# 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*.

### 1.2 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.

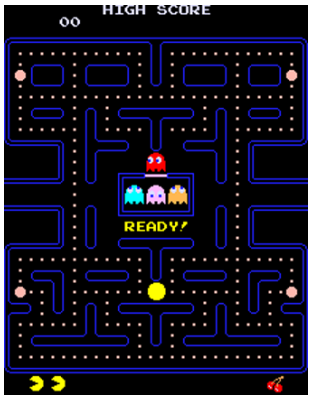


**Figura 1:** Pac-Man.
O clássico dos anos 80 só foi ter um score perfeito - máximo de pontos, sem falhas ou mortes - em 1999, quando *Billy Mitchell* consegui a incrível marca de 3,333,360 pontos, após vencer os consecutivos 256 leveis do jogo.

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 -lglee -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 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
```

### 2.1.3 Artefatos

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

- button.cpp
- camera.h
- entidade.cpp

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

2. Atualizado em 7 de Junho de 2012

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

2.1.3.2 **README**: O arquivo README pode ser localizado dentre os arquivos fontes, em A.2.12.

### *2.1.4 Problemas Técnicos*

No decorrer da construção do programa a maior dificuldade foi ...

**TODO :**
**VERIFICAR ISSO**
**SEGUNDO a professora:**
Na seção desenvolvimento deve ser respondidas as seguintes perguntas:

- Como os pontos relacionados à disciplina foram abordados no problema? Quais as lições aprendidas? Quais as principais dificuldades?
- Quais elementos teóricos abordado na disciplina foram implementados no programa?
- Quais adaptações, extensões, bibliotecas externas, foram necessários para a solução do problema?
- Caso use parte de códigos disponibilizados na Web, colocar referência [3]

## 3 Caso de Teste

Nessa seção deve ser apresentado pelo menos um exemplo de caso de teste. Se não for especificado na descrição do problema, ela deve definida, explicada e ilustrada pelos autores.

3. A home-page de onde tirei este material:http://en.wikibooks.org/wiki/LaTeX.Estou formatando para LaTeXapenas para os estudantes irem se orientando de como e o quê escrever.Assim, me isento de responsabilidade sobre o conteúdo deste texto. Dúvidas: carla(rocha.carla@gmail.com)

## 4 Conclusão

Discutir os principais pontos relativos ao desenvolvimento do programa:

- Dificuldades encontradas em atingir os objetivos propostos. Caso não tenha sido possível, concluir 100% da tarefa, listar razões para tal.
- Sugestões de melhorias do programa.
- Pontos teóricos mais relevantes abordados na prática e a relevância de tais conceitos (Exemplo de aplicações que tais conceitos seriam úteis). Com citações se necessário.

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


**Gustavo Jaruga Cruz**
Matricula: 09/0066634
E-mail: darksshades@hotmail.com

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

# APÊNDICE A
# CÓDIGOS FONTES

## A.1 Headers

### *A.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
```

### *A.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"

```

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

### A.1.3 Framerate

```
1#ifndef __FRAMERATE_H_
2#define __FRAMERATE_H_
3
4#include "defines.h"
5
6
7class FrameRate
8{
9    private:
10        unsigned int ticks;
11        unsigned int ticksControl;
12        unsigned int frames;
```

```
13        float fps;
14    public:
15        void loop();
16
17        bool fpsCap;
18
19        void setFPSCap(bool cap);
20        bool isFPSCap();
21        float getFPS();
22        FrameRate();
23
24        void regulaFPS();
25
26        static FrameRate FPSControl;
27};
28
29
30#endif
```

### A.1.4 Map

```
 1#ifndef _MAPS_H_
 2#define _MAPS_H_
 3
 4#include "defines.h"
 5#include "tile.h"
 6#include "camera.h"
 7#include "text.h"
 8#include <vector>
 9#include <stdio.h>
10#include <math.h>
11
12
13class Map
14{
15    private:
16        std::vector<Tile> listaTiles;
17        std::vector<Tile> listaTilesOptimizados;
18        void geraQuadradosOptimizados();
19
20        int RENDER_MODE;
21
22
23        //void renderTile(unsigned int i);
24        void renderTileOptimizado(unsigned int i);
25        void renderBloco(float width, float height, float flatness, bool left,
26                         bool right, bool front, bool back, bool top, int TYPE);
27
28
29        bool mostraWired;
30    public:
31        Tile* getTile(int x, int y);
32        inline int getX(int i);
33        inline int getY(int i);
34        int MAP_HEIGHT;
35        int MAP_WIDTH;
36
37        float origemX; // Where the map start to render
38        float origemZ; //Tile 0,0, growls on right-down
39
40        void setWired(int wired);
41        bool isWire();
42
43        Map();
44
45        //void render();
46        void render();
47        int load(char* filename);
48
49        //void iniciaDisplayList();
50        GLuint dlMap;
51
52        //Used to others classes to get info about the map
53        static Map MapControl;
54
55
56
57        //Operator overload
58        inline Tile* operator ()  (const int x, const int y)
59        {
60            return this->getTile(x,y);
61        }
62
63
64
65};
```

```
66
67
68#endif
```

### A.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}
```

### A.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 GAME_FOV                28
50
51#define PONTOS_BOLA             10
52#define PONTOS_BOLA_ESPECIAL    50
53
54#define TAMANHO_INIMIGO         5
55
56
57
58//Size of the current screen
59extern float wScreen;
60extern float hScreen;
61//textures
62extern GLuint wallTexture;
63extern GLuint floorTexture;
64//Menu
65extern bool menuPrincipal;
66extern int status;
67
68//Sounds
69extern int SOUND_main;
70extern int SOUND_inter1;
71extern int SOUND_inter2;
72extern int SOUND_inter3;
73extern int SOUND_attack;
74extern int SFX_die;
75extern int SFX_eat;
76extern int SFX_eat2;
77extern int SFX_alert;
78//Global from gameplay
79extern int attack_mode;
80
81#define STATUS_NORMAL 0
82#define STATUS_VITORIA 1
83#define STATUS_DERROTA 2
84
85
86
87#endif
```

### A.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
```

### A.1.8 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
```

```
}


#endif
```

## A.2 Sources

### *A.2.1 Camera*

```
#include "camera.h"

#include <math.h>
Camera Camera::CameraControl;
Camera::Camera()
{
    angleX = 90.0f;
    angleY = 0.0f;
    angleOffsetX = angleOffsetY = 0;

    lookX = 0.5f;
    lookY = 0.0f;
    lookZ = -1.0f;

    cameraX = (TAMANHO_BLOCO*1) + TAMANHO_BLOCO/2;
    cameraY = 5.0f;
    cameraZ = (TAMANHO_BLOCO*1) + TAMANHO_BLOCO/2;
    //tests

    //tests
    deltaAngleX = deltaAngleY = 0.0f; //Angle of rotation of the horizontal and vertical direction

    deltaMouseX = deltaMouseY = 0.0f;

    deltaMove = deltaMoveLado = 0.0f;


    velocidadeMoveAndar = CAMERA_ANDA;
    velocidadeMoveCorre = CAMERA_CORRE;
    velocidadeMove = velocidadeMoveAndar;
    velocidadeVira = 45.f;
    velocidadeViraMouse = 0.1f;

    xOrigem = -1;
    yOrigem = -1;
    ticks = 0;

    calculaDirecao();
}

void Camera::reset()
{
    angleX = 90.0f;
    angleY = 0.0f;
    angleOffsetX = angleOffsetY = 0;

    lookX = 0.5f;
    lookY = 0.0f;
    lookZ = -1.0f;

    cameraX = (TAMANHO_BLOCO*1) + TAMANHO_BLOCO/2;
    cameraY = 5.0f;
    cameraZ = (TAMANHO_BLOCO*1) + TAMANHO_BLOCO/2;
    //tests

    //tests
    deltaAngleX = deltaAngleY = 0.0f; //Angle of rotation of the horizontal and vertical direction

    deltaMouseX = deltaMouseY = 0.0f;

    deltaMove = deltaMoveLado = 0.0f;


    velocidadeMoveAndar = CAMERA_ANDA;
    velocidadeMoveCorre = CAMERA_CORRE;
    velocidadeMove = velocidadeMoveAndar;
    velocidadeVira = 45.f;
    velocidadeViraMouse = 0.1f;

    xOrigem = -1;
    yOrigem = -1;
    ticks = 0;

    calculaDirecao();
    ticks = glutGet(GLUT_ELAPSED_TIME);
}
```

```
77
78
79//Called internally by Player.
80void 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
93void Camera::loop()
94{
95    deltaTicks = glutGet(GLUT_ELAPSED_TIME) - ticks;
96}
97
98void 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}
122void 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}
131void 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
144void Camera::moveFrente(bool mover)
145{
146    if(mover)
147        deltaMove = velocidadeMove;
148    else
149        deltaMove = 0.0f;
150}
151void Camera::moveTraz(bool mover)
152{
153    if(mover)
154        deltaMove = -velocidadeMove;
155    else
156        deltaMove = 0.0f;
157
158}
159void 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 }
```

### *A.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 void Entidade::cleanup()
43 {
44 }
45 bool Entidade::isColisaoObjeto(Entidade* objeto)
46 {
47     //Note: The point marks position 0 .... ex: position 0 beginning of the block end of the block in the x, y, z
48     //Such that y lower = y ; y highest = y+tamanhoY
49     int baixo1 = this->posicao.y;
50     int cima1 = this->posicao.y + this->tamanho.y;
51     int esquerda1 = this->posicao.x;
52     int direita1 = this->posicao.x + this->tamanho.x;
53     int frente1 = this->posicao.z;
54     int traz1 = this->posicao.z + this->tamanho.z;
55
56     int baixo2 = objeto->posicao.y;
57     int esquerda2 = objeto->posicao.x;
58     int frente2 = objeto->posicao.z;
59     int direita2 = objeto->posicao.x + objeto->tamanho.x;
60     int cima2 = objeto->posicao.y + objeto->tamanho.y;
61     int traz2 = objeto->posicao.z + objeto->tamanho.z;
62
63     if (
64         !(baixo1 > cima2) &&
65         !(cima1  < baixo2) &&
66         !(esquerda1 > direita2) &&
67         !(direita1 < esquerda2) &&
68         !(frente1 > traz2) &&
69         !(traz1 < frente2)
70         )
71         {
72             return true;
73         }
74
75     return false;
76
77 }
78 //=============================================================================
79 // Returns true if colliding with the map
80 //=============================================================================
81 Tile* Entidade::isColisaoMapa(Vetor3D newPosicao, int type)
82 {
83     //Calculates Id tile to be tested
84     //Ex: X = 5 Such that startX = 0,41 = 0 endX = 1,3 = 1
85     int startX = (newPosicao.x) / TAMANHO_BLOCO;
86     int startZ = (newPosicao.z) / TAMANHO_BLOCO;
87     int endX = (newPosicao.x + (tamanho.x)) / TAMANHO_BLOCO;
88     int endZ = (newPosicao.z + (tamanho.z)) / TAMANHO_BLOCO;
89
90     //Check collisions with tiles
91     for(int iZ = startZ; iZ <= endZ; iZ++) {
92         for(int iX = startX; iX <= endX; iX++) {
93             Tile* bloco = Map::MapControl(iX, iZ);
```

```
94
95              if(
96                 (bloco->typeId == type) &&
97                 (posicao.y < (bloco->posY+bloco->tamanho) ) &&
98                 ((posicao.y+tamanho.y) > bloco->posY)
99                 )
100                 return bloco;
101             }
102     }
103     return 0;
104 }
105
106 void Entidade::removeFromEntidadeList()
107 {
108     for(unsigned int i = 0; i < EntidadeList.size(); i++)
109     {
110         if (EntidadeList[i] == this)
111             EntidadeList.erase(EntidadeList.begin()+i);
112     }
113 }
114 void Entidade::addToEntidadeList()
115 {
116
117
118     for(unsigned int i = 0; i < EntidadeList.size(); i++)
119     {
120         if (EntidadeList[i] == this)
121             return; //Se ja estiver na lista, retorna
122     }
123
124     EntidadeList.push_back(this);
125 }
126
127 bool Entidade::carregaModelo(char* file){return true;}
128 //=========================================================================
129 // Performs actions of the loop, acceleration, speed.
130 //=========================================================================
131 void Entidade::loop()
132 {
133     //3 seconds has the spawn
134     if ( (flags == ENTIDADE_FLAG_RESPAWN) && ( (glutGet(GLUT_ELAPSED_TIME) - respawnTicks) > 3000) )
135     {
136         dead = false;
137         visible = true;
138         setRandomPosition();
139         flags = ENTIDADE_FLAG_NENHUM;
140     }
141
142     if(dead) return;
143     //deltaTicks reset the surrender
144     delta = glutGet(GLUT_ELAPSED_TIME) - deltaTicks;
145     float fator = delta/1000.f;
146
147     //calculates accelerations
148     if ( velocidade.x + aceleracao.x <= maxVelocidade.x)
149         velocidade.x += (aceleracao.x * fator);
150     if ( velocidade.y + aceleracao.y <= maxVelocidade.y)
151         velocidade.y += (aceleracao.y * fator);
152     if ( velocidade.z + aceleracao.z <= maxVelocidade.z)
153         velocidade.z += (aceleracao.z * fator);
154
155     Vetor3D newPosicao = posicao + (velocidade * fator );
156
157     if (isColisaoMapa(newPosicao) == false)
158         posicao = newPosicao;
159     else
160     {
161         velocidade.x = 0;
162         velocidade.z = 0;
163         aceleracao.x = 0;
164         aceleracao.z = 0;
165         int pos = (int)(rand() % 4);
166         switch(pos)
167         {
168             case 0:
169                 aceleracao.x = 20;break;
170             case 1:
171                 aceleracao.x = -20;break;
172             case 2:
173                 aceleracao.z = 20;break;
174             case 3:
175                 aceleracao.z = -20;break;
176             default:;
177         }
178
179     }
```

```
180
181     deltaTicks = glutGet(GLUT_ELAPSED_TIME);
182 }
183 void Entidade::render()
184 {
185     if (!isVisible())
186         return;
187
188     int tamanhoCubo = tamanho.x; //Temp while using glutCube
189     glPushMatrix();
190     //Centers due to GLUT
191     if (flags == ENTIDADE_FLAG_ESPECIAL)
192         glColor3f( getColor(1), getColor(2), getColor(3) );
193     else
194         glColor3f(r,g,b);
195     glTranslated(posicao.x+tamanho.x/2,
196                  posicao.y+tamanho.y/2,
197                  posicao.z+tamanho.z/2);
198     if (showWired)
199         glutWireCube(tamanhoCubo);
200     else
201         glutSolidCube(tamanhoCubo);
202     glPopMatrix();
203
204
205 }
206 void Entidade::testaColisao()
207 {
208     if(dead) return;
209
210     unsigned int thisID = -1;
211     for (unsigned int i = 0; i < EntidadeList.size(); i++)
212         if (EntidadeList[i] == this)
213         {
214             thisID = i;
215             break;
216         }
217     //Tests with all the entities of this forward.
218     //Ex:    lista: 1 2 3 4
219     // thisID =1,  tests with 2, 3 , 4
220     // thisID = 2  tests with 3, 4      this way, thisID = 2 no collisions with 1 as has already been tested previously.
221     for (unsigned int i = thisID+1; i < EntidadeList.size(); i++)
222     {
223         if (EntidadeList[i] != this && !EntidadeList[i]->dead)
224         {
225             if(isColisaoObjeto(EntidadeList[i]) )
226             {   //adds this element collisions so as tested in
227                 setColisao(EntidadeList[i]);
228                 EntidadeList[i]->setColisao(this);
229             }
230         }
231     }
232 }
233 //Set collision through the public method
234 void Entidade::setColisao(Entidade* ent)
235 {
236     entidadeColidida.push_back(ent);
237 }
238 bool Entidade::isColidido()
239 {
240     if (entidadeColidida.size() == 0)
241         return false;
242     else
243         return true;
244 }
245 void Entidade::executaColisao()
246 {
247     if ( !isColidido() )
248         return; // no collisions
249
250
251 /*
252
253     //Back what had moved.
254     float fator = delta/1000.f;
255     posicao = posicao - (velocidade * fator );
256     //For, and go in the opposite direction
257     velocidade.x = 0;
258     velocidade.z = 0;
259     aceleracao.x = -aceleracao.x;
260     aceleracao.z = -aceleracao.z;
261 */
262     if ( (flags == ENTIDADE_FLAG_ESPECIAL) && (entidadeColidida[0]->flags == ENTIDADE_FLAG_PLAYER_ESPECIAL) )
263     {
264         flags = ENTIDADE_FLAG_RESPAWN;
265         respawnTicks = glutGet(GLUT_ELAPSED_TIME);
```

```
266         dead = true;
267         visible = false;
268         SoundAL sc;
269         sc.play(SFX_eat2);
270     }
271 
272     entidadeColidida.clear();
273 }
274 
275 void Entidade::setRandomPosition()
276 {
277     bool isOK = false;
278         while(!isOK) {
279             int posX = rand() % Map::MapControl.MAP_WIDTH;
280             int posZ = rand() % Map::MapControl.MAP_HEIGHT;
281 
282             //If the position is different from the wall, then ground .... put cube
283             if (Map::MapControl.getTile(posX, posZ)->typeId != TILE_TIPO_PAREDE) {
284                 //Note: (TAMANHO_BLOCO/2 - tamanho.x/2)  is used to find the center of the floor
285                 posicao.x = (TAMANHO_BLOCO/2 - tamanho.x/2) + TAMANHO_BLOCO*posX;
286                 posicao.y = 0;
287                 posicao.z = (TAMANHO_BLOCO/2 - tamanho.z/2) + TAMANHO_BLOCO*posZ;
288                 //1 to 10
289                 aceleracao.x = 1 + rand() % 10;
290                 aceleracao.z = 1 + rand() % 10;
291                 init();
292                 isOK = true;
293                 ///Possible to add verification that the entity was not in the same place using isColisao and clear() from list
294             }
295         }
296 }
297 
298 bool Entidade::isVisible()
299 {
300     return visible;
301 }
302 void Entidade::setTamanho(float newTamanho)
303 {
304     tamanho.x = tamanho.y = tamanho.z = newTamanho;
305 }
306 void Entidade::setPosicao(float x, float y, float z)
307 {
308     posicao.x = x;
309     posicao.y = y;
310     posicao.z = z;
311 }
312 void Entidade::setColor3f(float fr, float fg, float fb)
313 {
314     r = fr;
315     g = fg;
316     b = fb;
317 }
318 float Entidade::getColor(int rgb_i)
319 {
320     float color = 0.0f;
321     switch(rgb_i)
322     {
323         case 1:
324             color = r;
325             if (flags == ENTIDADE_FLAG_ESPECIAL)
326                 color -= 0.55f;
327             break;
328         case 2:
329             color = g;
330             if (flags == ENTIDADE_FLAG_ESPECIAL)
331                 color += 1;
332             break;
333         case 3:
334             color = b;
335             if (flags == ENTIDADE_FLAG_ESPECIAL)
336                 color += 0.95f;
337             break;
338     }
339     return color;
340 }
```

### A.2.3 Framerate

```
1 #include "framerate.h"
2 
3 
4 FrameRate FrameRate::FPSControl;
5 
6 
7 
8 float 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
29void 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
39void 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}
```

### A.2.4 Map

```
 1#include "map.h"
 2
 3//Used by others classes to get info about the map
 4Map 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
 9Tile* 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}
17inline int Map::getX(int i)
18{
19    return i % MAP_WIDTH;
20}
21inline int Map::getY(int i)
22{
23    return (int) i/MAP_WIDTH;
24}
25
26Map::Map()
27{
28    origemX = -TAMANHO_BLOCO;
29    origemZ = -TAMANHO_BLOCO;
30    mostraWired = false;
31    RENDER_MODE = 0x0007; //GL_QUADS
32}
33
34void Map::renderBloco(float width, float height, float flatness, bool left,
35        bool right, bool front, bool back, bool top, int TYPE = GL_QUADS)
36{
37    float w = width/2;
```

```
38      float h = height/2;
39      float f = flatness/2;
40
41      float xTexNumber = width/TAMANHO_BLOCO;
42
43      glEnable(GL_TEXTURE_2D);
44      glBindTexture(GL_TEXTURE_2D, wallTexture);
45      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
46      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
47
48
49      glBegin(TYPE);
50      //Front
51      if(front)
52      {
53           glNormal3f(0.0f, 0.0f, 1.0f);
54              //glNormal3f(-1.0f, 0.0f, 1.0f);
55              glTexCoord2f(0.0f, 0.0f);
56          glVertex3f(-w, -h, f);
57              //glNormal3f(1.0f, 0.0f, 1.0f);
58              glTexCoord2f(xTexNumber, 0.0f);
59          glVertex3f(w, -h, f);
60              //glNormal3f(1.0f, 0.0f, 1.0f);
61              glTexCoord2f(xTexNumber, 1.0f);
62          glVertex3f(w, h, f);
63              //glNormal3f(-1.0f, 0.0f, 1.0f);
64              glTexCoord2f(0.0f, 1.0f);
65          glVertex3f(-w, h, f);
66      }
67
68      //Right
69      if(right)
70      {
71           glNormal3f(1.0f, 0.0f, 0.0f);
72              //glNormal3f(1.0f, 0.0f, -1.0f);
73              glTexCoord2f(0.0f, 0.0f);
74          glVertex3f(w, -h, -f);
75              //glNormal3f(1.0f, 0.0f, -1.0f);
76              glTexCoord2f(0.0f, 1.0f);
77          glVertex3f(w, h, -f);
78              glTexCoord2f(1.0f, 1.0f);
79              //glNormal3f(1.0f, 0.0f, 1.0f);
80          glVertex3f(w, h, f);
81              glTexCoord2f(1.f, 0.0f);
82              //glNormal3f(1.0f, 0.0f, 1.0f);
83          glVertex3f(w, -h, f);
84      }
85
86      //Back
87      if(back)
88      {
89              glNormal3f(0.0f, 0.0f, -1.0f);
90              //glNormal3f(-1.0f, 0.0f, -1.0f);
91              glTexCoord2f(0.0f, 0.0f);
92          glVertex3f(-w, -h, -f);
93              //glNormal3f(-1.0f, 0.0f, -1.0f);
94              glTexCoord2f(0.0f, 1.0f);
95          glVertex3f(-w, h, -f);
96              //glNormal3f(1.0f, 0.0f, -1.0f);
97              glTexCoord2f(xTexNumber, 1.0f);
98          glVertex3f(w, h, -f);
99              //glNormal3f(1.0f, 0.0f, -1.0f);
100             glTexCoord2f(xTexNumber, 0.0f);
101         glVertex3f(w, -h, -f);
102     }
103
104
105     //Left
106     if(left)
107     {
108         glNormal3f(-1.0f, 0.0f, 0.0f);
109             //glNormal3f(-1.0f, 0.0f, -1.0f);
110             glTexCoord2f(0.0f, 0.0f);
111         glVertex3f(-w, -h, -f);
112             //glNormal3f(-1.0f, 0.0f, 1.0f);
113             glTexCoord2f(1.0f, 0.0f);
114         glVertex3f(-w, -h, f);
115             //glNormal3f(-1.0f, 0.0f, 1.0f);
116             glTexCoord2f(1.0f, 1.0f);
117         glVertex3f(-w, h, f);
118             //glNormal3f(-1.0f, 0.0f, -1.0f);
119             glTexCoord2f(0.0f, 1.0f);
120         glVertex3f(-w, h, -f);
121     }
122     glEnd();
123 glDisable(GL_TEXTURE_2D);
```

```
124     glBegin(TYPE);
125     //Top
126     if(top)
127     {
128         glNormal3f(0.0f, 1.0f, 0.0f);
129         //glNormal3f(-1.0f, 1.0f, -1.0f);
130         glVertex3f(-w, h, -f);
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     }
138
139     // Don't need background
140     /*
141     //Bottom
142     glNormal3f(0.0f, -1.0f, 0.0f);
143         //glNormal3f(-1.0f, -1.0f, -1.0f);
144     glVertex3f(-w, -h, -f);
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     */
152     glEnd();
153 }
154
155 void Map::render()
156 {
157     glPushMatrix();
158     float offset = (float)TAMANHO_BLOCO/2.0f;
159
160     // Glut start printing starting from the center
161     glTranslated(offset, offset, offset);
162     glColor3f(COR_PAREDE);
163
164     int indexX = (Camera::CameraControl.cameraX / TAMANHO_BLOCO);
165     int indexY = (Camera::CameraControl.cameraZ / TAMANHO_BLOCO);
166
167     int beginX = indexX - GAME_FOV;
168     int beginY = indexY - GAME_FOV;
169     int endX = indexX + GAME_FOV;
170     int endY = indexY + GAME_FOV;
171     if(endX > MAP_WIDTH)
172             endX = MAP_WIDTH;
173     if(endY > MAP_HEIGHT)
174         endY = MAP_HEIGHT;
175     if(beginX < 0)
176         beginX = 0;
177     if(beginY < 0)
178         beginY = 0;
179
180
181     for(int i = beginY; i < endY; i++)
182     {
183         for(int j = beginX; j < endX; j++)
184         {
185             glPushMatrix();
186                 renderTileOptimizado(j+i*MAP_WIDTH);
187             glPopMatrix();
188         }
189     }
190
191     //Desenha chao
192     glPopMatrix();
193
194 }
195 void Map::renderTileOptimizado(unsigned int i)
196 {
197     //Camera on center of square 0,0,0
198     glTranslated(listaTilesOptimizados[i].posX * TAMANHO_BLOCO,
199                  listaTilesOptimizados[i].posY * TAMANHO_BLOCO,
200                  listaTilesOptimizados[i].posZ * TAMANHO_BLOCO);
201
202
203     if(listaTilesOptimizados[i].typeId == TILE_TIPO_PAREDE )
204     {
205         renderBloco(listaTilesOptimizados[i].tamanho, listaTilesOptimizados[i].tamanho, listaTilesOptimizados[i].tamanho,
206                     listaTilesOptimizados[i].left,listaTilesOptimizados[i].right,listaTilesOptimizados[i].front,
207                     listaTilesOptimizados[i].back,listaTilesOptimizados[i].top,
208                     RENDER_MODE);
209
```

```
210     }
211     else //Print ground
212     {
213         glEnable(GL_TEXTURE_2D);
214         glBindTexture(GL_TEXTURE_2D, floorTexture);
215         glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
216         glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
217 
218         float offset = (float)TAMANHO_BLOCO/2.0f;
219         glColor3f(COR_CHAO);
220         glBegin(RENDER_MODE);
221             glNormal3f(0.0f, 1.0f, 0.0f);
222                 glTexCoord2f(0.0f, 0.0f);
223             glVertex3f(-offset, -offset, -offset);
224                 glTexCoord2f(0.0f, 1.0f);
225             glVertex3f(-offset, -offset, offset);
226                 glTexCoord2f(1.0f, 1.0f);
227             glVertex3f(offset, -offset, offset);
228                 glTexCoord2f(1.0f, 0.0f);
229             glVertex3f(offset, -offset, -offset);
230         glEnd();
231         glColor3f(COR_PAREDE);
232         glDisable(GL_TEXTURE_2D);
233         if (listaTilesOptimizados[i].typeId == TILE_TIPO_CHAO_COM_BOLA)
234         {
235             glTranslated(0,-2,0);
236             glutSolidSphere(1,8,8);
237         }
238         else
239         if (listaTilesOptimizados[i].typeId == TILE_TIPO_CHAO_COM_BOLA_ESPECIAL)
240         {
241             glTranslated(0,-2,0);
242             glutSolidSphere(3,8,8);
243         }
244 
245     }
246 }
247 
248 
249 int Map::load(char* filename)
250 {
251     listaTiles.clear();
252 
253     FILE* file = fopen(filename, "r");
254 
255     if(file == NULL)
256         return -1;
257 
258     MAP_HEIGHT = MAP_WIDTH = 0;
259 
260     // Take the map size (blocks)
261     int error = fscanf(file, "%d-%d\n", &MAP_WIDTH, &MAP_HEIGHT);
262 
263     for (int y = 0; y < MAP_HEIGHT; y++)
264     {
265         for (int x = 0; x < MAP_WIDTH; x++)
266         {
267             Tile tempTile;
268             error = fscanf(file, "[%d] ",&tempTile.typeId);
269 
270             listaTiles.push_back(tempTile);
271         }
272         error = fscanf(file, "\n");
273     }
274     fclose(file);
275     ///TEST
276     geraQuadradosOptimizados();
277     return error;
278 }
279 
280 void Map::geraQuadradosOptimizados()
281 {
282     listaTilesOptimizados.clear();
283 
284     for(int iY = 0; iY < MAP_HEIGHT; iY++)
285     {
286        for(int iX = 0; iX < MAP_WIDTH; iX++) //Test all the blocks after this one in X
287        {
288             Tile retangulo;
289             int index = iX + MAP_WIDTH*iY;
290             if (listaTiles[index].typeId != TILE_TIPO_PAREDE)
291             {
292                 retangulo.typeId = listaTiles[index].typeId;
293                 retangulo.posX = iX;
294                 retangulo.posZ = iY;
295                 listaTilesOptimizados.push_back(retangulo);
```

```
296                 continue;
297             }
298 
299              retangulo.top = true;
300              //If wall, check out of the boards
301              if (index-1 < 0)
302                  retangulo.left = true;
303              else // If ground, than have any wall in this direction
304                  if (listaTiles[index-1].typeId != TILE_TIPO_PAREDE)
305                      retangulo.left = true;
306              if (index - MAP_WIDTH < 0)
307                  retangulo.back = true;
308              else // If ground, than have any wall in this direction
309                  if (listaTiles[index - MAP_WIDTH].typeId != TILE_TIPO_PAREDE)
310                      retangulo.back = true;
311              if (index +1 >= (int)listaTiles.size())
312                  retangulo.right = true;
313              else // If ground, than have any wall in this direction
314                  if (listaTiles[index +1].typeId != TILE_TIPO_PAREDE)
315                      retangulo.right = true;
316              if (index + MAP_WIDTH >= (int)listaTiles.size())
317                  retangulo.front = true;
318              else // If ground, than have any wall in this direction
319                  if (listaTiles[index + MAP_WIDTH].typeId != TILE_TIPO_PAREDE)
320                      retangulo.front = true;
321 
322              retangulo.posX = iX;
323              retangulo.posZ = iY;
324              retangulo.typeId = listaTiles[index].typeId;
325 
326              listaTilesOptimizados.push_back(retangulo);
327 
328          }
329      }
330 }
331 
332 
333 
334 void Map::setWired(int wired)
335 {
336     if (wired)
337     {
338         mostraWired = true;
339         RENDER_MODE = GL_LINES;
340     }
341     else
342     {
343         mostraWired = false;
344         RENDER_MODE = GL_QUADS;
345     }
346 
347 }
348 bool Map::isWire()
349 {
350     return mostraWired;
351 }
```

### A.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
112 namespace 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 }
```

### A.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;
```

### A.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 
155void 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 
177void 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 
198void 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}
```

### A.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    //glEnable(GL_LIGHT1); //enables light #1
52
53    glEnable(GL_DEPTH_TEST);
54    glShadeModel(GL_SMOOTH); //Shading
55
56    glEnable(GL_CULL_FACE); //Reduces the amount of triangles drawn.
57    glCullFace(GL_CW);
58
59    wallTexture = texture::loadTextureBMP("data/wall.bmp");
60    floorTexture = texture::loadTextureBMP("data/floor.bmp");
61
62
63}
64void GameManager::inicializa(void)
65{
66    inicializaRender();
67    inicializaSons();
68
69//----------------------------
70    //Specifies the background color
71    glClearColor(0.3f,0.3f,0.9f,1.0f);
72
73    GLfloat fog_color[4] = {0.0f,0.0f,0.0f,1.0};
74    glFogfv(GL_FOG_COLOR, fog_color);
75    glFogf(GL_FOG_DENSITY, 0.35f);
76
77    glFogi(GL_FOG_MODE, GL_LINEAR);
78    glHint(GL_FOG_HINT, GL_DONT_CARE);
79    glFogf(GL_FOG_START, TAMANHO_BLOCO*4.0f);
80    glFogf(GL_FOG_END, TAMANHO_BLOCO*10.0f);
81    glEnable(GL_FOG);
82
```

```
83      //Tests menu
84      menuPrincipal = true;
85
86      Button* start = new Button();
87
88      start->setXY(220, 200);
89      start->setEstados(1, 350, 60, 0);
90
91      start->ClickAction = startButtonAction;
92
93      Button::ButtonList.push_back(start);
94
95      for(unsigned int i = 0; i < MAX_ENEMY; i++) {
96          enemy[i] = new Entidade();
97          enemy[i]->addToEntidadeList();
98          enemy[i]->setTamanho(5);
99      }
100
101     Player::PlayerControl = new Player();
102     Player::PlayerControl->addToEntidadeList();
103
104 }
105
106 void GameManager::inicializaSons(void)
107 {
108     sc.init();
109
110     SOUND_main = sc.loadSound("data/mus/main.wav", 1);
111     SOUND_inter1 = sc.loadSound("data/mus/M1.WAV", 1);  //Linux & MAC are sensitive case
112     SOUND_inter2 = sc.loadSound("data/mus/M2.WAV", 1);
113     SOUND_inter3 = sc.loadSound("data/mus/M3.WAV", 1);
114     SOUND_attack = sc.loadSound("data/mus/atk.wav", 1);
115
116     SFX_die = sc.loadSound("data/sfx/die.wav", 0);
117     SFX_eat = sc.loadSound("data/sfx/eat.wav", 0);
118     SFX_eat2 = sc.loadSound("data/sfx/eat2.wav", 0);
119     SFX_alert = sc.loadSound("data/sfx/alert.wav", 0);
120
121
122     sc.play(SOUND_inter1);
123
124
125 }
126 void GameManager::resetPositions(void)
127 {
128     printf("Posicoes resetadas: %lu\n", Entidade::EntidadeList.size());
129
130     Map::MapControl.load((char*) "map_pacman_new.txt");
131
132     srand( time(NULL) );
133
134     for(int i = 0; i < MAX_ENEMY; i++) {
135         enemy[i]->setRandomPosition();
136     }
137
138     Player::PlayerControl->init();
139     Player::PlayerControl->resetPosition();
140 }
141 void desenhaTela(void)
142 {
143
144     game.render();
145
146
147     glutSwapBuffers();
148 }
149
150 void GameManager::loop(void)
151 {
152
153     FrameRate::FPSControl.loop();
154     for(unsigned int i = 0; i < Entidade::EntidadeList.size(); i++)
155     {
156         Entidade::EntidadeList[i]->loop();
157     }
158     for(unsigned int i = 0; i < Entidade::EntidadeList.size(); i++)
159     {
160         Entidade::EntidadeList[i]->testaColisao();
161     }
162     for(unsigned int i = 0; i < Entidade::EntidadeList.size(); i++)
163     {
164         Entidade::EntidadeList[i]->executaColisao();
165     }
166
167
168     //Verifies change of states on the special ball
```

```
169     if(attack_mode == 1) //notified change and play music
170     {
171         //Ste SPECIAL flag active for all entities. Even the player
172         for(unsigned int i = 0; i < Entidade::EntidadeList.size(); i++)
173         {
174             Entidade::EntidadeList[i]->flags = ENTIDADE_FLAG_ESPECIAL;
175         }
176         Player::PlayerControl->flags = ENTIDADE_FLAG_PLAYER_ESPECIAL; // resets the player's flag
177         ticksAttack = glutGet(GLUT_ELAPSED_TIME);
178         sc.stopAll();
179         sc.play(SFX_alert);
180         attack_mode = 2;
181     } else
182     if (attack_mode == 2)
183     {
184         //after 3 seconds
185         if( (glutGet(GLUT_ELAPSED_TIME) - ticksAttack) > 3000 )
186         {
187             sc.stopAll();
188             sc.play(SOUND_attack);
189             attack_mode = 3;
190             ticksAttack = glutGet(GLUT_ELAPSED_TIME);
191         }
192     } else
193     if (attack_mode == 3)
194     {
195         //over the end of the ball efects 10 seconds + 3 the preceding sfx
196         if( (glutGet(GLUT_ELAPSED_TIME) - ticksAttack) > 10000)
197         {
198             sc.stopAll();
199             sc.play(SOUND_inter2);
200             attack_mode = 0;
201             for(unsigned int i = 0; i < Entidade::EntidadeList.size(); i++)
202             {
203                 Entidade::EntidadeList[i]->flags = ENTIDADE_FLAG_NENHUM;
204             }
205             Player::PlayerControl->flags = ENTIDADE_FLAG_PLAYER_NORMAL; // resets the player's flag
206         }
207     }
208
209}
210void GameManager::render(void)
211{
212
213     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
214
215     glMatrixMode(GL_MODELVIEW);
216     glLoadIdentity();
217
218     if (menuPrincipal)
219     {
220         for(unsigned int i = 0; i < Button::ButtonList.size();i++)
221             Button::ButtonList[i]->render();
222
223         txt::renderText2dOrtho(30,150,8,"Aperte o grande quadrado branco para comecar!!!");
224
225         switch(status)
226         {
227             case STATUS_DERROTA:
228                 txt::renderText2dOrtho(30,130,8,"Derrota!!!");
229                 break;
230             case STATUS_NORMAL:
231                 txt::renderText2dOrtho(30,130,8,"Novo jogo!!!");
232                 break;
233             case STATUS_VITORIA:
234                 txt::renderText2dOrtho(30,130,8,"Vitoria!!!");
235                 break;
236                 default:;
237         }
238
239         return;
240     }
241
242
243
244
245     //Lighting
246     GLfloat ambientLight[] = {0.1f, 0.1f, 0.1f, 1.0f};
247     glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientLight);
248     GLfloat directedLight[] = {0.7f, 0.7f, 0.7f, 0.0f};
249     GLfloat directedLightPos[] = {0.0f, 20.0f, -20.0f, 1.0f};
250     GLfloat light[] = {0.9f, 0.9f, 0.9f, 1.0f};
251     GLfloat lightPos[] = {100.0f, 30.0f, -10.0f, 1.0f};
252     glLightfv(GL_LIGHT0, GL_DIFFUSE, directedLight);
253     glLightfv(GL_LIGHT0, GL_POSITION, directedLightPos);
254     glLightfv(GL_LIGHT1, GL_DIFFUSE, light);
```

```
255     glLightfv(GL_LIGHT1, GL_POSITION, lightPos);
256     //end of lighting
257
258
259     //calculates iterations
260     this->loop();
261
262     //Print SOL's
263     glPushMatrix();
264         glColor3f(1.0f, 1.0f, 1.0f);
265         glTranslatef(directedLightPos[0],directedLightPos[1],directedLightPos[2]);
266         glutSolidSphere(10.0f, 18.0f, 18.0f);
267     glPopMatrix();
268     glPushMatrix();
269         glColor3f(1.0f, 0.0f, 0.0f);
270         glTranslatef(lightPos[0],lightPos[1],lightPos[2]);
271         glutSolidSphere(10.0f, 18.0f, 18.0f);
272     glPopMatrix();
273
274     Map::MapControl.render();
275     //unsigned int temp = Entidade::EntidadeList.size();
276     for(unsigned int i = 0; i < Entidade::EntidadeList.size(); i++)
277     {
278         if (Entidade::EntidadeList[i]->isVisible())
279             Entidade::EntidadeList[i]->render();
280     }
281
282     txt::renderText2dOrtho(10,15,0,"FPS: %.2f",FrameRate::FPSControl.getFPS());
283
284
285
286
287     MiniMap::renderMiniMap();
288
289}
290
291
292// when called during cleanup destructor,
293// segmentation fault occurs only delete the Entity
294GameManager::~GameManager()
295{
296     sc.stopAll();
297     sc.exit();
298}
299void cleanup(void)
300{
301     unsigned int sizeEnt = Entidade::EntidadeList.size();
302     unsigned int sizeBtn = Button::ButtonList.size();
303     printf("Entidade cleanup size: %u\n", sizeEnt);
304     for(unsigned int i = 0; i < sizeEnt; i++)
305         delete Entidade::EntidadeList[i];
306     printf("Button cleanup size: %u\n", sizeBtn);
307     for(unsigned int i = 0; i < sizeBtn; i++)
308         delete Button::ButtonList[i];
309     printf("EXIT\n");
310}
311void testOpenAL()
312{
313     unsigned int g_buf = -1;
314     unsigned int g_src = -1;
315
316     if(!alutInit(NULL, NULL))
317     {
318         printf("%s",alutGetErrorString(alutGetError()));
319         return;
320     }
321     alGetError();
322     alutGetError();
323
324     g_buf = alutCreateBufferFromFile("testing.wav");
325
326     if (alutGetError() != ALUT_ERROR_NO_ERROR)
327      {
328          alDeleteBuffers(1, &g_buf);
329          alutExit();
330          return;
331      }
332
333      alGenSources(1, &g_src);
334
335      if(alGetError() != AL_NO_ERROR)
336      {
337          alDeleteBuffers(1, &g_buf);
338          alDeleteSources(1, &g_buf);
339          alutExit();
340          return;
```

```
341     }
342
343     alSourcei(g_src, AL_BUFFER, g_buf);
344
345     alSourcePlay(g_src);
346     alutSleep(4.0f);
347
348     alutExit();
349}
350void testSoundALClass()
351{
352    SoundAL sn;
353    sn.init();
354
355    int m_i = sn.loadSound("testing.wav", 1);
356    sn.play(m_i);
357
358    alutSleep(4.0f);
359
360    sn.exit();
361}
362int main(int argc, char* args[])
363{
364
365    //testOpenAL();
366    //testSoundALClass();
367
368    game.executa(argc, args);
369    return 0;
370}
371void GameManager::executa(int argc, char* args[])
372{
373    glutInit(&argc, args);
374    glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
375    glutInitWindowPosition(100,100);
376    glutInitWindowSize(SCREEN_WIDTH, SCREEN_HEIGHT);
377    glutCreateWindow("Labirinth");
378
379    inicializa();
380
381    glutDisplayFunc(desenhaTela);
382    glutReshapeFunc(changeSize);
383    glutIdleFunc(desenhaTela);
384
385    glutKeyboardFunc(teclasNormais);
386    glutKeyboardUpFunc(teclasNormaisUp);
387    glutSpecialFunc(teclasEspeciais);
388    glutSpecialUpFunc(teclasEspeciaisSoltar);
389    glutMotionFunc(moveMouse);
390    glutMouseFunc(mouseButton);
391
392    atexit(cleanup);
393
394    glutIgnoreKeyRepeat(0);
395    //Get in the loop processing events
396    glutMainLoop();
397}
```

### A.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  }
```

### A.2.10 Title

```
1 #include "tile.h"
2
3 Tile::Tile()
4 {
5     tamanho = TAMANHO_BLOCO;
6     posY = 0;
7
8     left = right = front = back = top = bottom = false;
9 }
```

### A.2.11 Makefile

```
1  ###########################################################
2  #                        Makefile
3  #                Friday 17 August 2012
4  ###########################################################
5  CC = g++
6  CFLAGS =$(GLFLAGS) -I./ -O3 -Os -g $(PROBLENS)
7  CC_WINDOWS = x86_64-linux-gnu-g++
8
9  PROBLENS=-Wall -pedantic -fpermissive
10 UNAME = $(shell uname)
11 OUTPUT = Amaze.out
12
13 define PROGRAM_template
14 $(1): $(addsuffix .o,$(1))
15 endef
16 $(foreach t,$(compiling),$(eval $(call PROGRAM_template,$(t))))
17
18
19 ifeq ($(UNAME),Linux) # Linux OS
20     GLFLAGS = -lglut -lglui -lGLU -lGL -lalut  -lopenal
21     SEARCH = dpkg -l | grep -iq
22     else
23     ifeq ($(UNAME),Darwin) # MAC OS X
24         GLFLAGS = -framework OpenGL -framework GLUT -framework OpenAL
25         SEARCH = ls /System/Library/Frameworks | grep -i
26     else #Windows
27         GLFLAGS = -lopengl32 -lglu32 -lglut32 -lglee -lalut
28         SEARCH=
29     endif
30 endif
31
32 all: *.cpp
```

```
33      if $(MAKE) compiling ;\
34      then \
35          echo -n "ok\nCleaning..." ;\
36          rm *.o ;\
37          echo "done.\nRun "$(OUTPUT) ;\
38      else \
39          echo  "Error on compiling! Probably some package is missing"; \
40          $(MAKE) check;\
41      fi;
42
43  compiling:*.cpp
44      echo "System: "$(UNAME) "OS"
45      echo -n "Compiling..."
46      $(CC) *.cpp -c $(CFLAGS)
47      $(CC) *.cpp -MM $(CFLAGS) > depends.d
48      $(CC) *.o -o $(OUTPUT) $(CFLAGS)
49
50  clean:
51      echo "Cleaning all..."
52      rm -rfv $(OUTPUT) *.o *.d
53
54  run: all
55      echo "Running..."
56      ./$(OUTPUT)
57
58  valgrind: *.cpp
59  #   $(CC) -g -c *.cpp
60  #   ar rc libAmaze.a *.o
61  #   $(CC) -g gamemanager.cpp -o ToGring $(GLFLAGS) -L./ -lAmaze
62      valgrind --tool=callgrind --dsymutil=yes --trace-jump=yes ./ToGring -q --fullpath-after=string
    --show-possibly-lost=yes --trace-children=yes -v
63      echo "Valgrind files available: (newer first)"
64      ls -t| egrep -i grind
65
66  windows: *.cpp
67      echo "Cross compiling to" $@
68      $(CC_WINDOWS) *.cpp -c $(CFLAGS)
69      $(CC_WINDOWS) *.o -o ./bin/x86-x64-Amaze.exe $(CFLAGS)
70      echo "done.\nRun " x86-x64-Amaze.exe "on bin directory"
71
72  check:
73      echo "Checking if all dev packages are installed"
74  #   OPENGL
75      echo -n "opengl "
76      if $(SEARCH) "opengl" ;\
77      then \
78          echo "[OK]";\
79      else \
80          echo "[MISSING!]" ;\
81      fi;
82  #   OPENAL
83      echo -n "openal "
84      if $(SEARCH) "openal" ;\
85      then \
86          echo "[OK]";\
87      else \
88          echo "[MISSING!]" ;\
89      fi;
90  #   GLUT
91      echo -n "glut "
92      if $(SEARCH)  "glut" ;\
93      then \
94          echo "[OK]";\
95      else \
96          echo "[MISSING!]" ;\
97      fi;
98  #   GLUI
99      echo -n "glui "
100     if $(SEARCH)  "glui" ;\
101     then \
102         echo "[OK]";\
103     else \
104         echo "[MISSING!]" ;\
105     fi;
106 #   ALUT
107     echo -n "alut "
108 #   if $(SEARCH) | grep -qi "alut.*dev" ;\ #Como deveria de ser pra ficar otimo!
109     if $(SEARCH)  "alut" ;\
110     then \
111         echo "[OK]";\
112     else \
113         echo "[MISSING!]" ;\
114     fi;
115 #   GLEE
116     echo -n "glee "
117     if $(SEARCH)  "glee" ;\
```

```
118     then \
119         echo "[OK]";\
120     else \
121         echo "[MISSING!]" ;\
122     fi;
123 
124 .SILENT:
125 
126 #Obs
127 #
128 #   Bibliotecas incluidas:
129 #
130 #   alut-dev
131 #   openal-dev
132 #
133 #   Descobrindo pacotes instalados:
134 # $ dpkg -l | grep alut
135 #
136 #   No MacOS os Frameworks ficam no diretorio/System/Library/Frameworks
137 # e possuem a nomeclatura semelhante a:
138 # OpenAL.framework
```

### A.2.12 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.

# 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.

# Mac OS

    Similar to the steps on the Linux system, the user must run the command "make run" in the directory containing
the makefile to compile the files and start the program correctly.
```

# APÊNDICE B
# ANEXOS

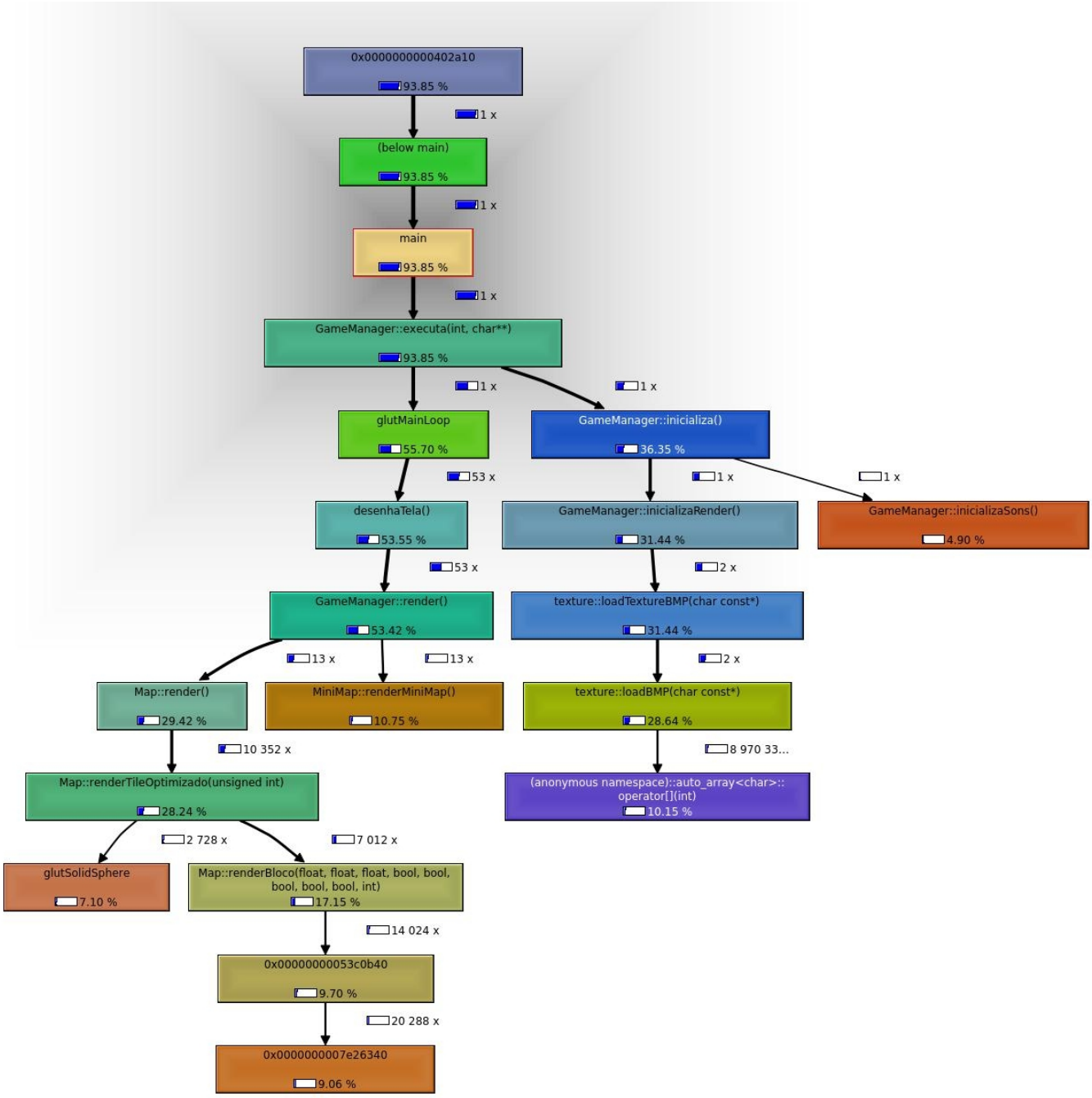


**Figura 2:** Saída gerada pelo Valgrind