDTH; x++)
286         {
287             Tile tempTile;
288             error = fscanf(file, "[%d] ",&tempTile.typeId);
289 
290             listaTiles.push_back(tempTile);
291         }
292         error = fscanf(file, "\n");
293     }
294     fclose(file);
295     ///TEST
296     geraQuadradosOptimizados();
297     return error;
298 }
299 
300 void Map::geraQuadradosOptimizados()
301 {
302     listaTilesOptimizados.clear();
303 
304     for(int iY = 0; iY < MAP_HEIGHT; iY++)
305     {
306        for(int iX = 0; iX < MAP_WIDTH; iX++) //Test all the blocks after this one in X
307        {
308            Tile retangulo;
309            int index = iX + MAP_WIDTH*iY;
```

```
310             if (listaTiles[index].typeId != TILE_TIPO_PAREDE)
311             {
312                 retangulo.typeId = listaTiles[index].typeId;
313                 retangulo.posX = iX;
314                 retangulo.posZ = iY;
315                 listaTilesOptimizados.push_back(retangulo);
316                 continue;
317             }
318 
319              retangulo.top = true;
320              //If wall, check out of the boards
321              if (index-1 < 0)
322                  retangulo.left = true;
323              else // If ground, than have any wall in this direction
324                  if (listaTiles[index-1].typeId != TILE_TIPO_PAREDE)
325                      retangulo.left = true;
326              if (index - MAP_WIDTH < 0)
327                  retangulo.back = true;
328              else // If ground, than have any wall in this direction
329                  if (listaTiles[index - MAP_WIDTH].typeId != TILE_TIPO_PAREDE)
330                      retangulo.back = true;
331              if (index +1 >= (int)listaTiles.size())
332                  retangulo.right = true;
333              else // If ground, than have any wall in this direction
334                  if (listaTiles[index +1].typeId != TILE_TIPO_PAREDE)
335                      retangulo.right = true;
336              if (index + MAP_WIDTH >= (int)listaTiles.size())
337                  retangulo.front = true;
338              else // If ground, than have any wall in this direction
339                  if (listaTiles[index + MAP_WIDTH].typeId != TILE_TIPO_PAREDE)
340                      retangulo.front = true;
341 
342              retangulo.posX = iX;
343              retangulo.posZ = iY;
344              retangulo.typeId = listaTiles[index].typeId;
345 
346              listaTilesOptimizados.push_back(retangulo);
347 
348         }
349     }
350 }
351 
352 
353 
354 void Map::setWired(int wired)
355 {
356     if (wired)
357     {
358         mostraWired = true;
359         RENDER_MODE = GL_LINES;
360     }
361     else
362     {
363         mostraWired = false;
364         RENDER_MODE = GL_QUADS;
365     }
366 
367 }
368 bool Map::isWire()
369 {
370     return mostraWired;
371 }
```

B.2.5 Texture Loader

```
 1 #include "textureloader.h"
 2 
 3 #include <assert.h>
 4 #include <fstream>
 5 
 6 using namespace std;
 7 
 8 
 9 Image::Image(char* ps, int w, int h) : pixels(ps), width(w), height(h) {
10 
11 }
12 
13 Image::~Image() {
14     delete[] pixels;
15 }
16 
17 namespace {
18     //Converts a four-character array to an integer, using little-endian form
19     int toInt(const char* bytes) {
20         return (int)(((unsigned char)bytes[3] << 24) |
21                      ((unsigned char)bytes[2] << 16) |
```

```
22                      ((unsigned char)bytes[1] << 8) |
23                      (unsigned char)bytes[0]);
24      }
25
26      //Converts a two-character array to a short, using little-endian form
27      short toShort(const char* bytes) {
28          return (short)(((unsigned char)bytes[1] << 8) |
29                         (unsigned char)bytes[0]);
30      }
31
32      //Reads the next four bytes as an integer, using little-endian form
33      int readInt(ifstream &input) {
34          char buffer[4];
35          input.read(buffer, 4);
36          return toInt(buffer);
37      }
38
39      //Reads the next two bytes as a short, using little-endian form
40      short readShort(ifstream &input) {
41          char buffer[2];
42          input.read(buffer, 2);
43          return toShort(buffer);
44      }
45
46      //Just like auto_ptr, but for arrays
47      template<class T>
48      class auto_array {
49          private:
50              T* array;
51              mutable bool isReleased;
52          public:
53              explicit auto_array(T* array_ = NULL) :
54                  array(array_), isReleased(false) {
55              }
56
57              auto_array(const auto_array<T> &aarray) {
58                  array = aarray.array;
59                  isReleased = aarray.isReleased;
60                  aarray.isReleased = true;
61              }
62
63              ~auto_array() {
64                  if (!isReleased && array != NULL) {
65                      delete[] array;
66                  }
67              }
68
69              T* get() const {
70                  return array;
71              }
72
73              T &operator*() const {
74                  return *array;
75              }
76
77              void operator=(const auto_array<T> &aarray) {
78                  if (!isReleased && array != NULL) {
79                      delete[] array;
80                  }
81                  array = aarray.array;
82                  isReleased = aarray.isReleased;
83                  aarray.isReleased = true;
84              }
85
86              T* operator->() const {
87                  return array;
88              }
89
90              T* release() {
91                  isReleased = true;
92                  return array;
93              }
94
95              void reset(T* array_ = NULL) {
96                  if (!isReleased && array != NULL) {
97                      delete[] array;
98                  }
99                  array = array_;
100             }
101
102             T* operator+(int i) {
103                 return array + i;
104             }
105
106             T &operator[](int i) {
107                 return array[i];
```

```
108             }
109     };
110}
111
112namespace texture {
113     GLuint loadTextureBMP(const char* filename)
114     {
115         Image* image = loadBMP(filename);
116
117         GLuint textureId;
118         glGenTextures(1, &textureId); //Make room for our texture
119         glBindTexture(GL_TEXTURE_2D, textureId); //Tell OpenGL which texture to edit
120         //Map the image to the texture
121         glTexImage2D(GL_TEXTURE_2D,                 //Always GL_TEXTURE_2D
122                      0,                             //0 for now
123                      GL_RGB,                        //Format OpenGL uses for image
124                      image->width, image->height,  //Width and height
125                      0,                             //The border of the image
126                      GL_RGB, //GL_RGB, because pixels are stored in RGB format
127                      GL_UNSIGNED_BYTE, //GL_UNSIGNED_BYTE, because pixels are stored
128                                        //as unsigned numbers
129                      image->pixels);                //The actual pixel data
130
131         delete image;
132         return textureId; //Retorna id da textura
133
134     }
135
136     Image* loadBMP(const char* filename) {
137         ifstream input;
138         input.open(filename, ifstream::binary);
139         assert(!input.fail() || !"Could not find file");
140         char buffer[2];
141         input.read(buffer, 2);
142         assert( (buffer[0] == 'B' && buffer[1] == 'M' ) || !"Not a bitmap file");
143         input.ignore(8);
144         int dataOffset = readInt(input);
145
146         //Read the header
147         int headerSize = readInt(input);
148         int width;
149         int height;
150         switch(headerSize) {
151             case 40:
152                 //V3
153                 width = readInt(input);
154                 height = readInt(input);
155                 input.ignore(2);
156                 assert(readShort(input) == 24 || !"Image is not 24 bits per pixel");
157                 assert(readShort(input) == 0 || !"Image is compressed");
158                 break;
159             case 12:
160                 //OS/2 V1
161                 width = readShort(input);
162                 height = readShort(input);
163                 input.ignore(2);
164                 assert(readShort(input) == 24 || !"Image is not 24 bits per pixel");
165                 break;
166             case 64:
167                 //OS/2 V2
168                 assert(!"Can't load OS/2 V2 bitmaps");
169                 break;
170             case 108:
171                 //Windows V4
172                 assert(!"Can't load Windows V4 bitmaps");
173                 break;
174             case 124:
175                 //Windows V5
176                 assert(!"Can't load Windows V5 bitmaps");
177                 break;
178             default:
179                 assert(!"Unknown bitmap format");
180         }
181
182         //Read the data
183         int bytesPerRow = ((width * 3 + 3) / 4) * 4 - (width * 3 % 4);
184         int size = bytesPerRow * height;
185         auto_array<char> pixels(new char[size]);
186         input.seekg(dataOffset, ios_base::beg);
187         input.read(pixels.get(), size);
188
189         //Get the data into the right format
190         auto_array<char> pixels2(new char[width * height * 3]);
191         for(int y = 0; y < height; y++) {
192             for(int x = 0; x < width; x++) {
193                 for(int c = 0; c < 3; c++) {
```

```
194                     pixels2[3 * (width * y + x) + c] =
195                         pixels[bytesPerRow * y + 3 * x + (2 - c)];
196                 }
197             }
198         }
199 
200         input.close();
201         return new Image(pixels2.release(), width, height);
202     }
203 }
```

B.2.6 Defines

```
 1 #include "defines.h"
 2 
 3 float wScreen = SCREEN_WIDTH;
 4 float hScreen = SCREEN_HEIGHT;
 5 
 6 bool menuPrincipal = false;
 7 int status = 0;
 8 bool gameOver = false;
 9 GLuint wallTexture;
10 GLuint floorTexture;
11 
12 //sounds
13 int SOUND_main = -1;
14 int SOUND_inter1 = -1;
15 int SOUND_inter2 = -1;
16 int SOUND_inter3 = -1;
17 int SOUND_attack = -1;
18 int SFX_die = -1;
19 int SFX_eat = -1;
20 int SFX_eat2 = -1;
21 int SFX_alert = -1;
22 //gameplay
23 int attack_mode = 0;
```

B.2.7 Eventos

```
 1 #include "eventos.h"
 2 
 3 #include "gamemanager.h"
 4 
 5 #include "player.h"
 6 
 7 void teclasNormais(unsigned char key, int x, int y)
 8 {
 9             if(key==GLUT_KEY_ESC)
10                 exit(0);
11 
12     if (menuPrincipal)
13         return; /// IGNORA ABAIXO
14 
15     int mod = glutGetModifiers();
16     if (mod == GLUT_ACTIVE_SHIFT)
17         Player::PlayerControl->setCorrer();
18     else
19         Player::PlayerControl->setAndar();
20 
21     switch(key)
22     {
23         case GLUT_KEY_ESC:  //ESC
24             exit(0);
25             break;
26         case 'W':
27         case 'w':
28         {
29             Player::PlayerControl->moveFrente(true);
30             break;
31         }
32         case 'S':
33         case 's':
34         {
35 
36             Player::PlayerControl->moveTraz(true);
37             break;
38         }
39 
40         case 'A':
41         case 'a':
42             Player::PlayerControl->moveEsquerda(true);
43             break;
44         case 'D':
45         case 'd':
46             Player::PlayerControl->moveDireita(true);
47             break;
```

```
48          case 'Q':
49          case 'q':
50              Player::PlayerControl->giraEsquerda(true);
51              break;
52          case 'E':
53          case 'e':
54              Player::PlayerControl->giraDireita(true);
55              break;
56          case '2':
57              Player::PlayerControl->giraCima(true);
58              break;
59          case '3':
60              Player::PlayerControl->giraBaixo(true);
61              break;
62          case '1': // reseta angulo Y
63              Camera::CameraControl.angleY = 0;
64              Camera::CameraControl.calculaDirecao();
65              break;
66          case 'Z':
67          case 'z':
68              Camera::CameraControl.cameraY += 2;
69              break;
70          case 'X':
71          case 'x':
72              Camera::CameraControl.cameraY -= 2;
73              break;
74          case 'C':
75          case 'c':
76              Camera::CameraControl.cameraX = 6;
77              break;
78          case 'V':
79          case 'v':
80              Camera::CameraControl.cameraY = 3;
81              break;
82          case 'B':
83          case 'b':
84              Camera::CameraControl.cameraZ = 6;
85              break;
86          case 'F':
87          case 'f':
88          {
89              GLboolean isFog = false;
90              glGetBooleanv(GL_FOG, &isFog);
91              if (isFog)
92                  glDisable(GL_FOG);
93              else
94                  glEnable(GL_FOG);
95
96              break;
97
98          }
99          case 'R':
100         case 'r':
101             if (FrameRate::FPSControl.isFPSCap())
102                 FrameRate::FPSControl.setFPSCap(false);
103             else
104                 FrameRate::FPSControl.setFPSCap(true);
105             break;
106         default:break;
107     }
108 }
109 void teclasNormaisUp(unsigned char key, int x, int y)
110 {
111     if(key==GLUT_KEY_ESC)
112         exit(0);
113
114     if (menuPrincipal)
115         return; /// IGNORA ABAIXO
116
117     switch(key)
118     {
119         case GLUT_KEY_ESC:  //ESC
120             exit(0);
121             break;
122         case 'W':
123         case 'w':
124             Player::PlayerControl->moveFrente(false);
125             break;
126         case 'S':
127         case 's':
128             Player::PlayerControl->moveTraz(false);
129             break;
130         case 'A':
131         case 'a':
132             Player::PlayerControl->moveEsquerda(false);
133             break;
```

```
134         case 'D':
135         case 'd':
136             Player::PlayerControl->moveDireita(false);
137             break;
138         case 'Q': case 'q':
139             Player::PlayerControl->giraEsquerda(false);
140             break;
141         case 'E': case 'e':
142             Player::PlayerControl->giraDireita(false);
143             break;
144         case '2':
145             Player::PlayerControl->giraCima(false);
146             break;
147         case '3':
148             Player::PlayerControl->giraBaixo(false);
149             break;
150         default:break;
151 
152     }
153 }
154 
155 void teclasEspeciais(int key, int x, int y )
156 {
157     if(key==GLUT_KEY_ESC)
158         exit(0);
159     if (menuPrincipal)
160         return; /// IGNORA ABAIXO
161 
162     switch(key)
163     {
164         case GLUT_KEY_ESC:  //ESC
165             exit(0);
166             break;
167         case GLUT_KEY_UP: Player::PlayerControl->moveFrente(true); break;
168         case GLUT_KEY_DOWN: Player::PlayerControl->moveTraz(true); break;
169         case GLUT_KEY_LEFT: Player::PlayerControl->giraEsquerda(true); break;
170         case GLUT_KEY_RIGHT: Player::PlayerControl->giraDireita(true); break;
171         default: break;
172     }
173 
174 
175 }
176 
177 void teclasEspeciaisSoltar(int key, int x, int y)
178 {
179     if(key==GLUT_KEY_ESC)
180         exit(0);
181 
182     if (menuPrincipal)
183         return; /// IGNORA ABAIXO
184 
185     switch(key)
186     {
187         case GLUT_KEY_ESC:  //ESC
188             exit(0);
189             break;
190         case GLUT_KEY_UP: Player::PlayerControl->moveFrente(false); break;
191         case GLUT_KEY_DOWN: Player::PlayerControl->moveTraz(false); break;
192         case GLUT_KEY_LEFT: Player::PlayerControl->giraEsquerda(false); break;
193         case GLUT_KEY_RIGHT: Player::PlayerControl->giraDireita(false); break;
194         default: break;
195     }
196 }
197 
198 void mouseButton(int button, int state, int x, int y)
199 {
200     if (menuPrincipal)
201     {
202         for(unsigned int i = 0; i < Button::ButtonList.size();i++)
203             Button::ButtonList[i]->handleMouse(button, state, x, y);
204         return; /// IGNORA ABAIXO
205     }
206 
207     if (button == GLUT_LEFT_BUTTON)
208     {
209         if (state == GLUT_UP) //Reseta posicoes e ajusta deslocamento
210         {
211             Player::PlayerControl->setMouse(-1,-1);
212         }
213         else
214         {
215             Player::PlayerControl->setMouse(x,y);
216         }
217     }
218 }
219 
```

```
220void moveMouse(int x, int y)
221{
222    if (menuPrincipal)
223        return; /// IGNORA ABAIXO
224
225    Player::PlayerControl->moveMouse(x,y);
226}
```

B.2.8 Game Maneger

```
 1#include "gamemanager.h"
 2#include "eventos.h"
 3#include <time.h>
 4GameManager game;
 5
 6void startButtonAction()
 7{
 8    menuPrincipal = false;
 9
10    game.resetPositions();
11
12    SoundAL sc;
13    sc.stopAll();
14    sc.play(SOUND_inter2);
15}
16void changeSize(int w, int h)
17{
18    //Prevents division by zero
19    if ( h == 0)
20        h = 1;
21
22    float ratio = w*1.0 / h;
23
24    //Uses projection matrix
25    glMatrixMode(GL_PROJECTION);
26    //Reseta matriz
27    glLoadIdentity();
28
29    //Arranges viewport to entire window
30    glViewport(0,0,w,h);
31
32    //Arranges the right perspective
33    gluPerspective(45.0f, ratio, 1, GAME_FOV*TAMANHO_BLOCO);
34
35    //Back to modelView
36    glMatrixMode(GL_MODELVIEW);
37
38    wScreen = w;
39    hScreen = h;
40}
41void GameManager::inicializaRender(void)
42{
43    //transparency
44    glBlendFunc(GL_SRC_ALPHA,GL_ONE);
45
46    glEnable(GL_LIGHTING); //enables light
47    glEnable(GL_LIGHT0); //enables light #0
48    glEnable(GL_LIGHT1); //enables lightz #0
49    glEnable(GL_NORMALIZE); //Automatically normalize normals
50    glEnable(GL_COLOR_MATERIAL);
51
52    glEnable(GL_DEPTH_TEST);
53    glShadeModel(GL_SMOOTH); //Shading
54
55    glEnable(GL_CULL_FACE); //Reduces the amount of triangles drawn.
56    glCullFace(GL_CW);
57
58    wallTexture = texture::loadTextureBMP("data/wall.bmp");
59    floorTexture = texture::loadTextureBMP("data/floor.bmp");
60
61
62}
63void GameManager::inicializa(void)
64{
65    inicializaRender();
66    inicializaSons();
67
68//---Testes
69
70    coin.Load((char*)"suzane.obj");
71
72    Map::MapControl.coin.Load((char*)"suzane.obj");
73    Map::MapControl.bigCoin.Load((char*)"suzane.obj");
74//---------------------------
75    //Specifies the background color
76    glClearColor(0.3f,0.3f,0.9f,1.0f);
```

```
77
78      GLfloat fog_color[4] = {0.0f,0.0f,0.0f,1.0};
79      glFogfv(GL_FOG_COLOR, fog_color);
80      glFogf(GL_FOG_DENSITY, 0.35f);
81
82      glFogi(GL_FOG_MODE, GL_LINEAR);
83      glHint(GL_FOG_HINT, GL_DONT_CARE);
84      glFogf(GL_FOG_START, TAMANHO_BLOCO*4.0f);
85      glFogf(GL_FOG_END, TAMANHO_BLOCO*10.0f);
86      glEnable(GL_FOG);
87
88      //Tests menu
89      menuPrincipal = true;
90
91      Button* start = new Button();
92
93      start->setXY(220, 200);
94      start->setEstados(1, 350, 60, 0);
95
96      start->ClickAction = startButtonAction;
97
98      Button::ButtonList.push_back(start);
99
100     for(unsigned int i = 0; i < MAX_ENEMY; i++) {
101         enemy[i] = new Entidade();
102         enemy[i]->addToEntidadeList();
103         enemy[i]->setTamanho(5);
104         enemy[i]->createModel((char*)"suzane.obj");
105     }
106
107     Player::PlayerControl = new Player();
108     Player::PlayerControl->addToEntidadeList();
109
110 }
111
112 void GameManager::inicializaSons(void)
113 {
114     sc.init();
115
116     SOUND_main = sc.loadSound("data/mus/main.wav", 1);
117     SOUND_inter1 = sc.loadSound("data/mus/M1.WAV", 1);  //Linux & MAC are sensitive case
118     SOUND_inter2 = sc.loadSound("data/mus/M2.WAV", 1);
119     SOUND_inter3 = sc.loadSound("data/mus/M3.WAV", 1);
120     SOUND_attack = sc.loadSound("data/mus/atk.wav", 1);
121
122     SFX_die = sc.loadSound("data/sfx/die.wav", 0);
123     SFX_eat = sc.loadSound("data/sfx/eat.wav", 0);
124     SFX_eat2 = sc.loadSound("data/sfx/eat2.wav", 0);
125     SFX_alert = sc.loadSound("data/sfx/alert.wav", 0);
126
127
128     sc.play(SOUND_inter1);
129
130
131 }
132 void GameManager::resetPositions(void)
133 {
134     printf("Posicoes resetadas: %lu\n", Entidade::EntidadeList.size());
135
136     Map::MapControl.load((char*) "map_pacman_new.txt");
137
138     srand( time(NULL) );
139
140     for(int i = 0; i < MAX_ENEMY; i++) {
141         enemy[i]->setRandomPosition();
142     }
143
144     Player::PlayerControl->init();
145     Player::PlayerControl->resetPosition();
146 }
147
148 void GameManager::Testes()
149 {
150     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
151
152     glMatrixMode(GL_MODELVIEW);
153     glLoadIdentity();
154
155     glPushMatrix();
156         glColor3f(1.0f, 1.0f, 1.0f);
157         glTranslatef(2.0f, 0.0f, -10.0f);
158         glRotated(120, 0,1,0);
159         //glutSolidSphere(10.0f, 18.0f, 18.0f);
160         glDisable(GL_CULL_FACE);
161         coin.Draw();
162         glEnable(GL_CULL_FACE);
```

```
163     glPopMatrix();
164 }
165 void desenhaTela(void)
166 {
167
168     game.render();
169
170     //game.Testes();
171
172     glutSwapBuffers();
173 }
174
175 void GameManager::loop(void)
176 {
177
178     FrameRate::FPSControl.loop();
179     for(unsigned int i = 0; i < Entidade::EntidadeList.size(); i++)
180     {
181         Entidade::EntidadeList[i]->loop();
182     }
183     for(unsigned int i = 0; i < Entidade::EntidadeList.size(); i++)
184     {
185         Entidade::EntidadeList[i]->testaColisao();
186     }
187     for(unsigned int i = 0; i < Entidade::EntidadeList.size(); i++)
188     {
189         Entidade::EntidadeList[i]->executaColisao();
190     }
191
192
193     //Verifies change of states on the special ball
194     if(attack_mode == 1) //notified change and play music
195     {
196         //Ste SPECIAL flag active for all entities. Even the player
197         for(unsigned int i = 0; i < Entidade::EntidadeList.size(); i++)
198         {
199             Entidade::EntidadeList[i]->flags = ENTIDADE_FLAG_ESPECIAL;
200         }
201         Player::PlayerControl->flags = ENTIDADE_FLAG_PLAYER_ESPECIAL; // resets the player's flag
202         ticksAttack = glutGet(GLUT_ELAPSED_TIME);
203         sc.stopAll();
204         sc.play(SFX_alert);
205         attack_mode = 2;
206     } else
207     if (attack_mode == 2)
208     {
209         //after 3 seconds
210         if( (glutGet(GLUT_ELAPSED_TIME) - ticksAttack) > 3000 )
211         {
212             sc.stopAll();
213             sc.play(SOUND_attack);
214             attack_mode = 3;
215             ticksAttack = glutGet(GLUT_ELAPSED_TIME);
216         }
217     } else
218     if (attack_mode == 3)
219     {
220         //over the end of the ball efects 10 seconds + 3 the preceding sfx
221         if( (glutGet(GLUT_ELAPSED_TIME) - ticksAttack) > 10000)
222         {
223             sc.stopAll();
224             sc.play(SOUND_inter2);
225             attack_mode = 0;
226             for(unsigned int i = 0; i < Entidade::EntidadeList.size(); i++)
227             {
228                 Entidade::EntidadeList[i]->flags = ENTIDADE_FLAG_NENHUM;
229             }
230             Player::PlayerControl->flags = ENTIDADE_FLAG_PLAYER_NORMAL; // resets the player's flag
231         }
232     }
233
234 }
235 void GameManager::render(void)
236 {
237
238     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
239
240     glMatrixMode(GL_MODELVIEW);
241     glLoadIdentity();
242
243     if (menuPrincipal)
244     {
245         for(unsigned int i = 0; i < Button::ButtonList.size();i++)
246             Button::ButtonList[i]->render();
247
248         txt::renderText2dOrtho(30,150,8,"Aperte o grande quadrado branco para comecar!!!");
```

```
249
250         switch(status)
251         {
252             case STATUS_DERROTA:
253                 txt::renderText2dOrtho(30,130,8,"Derrota!!!");
254                 break;
255             case STATUS_NORMAL:
256                 txt::renderText2dOrtho(30,130,8,"Novo jogo!!!");
257                 break;
258             case STATUS_VITORIA:
259                 txt::renderText2dOrtho(30,130,8,"Vitoria!!!");
260                 break;
261                 default:;
262         }
263
264         return;
265     }
266
267
268
269
270     //Lighting
271     GLfloat ambientLight[] = {0.1f, 0.1f, 0.1f, 1.0f};
272     glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientLight);
273     GLfloat directedLight[] = {0.7f, 0.7f, 0.7f, 0.0f};
274     GLfloat directedLightPos[] = {0.0f, 20.0f, -20.0f, 1.0f};
275     GLfloat light[] = {0.9f, 0.9f, 0.9f, 1.0f};
276     GLfloat lightPos[] = {100.0f, 30.0f, -10.0f, 1.0f};
277     glLightfv(GL_LIGHT0, GL_DIFFUSE, directedLight);
278     glLightfv(GL_LIGHT0, GL_POSITION, directedLightPos);
279     glLightfv(GL_LIGHT1, GL_DIFFUSE, light);
280     glLightfv(GL_LIGHT1, GL_POSITION, lightPos);
281     //end of lighting
282
283
284     //calculates iterations
285     this->loop();
286
287     //Print SOL's
288     glPushMatrix();
289         glColor3f(1.0f, 1.0f, 1.0f);
290         glTranslatef(directedLightPos[0],directedLightPos[1],directedLightPos[2]);
291         glutSolidSphere(10.0f, 18.0f, 18.0f);
292     glPopMatrix();
293     glPushMatrix();
294         glColor3f(1.0f, 0.0f, 0.0f);
295         glTranslatef(lightPos[0],lightPos[1],lightPos[2]);
296         glutSolidSphere(10.0f, 18.0f, 18.0f);
297     glPopMatrix();
298
299     Map::MapControl.render();
300     //unsigned int temp = Entidade::EntidadeList.size();
301     for(unsigned int i = 0; i < Entidade::EntidadeList.size(); i++)
302     {
303         if (Entidade::EntidadeList[i]->isVisible())
304             Entidade::EntidadeList[i]->render();
305     }
306
307     txt::renderText2dOrtho(10,15,0,"FPS: %.2f",FrameRate::FPSControl.getFPS());
308
309
310
311
312     MiniMap::renderMiniMap();
313
314}
315
316
317// when called during cleanup destructor,
318// segmentation fault occurs only delete the Entity
319GameManager::~GameManager()
320{
321     sc.stopAll();
322     sc.exit();
323     coin.Release();
324     Map::MapControl.coin.Release();
325     Map::MapControl.bigCoin.Release();
326}
327void cleanup(void)
328{
329     unsigned int sizeEnt = Entidade::EntidadeList.size();
330     unsigned int sizeBtn = Button::ButtonList.size();
331     printf("Entidade cleanup size: %u\n", sizeEnt);
332     for(unsigned int i = 0; i < sizeEnt; i++)
333     {
334         delete Entidade::EntidadeList[i];
```

```
335     }
336
337     printf("Button cleanup size: %u\n", sizeBtn);
338     for(unsigned int i = 0; i < sizeBtn; i++)
339         delete Button::ButtonList[i];
340     printf("EXIT\n");
341 }
342 void testOpenAL()
343 {
344     unsigned int g_buf = -1;
345     unsigned int g_src = -1;
346
347     if(!alutInit(NULL, NULL))
348     {
349         printf("%s",alutGetErrorString(alutGetError()));
350         return;
351     }
352     alGetError();
353     alutGetError();
354
355     g_buf = alutCreateBufferFromFile("testing.wav");
356
357     if (alutGetError() != ALUT_ERROR_NO_ERROR)
358      {
359          alDeleteBuffers(1, &g_buf);
360          alutExit();
361          return;
362      }
363
364      alGenSources(1, &g_src);
365
366      if(alGetError() != AL_NO_ERROR)
367      {
368          alDeleteBuffers(1, &g_buf);
369          alDeleteSources(1, &g_buf);
370          alutExit();
371          return;
372      }
373
374      alSourcei(g_src, AL_BUFFER, g_buf);
375
376      alSourcePlay(g_src);
377      alutSleep(4.0f);
378
379      alutExit();
380 }
381 void testSoundALClass()
382 {
383     SoundAL sn;
384     sn.init();
385
386     int m_i = sn.loadSound("testing.wav", 1);
387     sn.play(m_i);
388
389     alutSleep(4.0f);
390
391     sn.exit();
392 }
393 int main(int argc, char* args[])
394 {
395
396     //testOpenAL();
397     //testSoundALClass();
398
399     game.executa(argc, args);
400     return 0;
401 }
402 void GameManager::executa(int argc, char* args[])
403 {
404     glutInit(&argc, args);
405     glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
406     glutInitWindowPosition(100,100);
407     glutInitWindowSize(SCREEN_WIDTH, SCREEN_HEIGHT);
408     glutCreateWindow("Labirinth");
409
410     inicializa();
411
412     glutDisplayFunc(desenhaTela);
413     glutReshapeFunc(changeSize);
414     glutIdleFunc(desenhaTela);
415
416     glutKeyboardFunc(teclasNormais);
417     glutKeyboardUpFunc(teclasNormaisUp);
418     glutSpecialFunc(teclasEspeciais);
419     glutSpecialUpFunc(teclasEspeciaisSoltar);
420     glutMotionFunc(moveMouse);
```

```
421     glutMouseFunc(mouseButton);
422
423     atexit(cleanup);
424
425     glutIgnoreKeyRepeat(0);
426     //Get in the loop processing events
427     glutMainLoop();
428}
```

B.2.9 Text

```
 1#include "text.h"
 2
 3namespace txt
 4{
 5     void renderBitmapString(
 6             float x,
 7             float y,
 8             int spacing,
 9             void *font,
10             char *string) {
11
12       char *c;
13       int x1 = x; //Guarda posicao rasterizada para computar espaco
14
15       for (c=string; *c != '\0'; c++) {
16         glRasterPos2d(x1,y);
17         glutBitmapCharacter(font, *c);
18         x1 = x1 + glutBitmapWidth(font, *c) + spacing;
19       }
20     }
21
22     void* font_glut = GLUT_BITMAP_8_BY_13;
23
24     ///ARRUMA PROJECOES
25     extern void setProjecaoOrto()
26     {
27         glDisable(GL_DEPTH_TEST);
28         glDisable(GL_LIGHTING);
29         glMatrixMode(GL_PROJECTION);
30         glPushMatrix(); //nao fecha
31         glLoadIdentity();
32
33         // coloca projecao ortografica 2d
34         gluOrtho2D(0, wScreen, hScreen, 0);
35         glMatrixMode(GL_MODELVIEW);
36
37         glPushMatrix();
38         glLoadIdentity();
39     }
40     extern void restauraProjecaoPerspectiva()
41     {
42         glPopMatrix();
43         glMatrixMode(GL_PROJECTION);
44         glPopMatrix(); // fecha o pushMatrix do projecaoOrtho
45         glEnable(GL_DEPTH_TEST);
46         glEnable(GL_LIGHTING);
47         glMatrixMode(GL_MODELVIEW);
48     }
49
50     extern void renderText2dOrtho(float x, float y, int spacing, const char*pStr, ...)
51     {
52         char string[128];
53         va_list valist; //info das variaveis
54         va_start(valist, pStr); //inicia lista de argumentos das variaveis
55         vsprintf(string, pStr, valist); // joga string formatado para string
56         va_end(valist); // realiza operacoes de fato
57
58         glDisable(GL_LIGHTING);
59         setProjecaoOrto();
60             renderBitmapString(x,y, spacing, font_glut, string);
61         restauraProjecaoPerspectiva();
62         glEnable(GL_LIGHTING);
63
64     }
65}
```

B.2.10 Title

```
1#include "tile.h"
2
3Tile::Tile()
4{
5     tamanho = TAMANHO_BLOCO;
6     posY = 0;
7
```

```
    left = right = front = back = top = bottom = false;
}
```

B.2.11 Makefile

```
#############################################################
#                       Makefile
#                Friday 17 August 2012
#############################################################
CC = g++
CFLAGS =$(GLFLAGS) -I./ -O3 -Os -g $(PROBLENS)
CC_WINDOWS = x86_64-linux-gnu-g++

PROBLENS=-Wall -pedantic -fpermissive
UNAME = $(shell uname)
OUTPUT = Amaze.out

SRC =   button.cpp  \
        defines.cpp   \
        eventos.cpp    \
        minimap.cpp    \
        player.cpp   \
        text.cpp           \
        tile.cpp \
        camera.cpp  \
        entidade.cpp  \
        framerate.cpp  \
        map.cpp          \
        model_obj.cpp  \
        soundAL.cpp  \
        textureloader.cpp

OBJS = ${SRC:.cpp=.o}

.SUFFIXES:.c.o
###
#   libghc-opengl-dev
#   libghc-openal-dev
#   freeglut3-dev
#   libglui-dev
#   libalut-dev
#   glee-dev
####


define PROGRAM_template
$(1): $(addsuffix .o,$(1))
endef
$(foreach t,$(compiling),$(eval $(call PROGRAM_template,$(t))))


ifeq ($(UNAME),Linux) # Linux OS
    GLFLAGS = -lglut -lglui -lGLU -lGL -lalut  -lopenal
    SEARCH = dpkg -l | grep -iq
    else
    ifeq ($(UNAME),Darwin) # MAC OS X
        GLFLAGS = -framework OpenGL -framework GLUT -framework OpenAL
        SEARCH = ls /System/Library/Frameworks | grep -i
    else #Windows
        GLFLAGS = -lopengl32 -lglu32 -lglut32 -lglee -lalut
        SEARCH=
    endif
endif

all: system out

debug: config

system:
    echo "System: "$(UNAME) "OS"
    echo -n "Compiling..."

system_debug:
    echo "System: "$(UNAME) "OS"
    if rm *.o;\
    then \
        echo -n "Cleaning && ";\
    fi;
    echo -n "Compiling..."

config: system_debug
    if $(MAKE) out ;\
    then \
        echo " " ;\
    else \
```

```
82      echo  "Error on compiling! Probably some package is missing"; \
83          $(MAKE) check;\
84      fi;
85
86  .c.o:
87      echo -n "compiling..." $<
88      $(CC) $< -c -g $(CFLAGS) $(GLFLAGS)
89      echo  "Done"
90
91  compiling: $(OBJS)
92
93  lib: compiling
94      echo  "Done"
95      echo -n "Making lib..."
96      ar rcs libAmaze.a *.o
97
98  out: lib
99      $(CC) gamemanager.cpp -o $(OUTPUT) $(CFLAGS) -L./ -lAmaze $(GLFLAGS)
100     echo "Done.\nRun "$(OUTPUT) ;\
101
102 clean:
103     echo "Cleaning all..."
104     rm -rfv $(OUTPUT) *.o *.d *.a
105
106 run: out
107     echo "Running..."
108     ./$(OUTPUT)
109
110 valgrind: *.cpp
111     $(CC) -g -c $(SRC) $(CFLAGS) $(GLFLAGS)
112     ar rc libAmaze.a *.o
113     $(CC) -g gamemanager.cpp -o ToGring $(GLFLAGS) -L./ -lAmaze
114 #   valgrind --tool=callgrind --dsymutil=yes --trace-jump=yes ./ToGring -q --fullpath-after=string
    --show-possibly-lost=yes --trace-children=yes -v --main-stacksize=512MB
115     echo "Valgrind files available: (newer first)"
116     ls -t| egrep -i grind
117
118 windows: *.cpp
119     echo "Cross compiling to" $@
120     $(CC_WINDOWS) *.cpp -c $(CFLAGS)
121     $(CC_WINDOWS) *.o -o ./bin/x86-x64-Amaze.exe $(CFLAGS)
122     echo "done.\nRun " x86-x64-Amaze.exe "on bin directory"
123
124 check:
125     echo "Checking if all dev packages are installed"
126 #   OPENGL
127     echo -n "opengl "
128     if $(SEARCH) "opengl.*dev" ;\
129     then \
130         echo "[OK]";\
131     else \
132         echo "[MISSING!] - Install libghc-opengl-dev" ;\
133     fi;
134 #   OPENAL
135     echo -n "openal "
136     if $(SEARCH) "openal.*dev" ;\
137     then \
138         echo "[OK]";\
139     else \
140         echo "[MISSING!] - Install libghc-openal-dev" ;\
141     fi;
142 #   GLUT
143     echo -n "glut "
144     if $(SEARCH)  "glut.*dev" ;\
145     then \
146         echo "[OK]";\
147     else \
148         echo "[MISSING!] - Install freeglut3-dev" ;\
149     fi;
150 #   GLUI
151     echo -n "glui "
152     if $(SEARCH)  "glui.*dev" ;\
153     then \
154         echo "[OK]";\
155     else \
156         echo "[MISSING!] - Install libglui-dev" ;\
157     fi;
158 #   ALUT
159     echo -n "alut "
160 #Como deveria de ser pra buscar por suporte para desenvolvedores
161 #   if $(SEARCH) | grep -qi "alut.*dev" ;\
162     if $(SEARCH)  "alut.*dev" ;\
163     then \
164         echo "[OK]";\
165     else \
166         echo "[MISSING!] - Install libalut-dev" ;\
```

```
167     fi;
168#    GLEE
169     echo -n "glee "
170     if $(SEARCH)  "glee.*dev" ;\
171     then \
172         echo "[OK]";\
173     else \
174         echo "[MISSING!] - Install glee-dev" ;\
175     fi;
176
177.SILENT:
178
179#Obs
180#
181#    Bibliotecas incluidas:
182#
183#    alut-dev
184#    openal-dev
185#
186#    Descobrindo pacotes instalados:
187# $ dpkg -l | grep alut
188#
189#    No MacOS os Frameworks ficam no diretorio/System/Library/Frameworks
190# e possuem a nomeclatura semelhante a:
191# OpenAL.framework
```

B.2.12 README

```
h1. README

# Windows

    The program was developed with the assistance of CodeBlocks IDE. To generate the executable on the platform,
just open the project file - Labirinto.cbp in CodeBlocks and have compile / build the project. In the IDE will
own the means of implementing the output file, but the project folder you can also locate the *.exe.

 # Mac OS

    -Similar to the steps on the Linux system, the user must run the command "make run" in the directory containing the makefil

    Not supported yet.

# Linux

    To build the program on the Linux platform, you need some libraries installed on your system. Among them is
valid highlight of OpenGL and audio (ALUT and OpenAL). In the folder where the source files, you can find the
makefile. In the terminal, just run the command "make run" in the directory containing the makefile to compile
the files and start the program correctly. If any of the required libraries are not installed, it will be seen
the list of warnings/errors, guiding which library should be installed. It is valid to remember that to install
the libraries for this purpose on the Linux platform, you should seek the names with the suffix "-dev", thereby
ensuring that the necessary files will be installed. The compilation will be done on silent mode.

* Example of compiling

    $ make
    System: Linux OS
    Compiling...ok
    Cleaning...done.
    Run Amaze.out

    $ make check
    Checking if all dev packages are installed
    opengl [OK]
    openal [OK]
    glut [OK]
    glui [OK]
    alut [OK]
    glee [OK]
```