

# 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 início 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 prática 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 várias pessoas editem o código simultaneamente.
- **Uso de ferramentas VCS/SVN**, permitindo vários backups e facilitando a construção de várias 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 várias paredes e corredores. Assim poderíamos levantar estruturas de colisões, movimentação, iluminação e texturas. De início, 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* conseguiu a incrível marca de 3,333,360 pontos, após vencer os consecutivos 256 níveis 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 está 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 <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 válido 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 auxílio da IDE *CodeBlocks*<sup>1</sup>. 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 válido 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 diretório 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 válido 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 diretório 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<sup>2</sup>:

- 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 <sup>3</sup>

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
- Arquivo 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  $\text{\LaTeX}$  apenas para os estudantes irem se orientando de como e o quê escrever. Assim, me isento de responsabilidade sobre o conteúdo deste texto. Dúvidas: [carla\(rocha.carla@gmail.com\)](mailto: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](mailto:ziuloliveira@gmail.com)



**Gustavo Jaruga Cruz**  
Matricula: 09/0066634  
E-mail: [darksshades@hotmail.com](mailto:darksshades@hotmail.com)



**Guilherme Fay Vergara**  
Matricula: 10/45547  
E-mail: [guifayvergara@hotmail.com](mailto:guifayvergara@hotmail.com)

## APÊNDICE A

### CÓDIGOS FONTES

#### A.1 Headers

##### A.1.1 Camera

```

1#ifndef _CAMERAS_H_
2#define _CAMERAS_H_
3
4#include "defines.h"
5
6
7#define CAMERA_ANDA 20
8#define CAMERA_CORRE 40
9
10class Camera
11{
12    public:
13        float lookX, lookY, lookZ;
14        float cameraX, cameraY, cameraZ;
15
16        float angleX, angleY;
17        float angleOffsetX, angleOffsetY;
18
19        float deltaAngleX, deltaAngleY;
20        float deltaMouseX, deltaMouseY;
21        float deltaMove, deltaMoveLado;
22
23        float velocidadeMove;
24        float velocidadeMoveAndar;
25        float velocidadeMoveCorre;
26        float velocidadeVira;
27        float velocidadeViraMouse;
28
29        int xOrigem, yOrigem;
30        unsigned int ticks;
31        unsigned int deltaTicks;
32    public:
33        Camera();
34        static Camera CameraControl;
35
36        void ajustaCamera(); //seta posicao e direcao da camera
37        void loop(); //ajusta timer
38        void reset();
39
40        void moveFrente(bool mover);
41        void moveTraz(bool mover);
42        void moveEsquerda(bool mover);
43        void moveDireita(bool mover);
44
45        void giraEsquerda(bool mover);
46        void giraDireita(bool mover);
47        void giraCima(bool mover);
48        void giraBaixo(bool mover);
49
50        void setMouse(int x, int y);
51        void moveMouse(int x, int y);
52        //temp como public
53        void calculaDirecao(void);
54
55        //Liga ou desliga correr
56        void setCorrer(void);
57        void setAndar(void);
58
59
60        void calculaMovimento(float delta);
61        void calculaMovimentoLateral(float delta);
62
63};
64#endif

```

##### A.1.2 Entidade

```

1
2#ifndef __ENTIDADE_H_
3#define __ENTIDADE_H_
4
5#include <vector>
6#include "vetor3d.h"
7#include "defines.h"
8#include "map.h"
9#include "camera.h"
10//List of flags
11enum

```

```

12{
13    ENTIDADE_FLAG_NENHUM =      0,
14    ENTIDADE_FLAG_GRAVIDADE =   0x00000001,
15    ENTIDADE_FLAG_GHOST =      0x00000002,
16    ENTIDADE_FLAG_GHOST_MAP =  0x00000004,
17    ENTIDADE_FLAG_TIRO =        0x00000008,
18    ENTIDADE_FLAG_PORTA =       0x00000016
19};
20
21
22class Entidade
23{
24    public:
25        static std::vector<Entidade*> EntidadeList;
26        Entidade();
27        virtual ~Entidade();
28    protected:
29        bool isColisaoObjeto(Entidade* objeto);
30        bool isColidido();
31        bool visible;
32        bool dead;
33        int delta;
34        std::vector<Entidade*> entidadeColidida;
35
36
37
38
39    public:
40        void addToEntidadeList();
41        Tile* isColisaoMapa(Vetor3D newPosicao, int type = TILE_TIPO_PAREDE);
42        void setColisao(Entidade* ent);
43        void setPosicao(float x, float y, float z);
44        //Ex: int delta = getTicks() - deltaTicks;
45        //Ex: posicao = posicao + (velocidade * (delta/1000.f) );
46        int deltaTicks; //ticks from last time that calculated the movement
47        Vetor3D posicao;
48        Vetor3D velocidade;
49        Vetor3D aceleracao;
50        Vetor3D maxVelocidade;
51        Vetor3D tamanho;
52        int flags;
53        bool showWired;
54    public:
55        bool isVisible();
56        void setTamanho(float newTamanho);
57    public:
58        void init();
59        void removeFromEntidadeList();
60
61
62        virtual bool carregaModelo(char* file);
63        virtual void loop();
64        virtual void render();
65        virtual void cleanup();
66        virtual void executaColisao();
67        virtual void testaColisao();
68
69
70};
71
72
73#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; //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        //Começa 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 __DEFINISS_H_
2#define __DEFINISS_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;
50
51
52

```

```
53#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
41 void Camera::reset()
42 {
43     Camera();
44     ticks = glutGet(GLUT_ELAPSED_TIME);
45 }
46
47
48 // Chamada internamente por Player.
49 void Camera::ajustaCamera()
50 {
51
52     if (deltaAngleX || deltaAngleY)
53         calculaDirecao();
54
55     gluLookAt( cameraX, cameraY, cameraZ,
56               cameraX+lookX, cameraY+lookY, cameraZ+lookZ,
57               0.0f, 1.0f, 0.0f);
58
59     ticks = glutGet(GLUT_ELAPSED_TIME);
60 }
61
62 void Camera::loop()
63 {
64     deltaTicks = glutGet(GLUT_ELAPSED_TIME) - ticks;
65 }
66
67 void Camera::calculaDirecao(void)
68 {
69     float fator = deltaTicks/1000.f;
70     angleX += deltaAngleX*fator;
71     angleY += deltaAngleY*fator;
72
73     // corrige angulo
74     if ( angleX+angleOffsetX >= 360 )
75         angleX -= 360;
76     if ( angleX+angleOffsetX < 0 )
77         angleX += 360;
78
79     // So permite rotacionar 180 graus em Y
80     if ( angleY+angleOffsetY >= 90 )
81         angleY = 90-angleOffsetY;
82     if ( angleY+angleOffsetY <= -90 )
83         angleY = -(90+angleOffsetY);
84
85
86     lookX = sin( (angleX+angleOffsetX)*M_PI/180);
87     lookZ = cos( (angleX+angleOffsetX)*M_PI/180);
88
89     lookY = sin( (angleY+angleOffsetY)*M_PI/180);
90 }
91 void Camera::calculaMovimento(float delta)
92 {
93     // Adiciona ao movimento
94     float fator = deltaTicks/1000.f;
95
96     // Fator delta vezes direcao. 0.1f para ajustar velocidade.
97     cameraX += (delta*fator) * lookX;
98     cameraZ += (delta*fator) * lookZ;
99 }
100 void Camera::calculaMovimentoLateral(float delta)
101 {
102     float fator = deltaTicks/1000.f;
103
104     float lateralX = sin( (angleX-90)*M_PI/180);
105     float lateralZ = cos( (angleX-90)*M_PI/180);
106     // Adiciona ao movimento
107     // Fator delta vezes direcao. 0.1f para ajustar velocidade.
108     cameraX += (delta*fator) * (lateralX);
109     cameraZ += (delta*fator) * (lateralZ);
110 }
111
112
113 void Camera::moveFrente(bool mover)
114 {
115     if(mover)
116         deltaMove = velocidadeMove;
117     else
118         deltaMove = 0.0f;
119 }
120 void Camera::moveTraz(bool mover)

```

```
121{
122    if(mover)
123        deltaMove = -velocidadeMove;
124    else
125        deltaMove = 0.0f;
126}
127}
128void Camera::moveEsquerda(bool mover)
129{
130    if(mover)
131        deltaMoveLado = -velocidadeMove;
132    else
133        deltaMoveLado = 0.0f;
134}
135void Camera::moveDireita(bool mover)
136{
137    if(mover)
138        deltaMoveLado = velocidadeMove;
139    else
140        deltaMoveLado = 0.0f;
141}
142}
143void Camera::giraEsquerda(bool mover)
144{
145    if(mover)
146        deltaAngleX = velocidadeVira;
147    else
148        deltaAngleX = 0.0f;
149}
150void Camera::giraDireita(bool mover)
151{
152    if(mover)
153        deltaAngleX = -velocidadeVira;
154    else
155        deltaAngleX = 0.0f;
156}
157void Camera::giraCima(bool mover)
158{
159    if(mover)
160        deltaAngleY = velocidadeVira;
161    else
162        deltaAngleY = 0.0f;
163}
164void Camera::giraBaixo(bool mover)
165{
166    if(mover)
167        deltaAngleY = -velocidadeVira;
168    else
169        deltaAngleY = 0.0f;
170}
171}
172void Camera::setMouse(int x, int y)
173{
174    xOrigem = x;
175    yOrigem = y;
176}
177    if (xOrigem == -1) // Ambos serao -1 necessariamente
178    {
179        angleX +=angleOffsetX;
180        angleY +=angleOffsetY;
181        angleOffsetX = 0;
182        angleOffsetY = 0;
183    }
184}
185void Camera::moveMouse(int x, int y)
186{
187    deltaMouseX = deltaMouseY = 0;
188    //Se houve deslocamento
189    if (xOrigem>0)
190    {
191        angleOffsetX = (xOrigem-x) * 0.1f;
192    }
193    if (yOrigem>0)
194    {
195        angleOffsetY = (yOrigem-y) * 0.1f;
196    }
197    calculaDirecao();
198}
199}
200void Camera::setCorrer(void)
201{
202    velocidadeMove = velocidadeMoveCorre;
203}
204void Camera::setAndar(void)
205{
206    velocidadeMove = velocidadeMoveAndar;
```

207}

## A.2.2 Entidade

```

1#include "entidade.h"
2
3#include <stdlib.h>
4
5
6
7
8//=====
9// Variaveis estaticas
10//=====
11std::vector<Entidade*> Entidade::EntidadeList;
12
13//=====
14// Construtores
15//=====
16Entidade::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    maxVelocidade.x = maxVelocidade.y = maxVelocidade.z = 50.f;
28    entidadeColidida.clear();
29
30}
31
32void Entidade::init()
33{
34    deltaTicks = glutGet(GLUT_ELAPSED_TIME);
35}
36Entidade::~Entidade()
37{
38}
39void Entidade::cleanup()
40{
41    //removeFromEntidadeList();
42}
43bool Entidade::isColisaoObjeto(Entidade* objeto)
44{
45    //Nota, o ponto posicao marca 0.... ex: posicao 0 comeco do bloco final do bloco em x,y,z
46    //Tal que y mais abaixo = y e y mais alto = y+tamanhoY
47    int baixo1 = this->posicao.y;
48    int cima1 = this->posicao.y + this->tamanho.y;
49    int esquerdal = this->posicao.x;
50    int direital = this->posicao.x + this->tamanho.x;
51    int frentel = this->posicao.z;
52    int traz1 = this->posicao.z + this->tamanho.z;
53
54    int baixo2 = objeto->posicao.y;
55    int esquerda2 = objeto->posicao.x;
56    int frente2 = objeto->posicao.z;
57    int direita2 = objeto->posicao.x + objeto->tamanho.x;
58    int cima2 = objeto->posicao.y + objeto->tamanho.y;
59    int traz2 = objeto->posicao.z + objeto->tamanho.z;
60
61    if (
62        !(baixo1 > cima2) &&
63        !(cima1 < baixo2) &&
64        !(esquerdal > direita2) &&
65        !(direital < esquerda2) &&
66        !(frentel > traz2) &&
67        !(traz1 < frente2)
68    )
69    {
70        return true;
71    }
72
73    return false;
74
75}
76//=====
77// Retorna true se estiver colidindo com o mapa
78//=====
79Tile* Entidade::isColisaoMapa(Vetor3D newPosicao, int type)
80{
81    //Calcula o Id do tile que deve ser testado
82    //Ex: X = 5 tal que startX = 0,41 = 0 endX = 1,3 = 1

```

```

83  int startX = (newPosicao.x) / TAMANHO_BLOCO;
84  int startZ = (newPosicao.z) / TAMANHO_BLOCO;
85  int endX = (newPosicao.x + (tamanho.x)) / TAMANHO_BLOCO;
86  int endZ = (newPosicao.z + (tamanho.z)) / TAMANHO_BLOCO;
87
88  //Checa colisoes com os tiles
89  for(int iZ = startZ; iZ <= endZ; iZ++) {
90      for(int iX = startX; iX <= endX; iX++) {
91          Tile* bloco = Map::MapControl(iX, iZ);
92
93          if(
94              (bloco->typeId == type) &&
95              (posicao.y < (bloco->posY+bloco->tamanho) ) &&
96              ((posicao.y+tamanho.y) > bloco->posY)
97          )
98              return bloco;
99      }
100 }
101 return 0;
102}
103
104void Entidade::removeFromEntidadeList()
105{
106    for(unsigned int i = 0; i < EntidadeList.size(); i++)
107    {
108        if (EntidadeList[i] == this)
109            EntidadeList.erase(EntidadeList.begin()+i);
110    }
111}
112void Entidade::addToEntidadeList()
113{
114
115
116    for(unsigned int i = 0; i < EntidadeList.size(); i++)
117    {
118        if (EntidadeList[i] == this)
119            return; //Se ja estiver na lista, retorna
120    }
121
122    EntidadeList.push_back(this);
123}
124
125bool Entidade::carregaModelo(char* file){return true;}
126//=====
127// Executa acoes do loop, aceleracao, velocidade.
128//=====
129void Entidade::loop()
130{
131    if(dead) return;
132    //deltaTicks reseta o render
133    delta = glutGet(GLUT_ELAPSED_TIME) - deltaTicks;
134    float fator = delta/1000.f;
135
136    if (flags & ENTIDADE_FLAG_GRAVIDADE)
137        aceleracao.y = -15.f; // sistemas de coordenadas do OpenGL -y baixo
138
139    //Calcula aceleracoes
140    if ( velocidade.x + aceleracao.x <= maxVelocidade.x)
141        velocidade.x += (aceleracao.x * fator);
142    if ( velocidade.y + aceleracao.y <= maxVelocidade.y)
143        velocidade.y += (aceleracao.y * fator);
144    if ( velocidade.z + aceleracao.z <= maxVelocidade.z)
145        velocidade.z += (aceleracao.z * fator);
146
147    Vetor3D newPosicao = posicao + (velocidade * fator );
148
149    if (isColisaoMapa(newPosicao) == false)
150        posicao = newPosicao;
151    else
152    {
153        velocidade.x = 0;
154        velocidade.z = 0;
155        aceleracao.x = 0;
156        aceleracao.z = 0;
157        int pos = (int)(rand() % 4);
158        switch(pos)
159        {
160            case 0:
161                aceleracao.x = 20;break;
162            case 1:
163                aceleracao.x = -20;break;
164            case 2:
165                aceleracao.z = 20;break;
166            case 3:
167                aceleracao.z = -20;break;
168            default;;

```

```

169     }
170
171 }
172
173 deltaTicks = glutGet(GLUT_ELAPSED_TIME);
174}
175void Entidade::render()
176{
177     int tamanhoCubo = tamanho.x; //Temp, enquanto utilizar glutCube
178     glPushMatrix();
179     //Centraliza devido ao GLUT
180     glColor3f(1.0f, 0.0f, 0.0f);
181     glTranslated(posicao.x+tamanho.x/2,
182                 posicao.y+tamanho.y/2,
183                 posicao.z+tamanho.z/2);
184     if (showWired)
185         glutWireCube(tamanhoCubo);
186     else
187         glutSolidCube(tamanhoCubo);
188     glPopMatrix();
189
190
191}
192void Entidade::testaColisao()
193{
194     if(dead) return;
195
196     unsigned int thisID = -1;
197     for (unsigned int i = 0; i < EntidadeList.size(); i++)
198         if (EntidadeList[i] == this)
199             {
200                 thisID = i;
201                 break;
202             }
203     //Testa com todas as entidades desta para frente.
204     //Ex: lista: 1 2 3 4
205     // thisID =1, testa com 2, 3 , 4
206     // thisID = 2 testa com 3, 4      desta, forma, thisID = 2 nao testa colisoes com 1 pois ja foi testado anteriormente.
207     for (unsigned int i = thisID+1; i < EntidadeList.size(); i++)
208     {
209         if (EntidadeList[i] != this && !EntidadeList[i]->dead)
210         {
211             if(isColisaoObjeto(EntidadeList[i]) )
212             { //adiciona colisoes tanto neste elemento quanto no testado
213                 setColisao(EntidadeList[i]);
214                 EntidadeList[i]->setColisao(this);
215             }
216         }
217     }
218}
219//Seta colisao atraves de metodo publico
220void Entidade::setColisao(Entidade* ent)
221{
222     entidadeColidida.push_back(ent);
223}
224bool Entidade::isColidido()
225{
226     if (entidadeColidida.size() == 0)
227         return false;
228     else
229         return true;
230}
231void Entidade::executaColisao()
232{
233     if ( !isColidido() )
234         return; // sem colisoes
235
236     //Volta o que tinha movido.
237     float fator = delta/1000.f;
238     posicao = posicao - (velocidade * fator );
239     //Para, e vai na direcao oposta
240     velocidade.x = 0;
241     velocidade.z = 0;
242     aceleracao.x = -aceleracao.x;
243     aceleracao.z = -aceleracao.z;
244
245     entidadeColidida.clear();
246}
247
248bool Entidade::isVisible()
249{
250     return visible;
251}
252void Entidade::setTamanho(float newTamanho)
253{
254     tamanho.x = tamanho.y = tamanho.z = newTamanho;

```

```

255}
256void Entidade::setPosicao(float x, float y, float z)
257{
258    posicao.x = x;
259    posicao.y = y;
260    posicao.z = z;
261}

```

### A.2.3 Framerate

```

1#include "framerate.h"
2
3
4FrameRate FrameRate::FPSControl;
5
6
7
8float FrameRate::getFPS()
9{
10    return fps;
11}
12void FrameRate::setFPSCap(bool cap)
13{
14    fpsCap = cap;
15}
16bool FrameRate::isFPSCap()
17{
18    return fpsCap;
19}
20FrameRate::FrameRate()
21{
22    ticks = glutGet(GLUT_ELAPSED_TIME);
23    ticksControl = glutGet(GLUT_ELAPSED_TIME);
24    frames = 0;
25    fps = 0;
26    fpsCap = false;
27}
28
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//Usado pra outras classes obterem info sobre o mapa.
4Map Map::MapControl;
5
6
7//Pega o Tile na posicao x,y do mapa.
8//Ex: Map 1 2 3    vector sera 1 2 3 4 5 6
9//      4 5 6
10Tile* Map::getTile(int x, int y)
11{
12    unsigned int ID = 0;
13
14    ID = (y * MAP_WIDTH) + x;
15
16    return &listaTilesOptimizados[ID];
17}
18inline int Map::getX(int i)
19{

```

```

20     return i % MAP_WIDTH;
21 }
22 inline int Map::getY(int i)
23 {
24     return (int) i/MAP_WIDTH;
25 }
26
27 Map::Map()
28 {
29     origemX = -TAMANHO_BLOCO;
30     origemZ = -TAMANHO_BLOCO;
31     mostraWired = false;
32     RENDER_MODE = 0x0007; //GL_QUADS
33 }
34
35 void Map::renderBloco(float width, float height, float flatness, bool left,
36     bool right, bool front, bool back, bool top, int TYPE = GL_QUADS)
37 {
38     float w = width/2;
39     float h = height/2;
40     float f = flatness/2;
41
42     float xTexNumber = width/TAMANHO_BLOCO;
43
44     glEnable(GL_TEXTURE_2D);
45     glBindTexture(GL_TEXTURE_2D, wallTexture);
46     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
47     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
48
49
50     glBegin(TYPE);
51     //Front
52     if(front)
53     {
54         glNormal3f(0.0f, 0.0f, 1.0f);
55         //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(1.0f, 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
156void 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}
194void 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        glTexParameterf(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
248int 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
279void Map::geraQuadradosOptimizados()
280{
281    for(int iY = 0; iY < MAP_HEIGHT; iY++)
282    {
283        for(int iX = 0; iX < MAP_WIDTH; iX++) //Testa todos os blocos a depois do atual em X
284        {
285            Tile retangulo;
286            int index = iX + MAP_WIDTH*iY;
287            if (listaTiles[index].typeId != TILE_TIPO_PAREDE)
288            {
289                retangulo.typeId = listaTiles[index].typeId;
290                retangulo.posX = iX;
291                retangulo.posZ = iY;
292                listaTilesOptimizados.push_back(retangulo);
293                continue;
294            }
295
296            retangulo.top = true;
297            //Se parede, verifica fora de bordas
298            if (index-1 < 0)
299                retangulo.left = true;
300            else //Se for chao, entao tem parede naquela direcao
301                if (listaTiles[index-1].typeId != TILE_TIPO_PAREDE)
302                    retangulo.left = true;
303            if (index - MAP_WIDTH < 0)
304                retangulo.back = true;
305            else //Se for chao, entao tem parede naquela direcao
306                if (listaTiles[index - MAP_WIDTH].typeId != TILE_TIPO_PAREDE)
307                    retangulo.back = true;
308            if (index +1 >= (int)listaTiles.size())
309                retangulo.right = true;
310            else //Se for chao, entao tem parede naquela direcao
311                if (listaTiles[index +1].typeId != TILE_TIPO_PAREDE)
312                    retangulo.right = true;
313            if (index + MAP_WIDTH >= (int)listaTiles.size())
314                retangulo.front = true;
315            else //Se for chao, entao tem parede naquela direcao
316                if (listaTiles[index + MAP_WIDTH].typeId != TILE_TIPO_PAREDE)
317                    retangulo.front = true;
318
319            retangulo.posX = iX;
320            retangulo.posZ = iY;
321            retangulo.typeId = listaTiles[index].typeId;
322
323            listaTilesOptimizados.push_back(retangulo);
324        }
325    }
326 }
327
328
329
330
331void Map::setWired(int wired)
332{
333    if (wired)
334    {
335        mostraWired = true;
336        RENDER_MODE = GL_LINES;
337    }
338    else
339    {
340        mostraWired = false;
341        RENDER_MODE = GL_QUADS;
342    }
343 }
344
345bool Map::isWire()
346{
347    return mostraWired;
348 }

```

## A.2.5 Texture Loader

```

1#include "textureloader.h"
2
3#include <assert.h>
4#include <fstream>
5
6using namespace std;
7
8
9Image::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(istream &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(istream &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         }

```

```

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
3float wScreen = SCREEN_WIDTH;
4float hScreen = SCREEN_HEIGHT;
5
6bool menuPrincipal = false;
7GLuint wallTexture;
8GLuint floorTexture;

```

### A.2.7 Eventos

```

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

```

```

54         Player::PlayerControl.giraCima(true);
55         break;
56     case '3':
57         Player::PlayerControl.giraBaixo(true);
58         break;
59     case '1': // reseta angulo Y
60         Camera::CameraControl.angleY = 0;
61         Camera::CameraControl.calculaDirecao();
62         break;
63     case 'Z':
64     case 'z':
65         Camera::CameraControl.cameraY += 2;
66         break;
67     case 'X':
68     case 'x':
69         Camera::CameraControl.cameraY -= 2;
70         break;
71     case 'C':
72     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}
106void 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.moveTras(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.moveTras(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 Manager

```

1#include "gamemanager.h"
2#include "eventos.h"
3
4GameManager game;
5
6void startButtonAction()
7{
8    menuPrincipal = false;
9    for(unsigned int i = 0;i < Entidade::EntidadeList.size(); i++)
10        Entidade::EntidadeList[i]->init();
11}
12void changeSize(int w, int h)
13{
14    //Previne divisao por zero
15    if ( h == 0)
16        h = 1;

```

```

17
18     float ratio = w*1.0 / h;
19
20     //Usa matriz de projecao
21     glMatrixMode(GL_PROJECTION);
22     //Reseta matriz
23     glLoadIdentity();
24
25     //Arruma viewport para janela inteira
26     glViewport(0,0,w,h);
27
28     //Arruma a perspectiva correta
29     gluPerspective(45.0f, ratio, 1, GAME_FOV*TAMANHO_BLOCO);
30
31     //Volta para o modelView
32     glMatrixMode(GL_MODELVIEW);
33
34     wScreen = w;
35     hScreen = h;
36 }
37 void GameManager::inicializaRender(void)
38 {
39     //Transparencia
40     glBlendFunc(GL_SRC_ALPHA, GL_ONE);
41
42     glEnable(GL_LIGHTING); //Habilita luz
43     glEnable(GL_LIGHT0); //Habilita luz #0
44     glEnable(GL_LIGHT1); //Habilita luz #0
45     glEnable(GL_NORMALIZE); //Automatically normalize normals
46     glEnable(GL_COLOR_MATERIAL);
47     //glEnable(GL_LIGHT1); //Habilita luz #1
48
49     glEnable(GL_DEPTH_TEST);
50     glShadeModel(GL_SMOOTH); //Shading
51
52     glEnable(GL_CULL_FACE); //Reduz quantidade de triangulos desenhados.
53     glCullFace(GL_CW);
54
55     wallTexture = texture::loadTextureBMP("data/wall.bmp");
56     floorTexture = texture::loadTextureBMP("data/floor.bmp");
57
58
59 }
60 void GameManager::inicializa(void)
61 {
62     inicializaRender();
63 //-----
64     //Especifica a cor de fundo
65     glClearColor(0.3f,0.3f,0.9f,1.0f);
66
67
68
69
70
71     GLfloat fog_color[4] = {0.0f,0.0f,0.0f,1.0};
72     glFogfv(GL_FOG_COLOR, fog_color);
73     glFogf(GL_FOG_DENSITY, 0.35f);
74
75     glFogi(GL_FOG_MODE, GL_LINEAR);
76     glHint(GL_FOG_HINT, GL_DONT_CARE);
77     glFogf(GL_FOG_START, TAMANHO_BLOCO*4.0f);
78     glFogf(GL_FOG_END, TAMANHO_BLOCO*10.0f);
79     glEnable(GL_FOG);
80
81     Map::MapControl.load((char*) "map_pacman_new.txt");
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     //testes
96     Entidade* enemy1 = new Entidade();
97     Entidade* enemy2 = new Entidade();
98
99
100     enemy1->init();
101     enemy1->posicao.x = 12*4;
102     enemy1->posicao.y = 0;

```



```

103     enemy1->posicao.z = 12*1;
104
105     enemy1->aceleracao.x = 10.f;
106     enemy1->aceleracao.z = 0.2f;
107
108     enemy1->setTamanho(5);
109     enemy1->addToEntidadeList();
110     //
111     enemy2->init();
112     enemy2->posicao.x = 12*3;
113     enemy2->posicao.y = 0;
114     enemy2->posicao.z = 12*1;
115
116     enemy2->aceleracao.x = 15.f;
117     enemy2->aceleracao.z = 4.2f;
118
119     enemy2->setTamanho(5);
120     enemy2->addToEntidadeList();
121
122     Player::PlayerControl.init();
123     Player::PlayerControl.addToEntidadeList();
124
125 }
126 void desenhaTela(void)
127 {
128
129     game.render();
130
131
132     glutSwapBuffers();
133 }
134
135 void GameManager::loop(void)
136 {
137
138     FrameRate::FPSControl.loop();
139     for(unsigned int i = 0; i < Entidade::EntidadeList.size(); i++)
140     {
141         Entidade::EntidadeList[i]->loop();
142     }
143     for(unsigned int i = 0; i < Entidade::EntidadeList.size(); i++)
144     {
145         Entidade::EntidadeList[i]->testaColisao();
146     }
147     for(unsigned int i = 0; i < Entidade::EntidadeList.size(); i++)
148     {
149         Entidade::EntidadeList[i]->executaColisao();
150     }
151
152 }
153 void GameManager::render(void)
154 {
155
156     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
157
158     glMatrixMode(GL_MODELVIEW);
159     glLoadIdentity();
160
161     if (menuPrincipal)
162     {
163         for(unsigned int i = 0; i < Button::ButtonList.size(); i++)
164             Button::ButtonList[i]->render();
165
166         txt::renderText2dOrtho(30,150,8,"Aperte o grande quadrado branco para comecar!!!");
167
168         return; /// IGNORA ABAIXO
169     }
170
171
172
173
174     //Iluminacao
175     GLfloat ambientLight[] = {0.1f, 0.1f, 0.1f, 1.0f};
176     glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientLight);
177     GLfloat directedLight[] = {0.7f, 0.7f, 0.7f, 0.0f};
178     GLfloat directedLightPos[] = {0.0f, 20.0f, -20.0f, 1.0f};
179     GLfloat light[] = {0.9f, 0.9f, 0.9f, 1.0f};
180     GLfloat lightPos[] = {100.0f, 30.0f, -10.0f, 1.0f};
181     glLightfv(GL_LIGHT0, GL_DIFFUSE, directedLight);
182     glLightfv(GL_LIGHT0, GL_POSITION, directedLightPos);
183     glLightfv(GL_LIGHT1, GL_DIFFUSE, light);
184     glLightfv(GL_LIGHT1, GL_POSITION, lightPos);
185     //Fim Iluminacao
186
187
188     //Calcula iteracoes

```

```

189     this->loop();
190
191     //Imprime SOL's
192     glPushMatrix();
193         glColor3f(1.0f, 1.0f, 1.0f);
194         glTranslatef(directedLightPos[0],directedLightPos[1],directedLightPos[2]);
195         glutSolidSphere(10.0f, 18.0f, 18.0f);
196     glPopMatrix();
197     glPushMatrix();
198         glColor3f(1.0f, 0.0f, 0.0f);
199         glTranslatef(lightPos[0],lightPos[1],lightPos[2]);
200         glutSolidSphere(10.0f, 18.0f, 18.0f);
201     glPopMatrix();
202
203     Map::MapControl.render();
204     //unsigned int temp = Entidade::EntidadeList.size();
205     for(unsigned int i = 0; i < Entidade::EntidadeList.size(); i++)
206     {
207         if (Entidade::EntidadeList[i]->isVisible())
208             Entidade::EntidadeList[i]->render();
209     }
210
211     txt::renderText2dOrtho(10,15,0,"FPS: %.2f",FrameRate::FPSControl.getFPS());
212
213
214
215     MiniMap::renderMiniMap();
216
217
218 }
219
220 //Quanda chamado cleanup durante o destructor ocorre falha de
221 //segmentacao somente no delete Entidade
222 GameManager::~~GameManager()
223 {
224 }
225 void cleanup(void)
226 {
227     printf("Entidade cleanup size: %lu\n", Entidade::EntidadeList.size());
228     for(unsigned int i = 0; i < Entidade::EntidadeList.size(); i++)
229         delete Entidade::EntidadeList[i];
230     printf("Button cleanup size: %lu\n", Button::ButtonList.size());
231     for(unsigned int i = 0; i < Button::ButtonList.size(); i++)
232         delete Button::ButtonList[i];
233 }
234 void testOpenAL()
235 {
236     unsigned int g_buf = -1;
237     unsigned int g_src = -1;
238
239     if(!alutInit(NULL, NULL))
240     {
241         printf("%s",alutGetErrorString(alutGetError()));
242         return;
243     }
244     alutGetError();
245     alutGetError();
246
247     g_buf = alutCreateBufferFromFile("testing.wav");
248
249     if (alutGetError() != ALUT_ERROR_NO_ERROR)
250     {
251         alDeleteBuffers(1, &g_buf);
252         alutExit();
253         return;
254     }
255
256     alGenSources(1, &g_src);
257
258     if(alutGetError() != AL_NO_ERROR)
259     {
260         alDeleteBuffers(1, &g_buf);
261         alDeleteSources(1, &g_src);
262         alutExit();
263         return;
264     }
265
266     alSourcei(g_src, AL_BUFFER, g_buf);
267
268     alSourcePlay(g_src);
269     alutSleep(4.0f);
270
271     alutExit();
272 }
273 void testSoundALClass()
274 {

```

```

275     SoundAL sn;
276     sn.init();
277
278     int m_i = sn.loadSound("testing.wav", 1);
279     sn.play(m_i);
280
281     alutSleep(4.0f);
282
283     sn.exit();
284 }
285 int main(int argc, char* args[])
286 {
287     testOpenAL();
288     testSoundALClass();
289
290     game.executa(argc, args);
291     return 0;
292 }
293
294 void GameManager::executa(int argc, char* args[])
295 {
296     glutInit(&argc, args);
297     glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
298     glutInitWindowPosition(100,100);
299     glutInitWindowSize(SCREEN_WIDTH, SCREEN_HEIGHT);
300     glutCreateWindow("Labirinth");
301
302     inicializa();
303
304     glutDisplayFunc(desenhaTela);
305     glutReshapeFunc(changeSize);
306     glutIdleFunc(desenhaTela);
307
308     glutKeyboardFunc(teclasNormais);
309     glutKeyboardUpFunc(teclasNormaisUp);
310     glutSpecialFunc(teclasEspeciais);
311     glutSpecialUpFunc(teclasEspeciaisSoltar);
312     glutMotionFunc(moveMouse);
313     glutMouseFunc(mouseButton);
314
315     atexit(cleanup);
316
317     glutIgnoreKeyRepeat(0);
318     //Entra no loop de processamento de eventos
319     glutMainLoop();
320 }

```

### 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
3Tile::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#####
4CC = g++
5CFLAGS = $(GLFLAGS) -I./ -O3 -Os -g $(PROBLEMS)
6
7PROBLEMS=-Wall -pedantic -fpermissive
8UNAME = $(shell uname)
9ifeq ($(UNAME),Linux) # Linux OS
10    GLFLAGS = -lglut -lglui -lGLU -lGL -lalut -lopenal
11    else
12    ifeq ($(UNAME),Darwin) # MAC OS X
13        GLFLAGS = -framework OpenGL -framework GLUT
14    else #Windows
15        GLFLAGS = -lopengl32 -lglu32 -lglut32 -lglee -lalut
16    endif
17endif
18
19all:*.cpp
20    echo "System: "$(UNAME) "OS"
21    echo -n "compiling..."
22    $(CC) *.cpp -o prog $(CFLAGS)
23    echo "ok"
24
25clean:
26    echo "cleaning..."
27    rm -rfv prog *.o
28
29run: all
30    echo "Running..."
31    ./prog
32
33check:
34    echo "Nothing to be check."
35
36.SILENT:
37
38#Obs
39#
40# Bibliotecas incluidas:
41#
42# alut-dev
43# 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**