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.

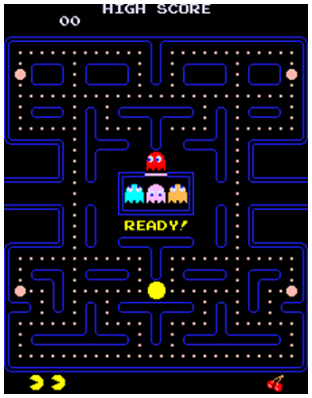

Figura 1: Pac-Man.
O clássico dos anos 80 só foi ter um score perfeito - máximo de pontos, sem falhas ou mortes - em 1999, quando *Billy Mitchell* consegui a incrível marca de $3,333,360$ pontos, após vencer os consecutivos 256 leveis do jogo.

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...
```

2.1.3 Artefatos

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

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

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

2. Atualizado em 7 de Junho de 2012

- 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]

As Figuras são simplesmente inseridas como mostrado na Fig. 2

Figura 2: Arquitetura do Programa.

2.2 Artefatos

Os artefatos entregues devem ser documentados no relatório:

- Arquivos contidos no programa. Lista dos nomes dos arquivos, assim como a extensão dos arquivo
- Aquivo README, com instruções de uso do software desenvolvido e necessidades técnicas para a execução do programa
- Arquivos de entrada/saída, caso necessário.

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(); //seta posicao e direcao da camera
        void loop();  //ajusta 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 como public
        void calculaDirecao(void);

        //Liga ou desliga correr
        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        //nao utilizado
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 da ultima vez que calculou o movimento
55        unsigned int respawnTicks;// ticks de quando morreu
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
13  class 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; //Posicao aonde o mapa comeca a renderizar,
38          float origemZ; //Tile 0,0, aumenta pra direita-baixo
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          //Usado pra outras classes obterem info sobre o mapa.
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
69
70#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#else
8    #ifdef _WIN32                          /* Windows */
9        #define WIN32_LEAN_AND_MEAN
10       #include <glee.h>
11       #include <gl/gl.h>
12       #include <gl/glut.h>
13       #include <windows.h>
14       #define sleep(x) Sleep(x)
15   #else                                  /*Linux*/
16       #include <cstdarg>
17       #include <unistd.h>
18       #include <GL/gl.h>
19       #include <GL/glut.h>
20       #include <GL/glu.h>
21       #define Sleep(x) usleep(x<1000000?10000+300*x:x)
22   #endif
23#endif
24
25#define SCREEN_WIDTH            800
26#define SCREEN_HEIGHT           600
27
28#define FRAMES_PER_SECOND       60.0f
29
30#define TAMANHO_BLOCO           12
31#define COR_PAREDE              1.0f, 1.0f, 1.0f
32#define COR_CHAO                1.0f, 1.0f, 1.0f
33#define GAME_FOV                28
34
35#define PONTOS_BOLA             10
36#define PONTOS_BOLA_ESPECIAL    50
37
38#define TAMANHO_INIMIGO         5
39
40
41
42//Tamanho da tela atual
43extern float wScreen;
```

```
44extern float hScreen;
45//Texturas
46extern GLuint wallTexture;
47extern GLuint floorTexture;
48//Menu
49extern bool menuPrincipal;
50extern int status;
51
52//Sons
53extern int SOUND_main;
54extern int SOUND_inter1;
55extern int SOUND_inter2;
56extern int SOUND_inter3;
57extern int SOUND_attack;
58extern int SFX_die;
59extern int SFX_eat;
60extern int SFX_eat2;
61extern int SFX_alert;
62//Globais de gameplay
63extern int attack_mode;
64
65#define STATUS_NORMAL 0
66#define STATUS_VITORIA 1
67#define STATUS_DERROTA 2
68
69
70
71#endif
```

A.1.7 Eventos

```
1#ifndef EVENTOS_H_
2#define EVENTOS_H_
3
4
5extern void teclasNormais(unsigned char key, int x, int y);
6extern void teclasNormaisUp(unsigned char key, int x, int y);
7extern void teclasEspeciais(int key, int x, int y );
8extern void teclasEspeciaisSoltar(int key, int x, int y);
9extern void mouseButton(int button, int state, int x, int y);
10extern void moveMouse(int x, int y);
11
12#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
24}
25
26
27
28#endif
```

A.2 Sources

A.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    //testes
19
20    //testes
21    deltaAngleX = deltaAngleY = 0.0f; //Angulo de rotacao da direcao horizontal e vertical
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
41void 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    //testes
55
56    //testes
57    deltaAngleX = deltaAngleY = 0.0f; //Angulo de rotacao da direcao horizontal e vertical
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//Chamada internamente por 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
98 void Camera::calculaDirecao(void)
99 {
100     float fator = deltaTicks/1000.f;
101     angleX += deltaAngleX*fator;
102     angleY += deltaAngleY*fator;
103
104     //corrige angulo
105     if ( angleX+angleOffsetX >= 360 )
106         angleX -= 360;
107     if ( angleX+angleOffsetX < 0)
108         angleX += 360;
109
110     //So permite rotacionar 180 graus em 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     //Adiciona ao movimento
125     float fator = deltaTicks/1000.f;
126
127     //Fator delta vezes direcao. 0.1f para ajustar velocidade.
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     //Adiciona ao movimento
138     //Fator delta vezes direcao. 0.1f para ajustar velocidade.
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;
```

```
}
void Camera::giraDireita(bool mover)
{
    if(mover)
        deltaAngleX = -velocidadeVira;
    else
        deltaAngleX = 0.0f;
}
void Camera::giraCima(bool mover)
{
    if(mover)
        deltaAngleY = velocidadeVira;
    else
        deltaAngleY = 0.0f;
}
void Camera::giraBaixo(bool mover)
{
    if(mover)
        deltaAngleY = -velocidadeVira;
    else
        deltaAngleY = 0.0f;
}

void Camera::setMouse(int x, int y)
{
    xOrigem = x;
    yOrigem = y;

    if (xOrigem == -1) // Ambos serao -1 necessariamente
    {
        angleX +=angleOffsetX;
        angleY +=angleOffsetY;
        angleOffsetX = 0;
        angleOffsetY = 0;
    }
}
void Camera::moveMouse(int x, int y)
{
    deltaMouseX = deltaMouseY = 0;
    //Se houve deslocamento
    if (xOrigem>0)
    {
        angleOffsetX = (xOrigem-x) * 0.1f;
    }
    if (yOrigem>0)
    {
        angleOffsetY = (yOrigem-y) * 0.1f;
    }
    calculaDirecao();
}

void Camera::setCorrer(void)
{
    velocidadeMove = velocidadeMoveCorre;
}
void Camera::setAndar(void)
{
    velocidadeMove = velocidadeMoveAndar;
}
```

A.2.2 Entidade

```
#include "entidade.h"

#include <stdlib.h>




//=============================================================================
// Variaveis estaticas
//=============================================================================
std::vector<Entidade*> Entidade::EntidadeList;

//=============================================================================
// Construtores
//=============================================================================
Entidade::Entidade()
{
    flags = ENTIDADE_FLAG_NENHUM;
    entidadeColidida.clear();
    deltaTicks = 9999999;
    deltaTicks = 0;
    tamanho.x = tamanho.y = tamanho.z = 10;
    visible = true;
    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      //Nota, o ponto posicao marca 0.... ex: posicao 0 comeco do bloco final do bloco em x,y,z
48      //Tal que y mais abaixo = y e y mais alto = 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  // Retorna true se estiver colidindo com o mapa
80  //=============================================================================
81  Tile* Entidade::isColisaoMapa(Vetor3D newPosicao, int type)
82  {
83      //Calcula o Id do tile que deve ser testado
84      //Ex: X = 5 tal que 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      //Checa colisoes com os 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 // Executa acoes do loop, aceleracao, velocidade.
130 //=============================================================================
131 void Entidade::loop()
132 {
133     //passou 3 segundos do respawn
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 reseta o render
144     delta = glutGet(GLUT_ELAPSED_TIME) - deltaTicks;
145     float fator = delta/1000.f;
146
147     //Calcula aceleracoes
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, enquanto utilizar glutCube
189     glPushMatrix();
190     //Centraliza devido ao 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     //Testa com todas as entidades desta para frente.
218     //Ex:    lista: 1 2 3 4
219     // thisID =1,  testa com 2, 3 , 4
220     // thisID = 2  testa com 3, 4      desta, forma, thisID = 2 nao testa colisoes com 1 pois ja foi testado anteriormente.
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             {   //adiciona colisoes tanto neste elemento quanto no testado
227                 setColisao(EntidadeList[i]);
228                 EntidadeList[i]->setColisao(this);
229             }
230         }
231     }
232 }
233 //Seta colisao atraves de metodo publico
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; // sem colisoes
249
250
251
252     //Volta o que tinha movido.
253     float fator = delta/1000.f;
254     posicao = posicao - (velocidade * fator );
255     //Para, e vai na direcao oposta
256     velocidade.x = 0;
257     velocidade.z = 0;
258     aceleracao.x = -aceleracao.x;
259     aceleracao.z = -aceleracao.z;
260
261     if ( (flags == ENTIDADE_FLAG_ESPECIAL) && (entidadeColidida[0]->flags == ENTIDADE_FLAG_PLAYER_ESPECIAL) )
262     {
263         flags = ENTIDADE_FLAG_RESPAWN;
264         respawnTicks = glutGet(GLUT_ELAPSED_TIME);
265         dead = true;
266         visible = false;
267         SoundAL sc;
268         sc.play(SFX_eat2);
269     }
270
271     entidadeColidida.clear();
272 }
273
274 void Entidade::setRandomPosition()
275 {
276     bool isOK = false;
277         while(!isOK) {
278             int posX = rand() % Map::MapControl.MAP_WIDTH;
279             int posZ = rand() % Map::MapControl.MAP_HEIGHT;
280
281             //Se a posicao for diferente de parede, entao chao.... coloca cubo
282             if (Map::MapControl.getTile(posX, posZ)->typeId != TILE_TIPO_PAREDE) {
```

```
283                 //nota (TAMANHO_BLOCO/2 - tamanho.x/2)  serve para achar o meio do chao
284                 posicao.x = (TAMANHO_BLOCO/2 - tamanho.x/2) + TAMANHO_BLOCO*posX;
285                 posicao.y = 0;
286                 posicao.z = (TAMANHO_BLOCO/2 - tamanho.z/2) + TAMANHO_BLOCO*posZ;
287                 //1 a 10
288                 aceleracao.x = 1 + rand() % 10;
289                 aceleracao.z = 1 + rand() % 10;
290                 init();
291                 isOK = true;
292                 ///Possivel adicionar verificacao se a entidade nao ficou no mesmo lugar usando isColisao e clear() da lista de colisoe
293             }
294         }
295 }
296
297 bool Entidade::isVisible()
298 {
299     return visible;
300 }
301 void Entidade::setTamanho(float newTamanho)
302 {
303     tamanho.x = tamanho.y = tamanho.z = newTamanho;
304 }
305 void Entidade::setPosicao(float x, float y, float z)
306 {
307     posicao.x = x;
308     posicao.y = y;
309     posicao.z = z;
310 }
311 void Entidade::setColor3f(float fr, float fg, float fb)
312 {
313     r = fr;
314     g = fg;
315     b = fb;
316 }
317 float Entidade::getColor(int rgb_i)
318 {
319     float color = 0.0f;
320     switch(rgb_i)
321     {
322         case 1:
323             color = r;
324             if (flags == ENTIDADE_FLAG_ESPECIAL)
325                 color -= 0.55f;
326             break;
327         case 2:
328             color = g;
329             if (flags == ENTIDADE_FLAG_ESPECIAL)
330                 color += 1;
331             break;
332         case 3:
333             color = b;
334             if (flags == ENTIDADE_FLAG_ESPECIAL)
335                 color += 0.95f;
336             break;
337     }
338     return color;
339 }
```

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 }
12 void FrameRate::setFPSCap(bool cap)
13 {
14     fpsCap = cap;
15 }
16 bool FrameRate::isFPSCap()
17 {
18     return fpsCap;
19 }
20 FrameRate::FrameRate()
21 {
22     ticks = glutGet(GLUT_ELAPSED_TIME);
23     ticksControl = glutGet(GLUT_ELAPSED_TIME);
24     frames = 0;
25     fps = 0;
26     fpsCap = false;
```

```
}

void FrameRate::regulaFPS()
{
    unsigned int step = 1000.0f/FRAMES_PER_SECOND;
    unsigned int decorrido = glutGet(GLUT_ELAPSED_TIME) - ticksControl;
    if(decorrido < step )
        Sleep( step - decorrido);

    ticksControl = glutGet(GLUT_ELAPSED_TIME);
}

void FrameRate::loop()
{
    unsigned int decorrido = glutGet(GLUT_ELAPSED_TIME) - ticks;
    frames++;
    if (decorrido > 1000)
    {
        fps = ((float)frames*1000.0f/(float)decorrido);

        frames = 0;
        ticks = glutGet(GLUT_ELAPSED_TIME);
    }

    if (fpsCap)
        regulaFPS();

}
```

A.2.4 Map

```
#include "map.h"

//Usado pra outras classes obterem info sobre o mapa.
Map Map::MapControl;


//Pega o Tile na posicao x,y do mapa.
//Ex: Map 1 2 3   vector sera 1 2 3 4 5 6
//        4 5 6
Tile* Map::getTile(int x, int y)
{
    unsigned int ID = 0;

    ID = (y * MAP_WIDTH) + x;

    return &listaTilesOptimizados[ID];
}
inline int Map::getX(int i)
{
    return i % MAP_WIDTH;
}
inline int Map::getY(int i)
{
    return (int) i/MAP_WIDTH;
}

Map::Map()
{
    origemX = -TAMANHO_BLOCO;
    origemZ = -TAMANHO_BLOCO;
    mostraWired = false;
    RENDER_MODE = 0x0007; //GL_QUADS
}

void Map::renderBloco(float width, float height, float flatness, bool left,
        bool right, bool front, bool back, bool top, int TYPE = GL_QUADS)
{
    float w = width/2;
    float h = height/2;
    float f = flatness/2;

    float xTexNumber = width/TAMANHO_BLOCO;

    glEnable(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D, wallTexture);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);


    glBegin(TYPE);
    //Front
    if(front)
    {
         glNormal3f(0.0f, 0.0f, 1.0f);
            //glNormal3f(-1.0f, 0.0f, 1.0f);
```

```
56              glTexCoord2f(0.0f, 0.0f);
57          glVertex3f(-w, -h, f);
58              //glNormal3f(1.0f, 0.0f, 1.0f);
59              glTexCoord2f(xTexNumber, 0.0f);
60          glVertex3f(w, -h, f);
61              //glNormal3f(1.0f, 0.0f, 1.0f);
62              glTexCoord2f(xTexNumber, 1.0f);
63          glVertex3f(w, h, f);
64              //glNormal3f(-1.0f, 0.0f, 1.0f);
65              glTexCoord2f(0.0f, 1.0f);
66          glVertex3f(-w, h, f);
67      }
68
69      //Right
70      if(right)
71      {
72           glNormal3f(1.0f, 0.0f, 0.0f);
73              //glNormal3f(1.0f, 0.0f, -1.0f);
74              glTexCoord2f(0.0f, 0.0f);
75          glVertex3f(w, -h, -f);
76              //glNormal3f(1.0f, 0.0f, -1.0f);
77              glTexCoord2f(0.0f, 1.0f);
78          glVertex3f(w, h, -f);
79              glTexCoord2f(1.0f, 1.0f);
80              //glNormal3f(1.0f, 0.0f, 1.0f);
81          glVertex3f(w, h, f);
82              glTexCoord2f(1.f, 0.0f);
83              //glNormal3f(1.0f, 0.0f, 1.0f);
84          glVertex3f(w, -h, f);
85      }
86
87      //Back
88      if(back)
89      {
90              glNormal3f(0.0f, 0.0f, -1.0f);
91              //glNormal3f(-1.0f, 0.0f, -1.0f);
92              glTexCoord2f(0.0f, 0.0f);
93          glVertex3f(-w, -h, -f);
94              //glNormal3f(-1.0f, 0.0f, -1.0f);
95              glTexCoord2f(0.0f, 1.0f);
96          glVertex3f(-w, h, -f);
97              //glNormal3f(1.0f, 0.0f, -1.0f);
98              glTexCoord2f(xTexNumber, 1.0f);
99          glVertex3f(w, h, -f);
100             //glNormal3f(1.0f, 0.0f, -1.0f);
101             glTexCoord2f(xTexNumber, 0.0f);
102         glVertex3f(w, -h, -f);
103     }
104
105
106     //Left
107     if(left)
108     {
109         glNormal3f(-1.0f, 0.0f, 0.0f);
110             //glNormal3f(-1.0f, 0.0f, -1.0f);
111             glTexCoord2f(0.0f, 0.0f);
112         glVertex3f(-w, -h, -f);
113             //glNormal3f(-1.0f, 0.0f, 1.0f);
114             glTexCoord2f(1.0f, 0.0f);
115         glVertex3f(-w, -h, f);
116             //glNormal3f(-1.0f, 0.0f, 1.0f);
117             glTexCoord2f(1.0f, 1.0f);
118         glVertex3f(-w, h, f);
119             //glNormal3f(-1.0f, 0.0f, -1.0f);
120             glTexCoord2f(0.0f, 1.0f);
121         glVertex3f(-w, h, -f);
122     }
123     glEnd();
124 glDisable(GL_TEXTURE_2D);
125     glBegin(TYPE);
126     //Top
127     if(top)
128     {
129         glNormal3f(0.0f, 1.0f, 0.0f);
130         //glNormal3f(-1.0f, 1.0f, -1.0f);
131         glVertex3f(-w, h, -f);
132             //glNormal3f(-1.0f, 1.0f, 1.0f);
133         glVertex3f(-w, h, f);
134             //glNormal3f(1.0f, 1.0f, 1.0f);
135         glVertex3f(w, h, f);
136             //glNormal3f(1.0f, 1.0f, -1.0f);
137         glVertex3f(w, h, -f);
138     }
139
140     ///Nao precisa imprimir fundo
141     /*
```

```
142     //Bottom
143     glNormal3f(0.0f, -1.0f, 0.0f);
144         //glNormal3f(-1.0f, -1.0f, -1.0f);
145     glVertex3f(-w, -h, -f);
146         //glNormal3f(-1.0f, -1.0f, 1.0f);
147     glVertex3f(-w, -h, f);
148         //glNormal3f(1.0f, -1.0f, 1.0f);
149     glVertex3f(w, -h, f);
150         //glNormal3f(1.0f, -1.0f, -1.0f);
151     glVertex3f(w, -h, -f);
152     */
153     glEnd();
154 }
155
156 void Map::render()
157 {
158     glPushMatrix();
159     float offset = (float)TAMANHO_BLOCO/2.0f;
160     glTranslated(offset, offset, offset); //Pois o glut imprime a partir do centro
161     glColor3f(COR_PAREDE);
162
163     int indexX = (Camera::CameraControl.cameraX / TAMANHO_BLOCO);
164     int indexY = (Camera::CameraControl.cameraZ / TAMANHO_BLOCO);
165
166     int beginX = indexX - GAME_FOV;
167     int beginY = indexY - GAME_FOV;
168     int endX = indexX + GAME_FOV;
169     int endY = indexY + GAME_FOV;
170     if(endX > MAP_WIDTH)
171         endX = MAP_WIDTH;
172     if(endY > MAP_HEIGHT)
173         endY = MAP_HEIGHT;
174     if(beginX < 0)
175         beginX = 0;
176     if(beginY < 0)
177         beginY = 0;
178
179
180     for(int i = beginY; i < endY; i++)
181     {
182         for(int j = beginX; j < endX; j++)
183         {
184             glPushMatrix();
185                 renderTileOptimizado(j+i*MAP_WIDTH);
186             glPopMatrix();
187         }
188     }
189
190     //Desenha chao
191     glPopMatrix();
192
193 }
194 void Map::renderTileOptimizado(unsigned int i)
195 {
196     //Camera no centro do quadrado 0,0,0
197     glTranslated(listaTilesOptimizados[i].posX * TAMANHO_BLOCO,
198                  listaTilesOptimizados[i].posY * TAMANHO_BLOCO,
199                  listaTilesOptimizados[i].posZ * TAMANHO_BLOCO);
200
201
202     if(listaTilesOptimizados[i].typeId == TILE_TIPO_PAREDE )
203     {
204         renderBloco(listaTilesOptimizados[i].tamanho, listaTilesOptimizados[i].tamanho, listaTilesOptimizados[i].tamanho,
205                     listaTilesOptimizados[i].left,listaTilesOptimizados[i].right,listaTilesOptimizados[i].front,
206                     listaTilesOptimizados[i].back,listaTilesOptimizados[i].top,
207                     RENDER_MODE);
208
209     }
210     else //Imprime chao
211     {
212         glEnable(GL_TEXTURE_2D);
213         glBindTexture(GL_TEXTURE_2D, floorTexture);
214         glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
215         glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
216
217         float offset = (float)TAMANHO_BLOCO/2.0f;
218         glColor3f(COR_CHAO);
219         glBegin(RENDER_MODE);
220             glNormal3f(0.0f, 1.0f, 0.0f);
221                 glTexCoord2f(0.0f, 0.0f);
222             glVertex3f(-offset, -offset, -offset);
223                 glTexCoord2f(0.0f, 1.0f);
224             glVertex3f(-offset, -offset, offset);
225                 glTexCoord2f(1.0f, 1.0f);
226             glVertex3f(offset, -offset, offset);
227                 glTexCoord2f(1.0f, 0.0f);
```

```
228             glVertex3f(offset, -offset, -offset);
229         glEnd();
230         glColor3f(COR_PAREDE);
231         glDisable(GL_TEXTURE_2D);
232         if (listaTilesOptimizados[i].typeId == TILE_TIPO_CHAO_COM_BOLA)
233         {
234             glTranslated(0,-2,0);
235             glutSolidSphere(1,8,8);
236         }
237         else
238         if (listaTilesOptimizados[i].typeId == TILE_TIPO_CHAO_COM_BOLA_ESPECIAL)
239         {
240             glTranslated(0,-2,0);
241             glutSolidSphere(3,8,8);
242         }
243
244     }
245 }
246
247
248 int Map::load(char* filename)
249 {
250     listaTiles.clear();
251
252     FILE* file = fopen(filename, "r");
253
254     if(file == NULL)
255         return -1;
256
257     MAP_HEIGHT = MAP_WIDTH = 0;
258
259     //Pega o tamanho do mapa, quanto por quantos blocos
260     int error = fscanf(file, "%d-%d\n", &MAP_WIDTH, &MAP_HEIGHT);
261
262     for (int y = 0; y < MAP_HEIGHT; y++)
263     {
264         for (int x = 0; x < MAP_WIDTH; x++)
265         {
266             Tile tempTile;
267             error = fscanf(file, "[%d] ",&tempTile.typeId);
268
269             listaTiles.push_back(tempTile);
270         }
271         error = fscanf(file, "\n");
272     }
273     fclose(file);
274     ///TESTE
275     geraQuadradosOptimizados();
276     return error;
277 }
278
279 void Map::geraQuadradosOptimizados()
280 {
281     listaTilesOptimizados.clear();
282
283     for(int iY = 0; iY < MAP_HEIGHT; iY++)
284     {
285        for(int iX = 0; iX < MAP_WIDTH; iX++) //Testa todos os blocos a depois do atual em X
286        {
287            Tile retangulo;
288            int index = iX + MAP_WIDTH*iY;
289            if (listaTiles[index].typeId != TILE_TIPO_PAREDE)
290            {
291                retangulo.typeId = listaTiles[index].typeId;
292                retangulo.posX = iX;
293                retangulo.posZ = iY;
294                listaTilesOptimizados.push_back(retangulo);
295                continue;
296            }
297
298             retangulo.top = true;
299             //Se parede, verifica fora de bordas
300             if (index-1 < 0)
301                 retangulo.left = true;
302             else //Se for chao, entao tem parede naquela direcao
303                 if (listaTiles[index-1].typeId != TILE_TIPO_PAREDE)
304                     retangulo.left = true;
305             if (index - MAP_WIDTH < 0)
306                 retangulo.back = true;
307             else //Se for chao, entao tem parede naquela direcao
308                 if (listaTiles[index - MAP_WIDTH].typeId != TILE_TIPO_PAREDE)
309                     retangulo.back = true;
310             if (index +1 >= (int)listaTiles.size())
311                 retangulo.right = true;
312             else //Se for chao, entao tem parede naquela direcao
313                 if (listaTiles[index +1].typeId != TILE_TIPO_PAREDE)
```

```
                    retangulo.right = true;
            if (index + MAP_WIDTH >= (int)listaTiles.size())
                retangulo.front = true;
            else //Se for chao, entao tem parede naquela direcao
                if (listaTiles[index + MAP_WIDTH].typeId != TILE_TIPO_PAREDE)
                    retangulo.front = true;

            retangulo.posX = iX;
            retangulo.posZ = iY;
            retangulo.typeId = listaTiles[index].typeId;

            listaTilesOptimizados.push_back(retangulo);

        }
    }
}



void Map::setWired(int wired)
{
    if (wired)
    {
        mostraWired = true;
        RENDER_MODE = GL_LINES;
    }
    else
    {
        mostraWired = false;
        RENDER_MODE = GL_QUADS;
    }

}
bool Map::isWire()
{
    return mostraWired;
}
```

A.2.5 Texture Loader

```
#include "textureloader.h"

#include <assert.h>
#include <fstream>

using namespace std;


Image::Image(char* ps, int w, int h) : pixels(ps), width(w), height(h) {

}

Image::~Image() {
    delete[] pixels;
}

namespace {
    //Converts a four-character array to an integer, using little-endian form
    int toInt(const char* bytes) {
        return (int)(((unsigned char)bytes[3] << 24) |
                     ((unsigned char)bytes[2] << 16) |
                     ((unsigned char)bytes[1] << 8) |
                     (unsigned char)bytes[0]);
    }

    //Converts a two-character array to a short, using little-endian form
    short toShort(const char* bytes) {
        return (short)(((unsigned char)bytes[1] << 8) |
                       (unsigned char)bytes[0]);
    }

    //Reads the next four bytes as an integer, using little-endian form
    int readInt(ifstream &input) {
        char buffer[4];
        input.read(buffer, 4);
        return toInt(buffer);
    }

    //Reads the next two bytes as a short, using little-endian form
    short readShort(ifstream &input) {
        char buffer[2];
        input.read(buffer, 2);
        return toShort(buffer);
    }

    //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 //sons
```

```
int SOUND_main = -1;
int SOUND_inter1 = -1;
int SOUND_inter2 = -1;
int SOUND_inter3 = -1;
int SOUND_attack = -1;
int SFX_die = -1;
int SFX_eat = -1;
int SFX_eat2 = -1;
int SFX_alert = -1;
//gameplay
int attack_mode = 0;
```

A.2.7 Eventos

```
#include "eventos.h"

#include "gamemanager.h"

#include "player.h"

void teclasNormais(unsigned char key, int x, int y)
{
    if (menuPrincipal)
        return; /// IGNORA ABAIXO

    int mod = glutGetModifiers();
    if (mod == GLUT_ACTIVE_SHIFT)
        Player::PlayerControl->setCorrer();
    else
        Player::PlayerControl->setAndar();

    switch(key)
    {
        case 27:  //ESC
            exit(0);
            break;
        case 'W':
        case 'w':
        {
            Player::PlayerControl->moveFrente(true);
            break;
        }
        case 'S':
        case 's':
        {

            Player::PlayerControl->moveTraz(true);
            break;
        }

        case 'A':
        case 'a':
            Player::PlayerControl->moveEsquerda(true);
            break;
        case 'D':
        case 'd':
            Player::PlayerControl->moveDireita(true);
            break;
        case 'Q':
        case 'q':
            Player::PlayerControl->giraEsquerda(true);
            break;
        case 'E':
        case 'e':
            Player::PlayerControl->giraDireita(true);
            break;
        case '2':
            Player::PlayerControl->giraCima(true);
            break;
        case '3':
            Player::PlayerControl->giraBaixo(true);
            break;
        case '1': // reseta angulo Y
            Camera::CameraControl.angleY = 0;
            Camera::CameraControl.calculaDirecao();
            break;
        case 'Z':
        case 'z':
            Camera::CameraControl.cameraY += 2;
            break;
        case 'X':
        case 'x':
            Camera::CameraControl.cameraY -= 2;
            break;
        case 'C':
        case 'c':
```

```
73              Camera::CameraControl.cameraX = 6;
74              break;
75          case 'V':
76          case 'v':
77              Camera::CameraControl.cameraY = 3;
78              break;
79          case 'B':
80          case 'b':
81              Camera::CameraControl.cameraZ = 6;
82              break;
83          case 'F':
84          case 'f':
85          {
86              GLboolean isFog = false;
87              glGetBooleanv(GL_FOG, &isFog);
88              if (isFog)
89                  glDisable(GL_FOG);
90              else
91                  glEnable(GL_FOG);
92
93              break;
94
95          }
96          case 'R':
97          case 'r':
98              if (FrameRate::FPSControl.isFPSCap())
99                  FrameRate::FPSControl.setFPSCap(false);
100             else
101                 FrameRate::FPSControl.setFPSCap(true);
102             break;
103         default:break;
104     }
105 }
106 void teclasNormaisUp(unsigned char key, int x, int y)
107 {
108     if (menuPrincipal)
109         return; /// IGNORA ABAIXO
110
111     switch(key)
112     {
113         case 'W':
114         case 'w':
115             Player::PlayerControl->moveFrente(false);
116             break;
117         case 'S':
118         case 's':
119             Player::PlayerControl->moveTraz(false);
120             break;
121         case 'A':
122         case 'a':
123             Player::PlayerControl->moveEsquerda(false);
124             break;
125         case 'D':
126         case 'd':
127             Player::PlayerControl->moveDireita(false);
128             break;
129         case 'Q': case 'q':
130             Player::PlayerControl->giraEsquerda(false);
131             break;
132         case 'E': case 'e':
133             Player::PlayerControl->giraDireita(false);
134             break;
135         case '2':
136             Player::PlayerControl->giraCima(false);
137             break;
138         case '3':
139             Player::PlayerControl->giraBaixo(false);
140             break;
141         default:break;
142
143     }
144 }
145
146 void teclasEspeciais(int key, int x, int y )
147 {
148     if (menuPrincipal)
149         return; /// IGNORA ABAIXO
150
151     switch(key)
152     {
153         case GLUT_KEY_UP: Player::PlayerControl->moveFrente(true); break;
154         case GLUT_KEY_DOWN: Player::PlayerControl->moveTraz(true); break;
155         case GLUT_KEY_LEFT: Player::PlayerControl->giraEsquerda(true); break;
156         case GLUT_KEY_RIGHT: Player::PlayerControl->giraDireita(true); break;
157         default: break;
158     }
```

```
159
160
161 }
162
163 void teclasEspeciaisSoltar(int key, int x, int y)
164 {
165     if (menuPrincipal)
166         return; /// IGNORA ABAIXO
167
168     switch(key)
169     {
170         case GLUT_KEY_UP: Player::PlayerControl->moveFrente(false); break;
171         case GLUT_KEY_DOWN: Player::PlayerControl->moveTraz(false); break;
172         case GLUT_KEY_LEFT: Player::PlayerControl->giraEsquerda(false); break;
173         case GLUT_KEY_RIGHT: Player::PlayerControl->giraDireita(false); break;
174         default: break;
175     }
176 }
177
178 void mouseButton(int button, int state, int x, int y)
179 {
180     if (menuPrincipal)
181     {
182         for(unsigned int i = 0; i < Button::ButtonList.size();i++)
183             Button::ButtonList[i]->handleMouse(button, state, x, y);
184         return; /// IGNORA ABAIXO
185     }
186
187     if (button == GLUT_LEFT_BUTTON)
188     {
189         if (state == GLUT_UP) //Reseta posicoes e ajusta deslocamento
190         {
191             Player::PlayerControl->setMouse(-1,-1);
192         }
193         else
194         {
195             Player::PlayerControl->setMouse(x,y);
196         }
197     }
198 }
199
200 void moveMouse(int x, int y)
201 {
202     if (menuPrincipal)
203         return; /// IGNORA ABAIXO
204
205     Player::PlayerControl->moveMouse(x,y);
206 }
```

A.2.8 Game Maneger

```
1 #include "gamemanager.h"
2 #include "eventos.h"
3 #include <time.h>
4 GameManager game;
5
6 void startButtonAction()
7 {
8     menuPrincipal = false;
9
10     game.resetPositions();
11
12     SoundAL sc;
13     sc.stopAll();
14     sc.play(SOUND_inter2);
15 }
16 void changeSize(int w, int h)
17 {
18     //Previne divisao por zero
19     if ( h == 0)
20         h = 1;
21
22     float ratio = w*1.0 / h;
23
24     //Usa matriz de projecao
25     glMatrixMode(GL_PROJECTION);
26     //Reseta matriz
27     glLoadIdentity();
28
29     //Arruma viewport para janela inteira
30     glViewport(0,0,w,h);
31
32     //Arruma a perspectiva correta
33     gluPerspective(45.0f, ratio, 1, GAME_FOV*TAMANHO_BLOCO);
34
35     //Volta para o modelView
```

```
36      glMatrixMode(GL_MODELVIEW);
37
38      wScreen = w;
39      hScreen = h;
40  }
41  void GameManager::inicializaRender(void)
42  {
43      //Transparencia
44      glBlendFunc(GL_SRC_ALPHA,GL_ONE);
45
46      glEnable(GL_LIGHTING); //Habilita luz
47      glEnable(GL_LIGHT0); //Habilita luz #0
48      glEnable(GL_LIGHT1); //Habilita luz #0
49      glEnable(GL_NORMALIZE); //Automatically normalize normals
50      glEnable(GL_COLOR_MATERIAL);
51      //glEnable(GL_LIGHT1); //Habilita luz #1
52
53      glEnable(GL_DEPTH_TEST);
54      glShadeModel(GL_SMOOTH); //Shading
55
56      glEnable(GL_CULL_FACE); //Reduz quantidade de triangulos desenhados.
57      glCullFace(GL_CW);
58
59      wallTexture = texture::loadTextureBMP("data/wall.bmp");
60      floorTexture = texture::loadTextureBMP("data/floor.bmp");
61
62
63  }
64  void GameManager::inicializa(void)
65  {
66      inicializaRender();
67      inicializaSons();
68
69  //---------------------------
70      //Especifica a cor de fundo
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      //Testes 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);
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     //Verifica mudanca de estados sobre a bola especial
169     if(attack_mode == 1) //notificou mudanca e toca musica
170     {
171         //Seta flag ESPECIAL ativa para todas as Entidades. Inclusive o 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; // reseta a flag player
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         //passados 3 segundos
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         //acabou o efeito da bola, 10 segundos + os 3 do sfx anterior
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; // reseta a flag player
206         }
207     }
```

```
208
209 }
210 void 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(50,130,8,"Derrota!!!");
229                 break;
230             case STATUS_NORMAL:
231                 txt::renderText2dOrtho(50,130,8,"Novo jogo!!!");
232                 break;
233             case STATUS_VITORIA:
234                 txt::renderText2dOrtho(50,130,8,"Vitoria!!!");
235                 break;
236                 default:;
237         }
238
239         return; /// IGNORA ABAIXO
240     }
241
242
243
244
245     //Iluminacao
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     //Fim Iluminacao
257
258
259     //Calcula iteracoes
260     this->loop();
261
262     //Imprime 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 //Quanda chamado cleanup durante o destructor ocorre falha de
293 //segmentacao somente no delete Entidade
```

```
294 GameManager::~GameManager()
295 {
296     sc.stopAll();
297     sc.exit();
298 }
299 void 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 }
311 void 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 }
350 void 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 }
362 int main(int argc, char* args[])
363 {
364 
365     //testOpenAL();
366     //testSoundALClass();
367 
368     game.executa(argc, args);
369     return 0;
370 }
371 void 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     //Entra no loop de processamento de eventos
396     glutMainLoop();
397 }
```

A.2.9 Text

```
1 #include "text.h"
2
3 namespace 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

```
#include "tile.h"

Tile::Tile()
{
    tamanho = TAMANHO_BLOCO;
    posY = 0;

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

A.2.11 Makefile

```
#############################################################
#                       Makefile
#############################################################
CC = g++
CFLAGS =$(GLFLAGS) -I./ -O3 -Os -g $(PROBLENS)

PROBLENS=-Wall -pedantic -fpermissive
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

all:*.cpp
    echo "System: "$(UNAME) "OS"
    echo -n "compiling..."
    $(CC) *.cpp -o prog $(CFLAGS)
    echo "ok"

clean:
    echo "cleaning..."
    rm -rfv prog *.o

run: all
    echo "Running..."
    ./prog

check:
    echo "Nothing to be check."

.SILENT:

#Obs
#
#   Bibliotecas incluidas:
#
#   alut-dev
#   openal-dev
```

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