# 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. Será abrangido conhecimentos de programação C++ e Makefile. Algumas dicas de como gerar um código que seja multiplataforma. Por fim teremos também alguns comentários sobre algumas ferramentas uteis para grupos de programação, tais como SVN e Valgrind. O projeto se diferencia dos demais por conter também funções de áudio, com o uso da biblioteca OpenAL.

### 1.1 A Historia

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

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

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

A aparência de Puck-Man continuou a evoluir por mais de um ano. Uma grande quantidade de tempo e esforço foi feito para desenvolver os fantasmas padrões de movimentos únicos através do labirinto e aprimorando as variáveis do jogo de dificuldade, como placas foram apuradas. Símbolos de bônus (incluindo o carro-chefe Galaxian) foram adicionados à mistura, em algum momento, e os fantasmas foram finalmente nomeados: Akabei, Pinky, Aosuke, e Guzuta. Efeitos sonoros e música foram alguns dos toques finais adicionados, com o desenvolvimento se aproximando do fim, eram feitos ajustes constante do comportamento dos fantasmas. Tornando-se por fim como apresentado na figura 1.

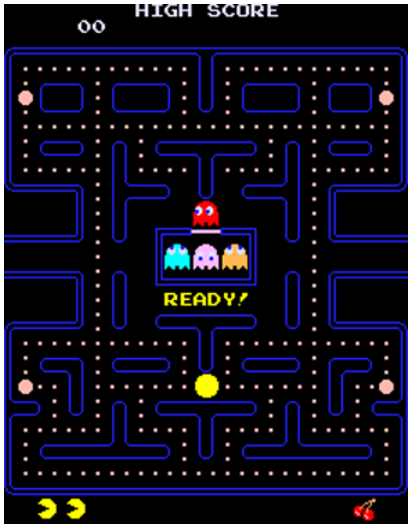



**Figura 1:** Pac-Man.
O clássico dos anos 80 só foi ter um score perfeito - máximo de pontos, sem falhas ou mortes - em 1999, quando *Billy Mitchell* consegui a incrível marca de $3,333,360$ pontos, após vencer os consecutivos 256 leveis do jogo.

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

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

## 1.2 Os fantasmas e seus comportamentos

Ao final da implementação do jogo original, os fantasmas ganharam características e personalidades, possuindo cada um uma AI distinta. Essa é provavelmente uma das ultimas implementações que será realizada neste nosso jogo - se vier a ser implementada. No jogo original, haviam apenas quatro fantasmas, dos quais falaremos um pouco mais sobre eles.

### *1.2.1 Blinky*

O fantasma vermelho é apropriadamente descrito como o de uma sombra e é mais conhecido como "Blinky". No Japão, seu personagem é representado pelo oikake palavra, que significa "correr para baixo ou prosseguir". Blinky parece ser sempre o primeiro dos fantasmas para acompanhar o Pac-Man no labirinto. Ele é de longe o mais agressivo dos quatro e vai obstinadamente buscar Pac-Man uma vez atrás dele.

Todos os fantasmas movem-se com a mesma taxa de velocidade quando se inicia um nível, mas Blinky irá aumentar a sua taxa de velocidade duas vezes por nível baseado no número de pontos que permanecem no labirinto. Enquanto neste estado acelerado, Blinky é comumente chamado de *"Cruise Elroy"*, mas ninguém parece saber onde esse costume se originou ou o que significa. No primeiro nível, por exemplo, Blinky torna-se Elroy quando existem 20 pontos remanescentes no labirinto, vindo a ser tão rápido como Pac-Man. [1]

### *1.2.2 Pinky*

Apelidado de "Pinky", o fantasma rosa é descrito como alguém que é rápido. No Japão, ele é caracterizado como machibuse, que significa "para realizar uma emboscada", talvez porque Pinky sempre parece ser capaz de chegar à frente de você e pega-lo quando você menos espera. Ele se move sempre à mesma velocidade como Inky e Clyde, no entanto, o que sugere que "rápido"é uma má tradução do machibuse . Pinky e Blinky muitas vezes parecem estar trabalhando em conjunto para encurralar Pac-Man, deixando-o sem ter para onde correr. [1]

No modo perseguição, Pinky se comporta assim, porque ele não tem como alvo o Pac-Man diretamente. Em vez disso, ele seleciona um deslocamento quatro peças adiante de Pac-Man na direção em que Pac-Man está se movimentando. Porém o jogo original carregava um bug. Se o Pac-Man estivesse se movimentando para cima, Pinky não apenas quatro posições para cima, mas também quatro posições para a esquerda, mirando assim a uma distancia

de $\sqrt{32}$ posições na diagonal superior esquerda de Pac-Man. Este bug ocorre devido a um problema de overflow, como pode ser observado no trexo abaixo onde a rotina de busca de Pinky é transcrita.

*Subrotina de alvo de Pinky [2]*

```
; load DE with Pac-man's position
278E ED5B394D    LD  DE,(#4D39)
; load HL with Pac-man's direction vector
2792 2A1C4D      LD  HL,(#4D1C)
; double Pac-man's direction vector
2795 29          ADD HL, HL
; quadruple Pac-man's direction vector
2796 29          ADD HL, HL
; add result to Pac-Man's position to give target
2797 19          ADD HL, DE
```

Em todas as demais direções, o vetor do Pac-man possui apenas uma coordenada não nula, porém quando quando esta subindo, este vetor recebe o valor $(1, -1)$, assim, HL passa a ter como valor final um vetor de valor $(4, -4)$.

#### 1.2.3 Inky

O fantasma azul é apelidado de "Inky"e seu personagem é descrito como alguém que é tímido. No Japão, ele é retratado como Kimagure, que significa "um temperamento inconstante, temperamental, ou irregular". Talvez não surpreendentemente, Inky é o menos previsível dos fantasmas. Às vezes, ele persegue agressivamente Pac-Man como Blinky, outras vezes ele salta à frente de Pac-Man como Pinky faria. Ele pode até mesmo vagar como Clyde na ocasião! Na verdade, Inky pode ser o fantasma mais perigoso de todos, devido ao seu comportamento errático. [1]

#### 1.2.4 Clyde

O fantasma laranjado é apelidado de "Clyde"e é caracterizado como aquele que é chato. No Japão, seu personagem é descrito como otoboke, ou seja, "ignorância fingindo", e seu apelido é "Guzuta", que significa "aquele que fica para trás". Na realidade, Clyde se move na mesma velocidade que Inky e Pinky, então sua descrição do personagem é um pouco erronea. Clyde é o fantasma último a deixar a caneta (local onde os fantasmas começam) e tende a separar-se dos outros fantasmas por se afastando do Pac-Man e fazer sua própria lógica, quando ele não está patrulhando seu canto do labirinto. Apesar de não ser tão perigosos quanto os outros três fantasmas, o seu comportamento pode parecer imprevisível, e ainda deve ser considerado uma ameaça.

Durante o modo perseguição, Clyde muda sua lógica com base em sua proximidade com Pac-Man. Ele primeiro calcula a distância euclidiana entre sua posição e a de Pac-Man. Se a distância entre eles é de oito peças ou mais, Clyde busca Pac-Man diretamente como Blinky faz. Se a distância entre eles é inferior a oito peças, no entanto, Clyde muda seu comportamento para a forma que ele normalmente usa durante o modo de dispersão e vai para seu canto até que ele estar longe o suficiente para começar a busca por Pac-Man de novo. [1]

### 1.3 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*.

### 1.4 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"* [3], 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. Ambos podem ser encontrados em [4] e [5].

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

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

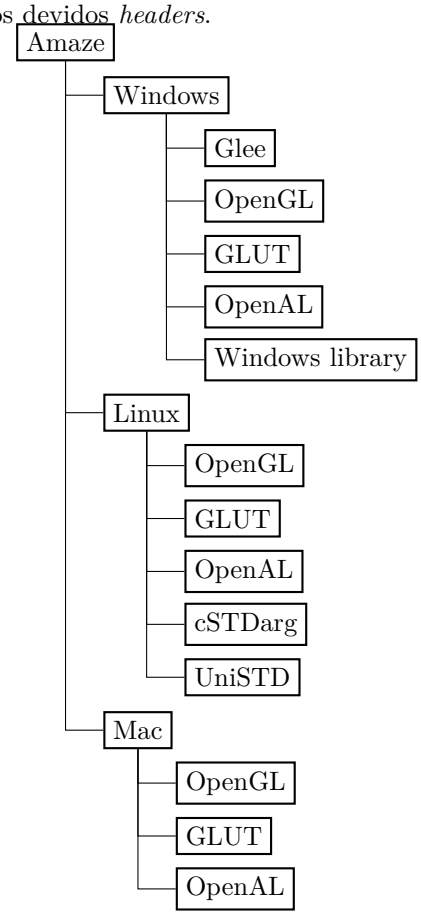
## 2 Desenvolvimento

### 2.1 Estruturas

#### 2.1.1 Arquitetura

No intuito de manter o jogo compatível com qualquer sistema operacional, foi decidido centralizar as inclusões de bibliotecas em um único arquivo. Para essa função

foi criado o arquivo *"defines.h"*, que é responsável por reconhecer o sistema em que esta sendo compilado e incluir os devidos *headers*.




Com base nas dependências de cada sistema operacional, foi montado então o *header* da seguinte forma:

*defines.h*

```
#if defined (__APPLE__) || defined (MACOSX) /*MAC OS*/
    #include <GLUT/glut.h>
    #include <OpenAL/alut.h>
    #include <OpenAL/al.h>
    #include <OpenAL/alc.h>

#else
    #ifdef _WIN32                           /* Windows */
        #define WIN32_LEAN_AND_MEAN
        #include <glee.h>
        #include <gl/gl.h>
        #include <gl/glut.h>
        #include <windows.h>
        #include <AL/al.h>
        #include <AL/alc.h>
        #include <AL/alut.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>
        #include <AL/al.h>
        #include <AL/alc.h>
        #include <AL/alut.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 e funções de espera. Assim, pode-se observar algumas adaptações da função **sleep()** no trecho acima.

Visando o problema de compilação, 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
    SEARCH=dpkg -l | grep -iq
    else
    ifeq ($(UNAME),Darwin) # MAC OS X
        GLFLAGS=-framework OpenGL -framework GLUT \
             -framework OpenAL
        SEARCH=ls /System/Library/Frameworks | grep -i
    else #Windows
        GLFLAGS=-lopengl32 -lglut32 -lglee -lalut
        SEARCH=
    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 Audio - OpenAL

O som do sistema é reproduzido por uma biblioteca chamada **OpenAL**, que permite a criação e reprodução de sons através dos seus buffers e sources. Além disso, ainda foi utilizado o **alut**, que é uma biblioteca auxiliar para o openAL. O alut consta de funções que efetivamente decodificam e carregam o arquivo de audio para os buffers do openAL e a criação das sources utilizadas para a reprodução a partir destes buffers..

O openAL é capaz de reproduzir sons em diversas frequencias e velocidades, com direção em um espaço 3d. No sistema porém é apenas utilizada uma versão super-simplificada disso, ou seja, consta apenas de um som estático. Não importa aonde o jogador se encontre, o som seja sempre ouvido da mesma forma.

Como comentado acima, o alut, assim como o openAL, trabalha emcima de buffers and sources. Primeiramente o arquivo de audio é decodificado e jogado num buffer. Após isso, caso o buffer teja sido criado com sucesso, é atribuido este buffer para uma nova source. As sources podem conter diversos atributos como direção, posição, velocidade e etc para a representação dos sons nos espaços 3d. Porém no caso do sistema, como o som ouvido é sem direção e posição, basta setarmos o seu buffer e quando o source for chamado o som será ouvido.

### 2.1.3 Sistema

A arquitetura do sistema é relativamente simples e consta somente de dois grandes elementos e suas variações. Primeiro temos a classe **Map**. Está classe é responsável por carregar um mapa através de um arquivo .map e coloca-lo em uma lista de **Tiles** que basicamente guardam a posição X, Y e o tipo de bloco que está ocupando. (O sistema possui um optimizador de Tiles que grava nos tiles também quais lados das paredes devem ser impressos caso o tile seja uma parede. Isto ajuda muito no framerate final do sistema.)

A otra importante classe que temos são os elementos que se movem do sistema. Isto é, elementos da classe **Entidade** e suas duas outras derivações **Player** e **Enemy**. A classe Entidade consta de todo e quaisquer elementos do sistema que se movem e possam ou não colidir entre si. Os elementos desta classe também possuem conhecimento do **Map** e a partir deste conhecimento é calculado aonde podem ou não se moverem.

A moedas que o jogador deve coletar não são um objeto do sistema, e sim apesar uma definição do **Tile**. Desta forma, não é necessário calcular uma colisão entre o jogador e as moedas. Caso o jogador esteja próximo ao centro de um **Tile** com uma moeda, será detectado que o jogador pegou a moeda e o **Tile** agora será de um chão vazio.

### 2.1.4 O loop do jogo

O *"loop"* do jogo significa tudo aquilo que é feito durante cada frame mostrado na tela. As teclas pressionadas, os cálculos de novas posições, detecção de colisões, desenho do mapa, desenha das entidades e etc.

1) A iteração começa calculando as novas velocidades e acelerações, assim como as posições para todas as **Entidades**. Isso também já testa as posições do mapa para garantir que está em um espaço que possa se mover.
   a) Caso haja uma colisão com o mapa a entidade simplesmente para de se mover.
2) É chamado o método que testa as colisões entre **Entidade** e **Entidade**.
   a) Caso haja uma colisão entre outra entidade, é gerado uma notificação para tanto a classe que achou a colisão quanto para a classe que sofreu esta colisão. E nada é feito.
3) É chamado o método que executa as colisões em todas as **Entidades**.
   a) Este método irá verificar se foi notificada uma colisão para o objeto.
      i) Caso haja uma colisão é calculado a reação mais apropriada.
      *No exemplo do **Player** colidindo com um **Enemy**: o sistema irá verificar se o jogador está em sua forma especial, caso esteja, o **Enemy** é destruído caso não esteja, o inverso acontece.*
      ii) Após as reações, é então limpado as notificações de colisão.
4) É calculado em qual modo o **Player** se encontra. E modificado a música de acordo.
   a) Ao comer uma moeda especial, a música atual para, é tocado um efeito por 5 segundos e outra música começa a ser tocada até o final do efeito.
5) É executado o Render.
   a) As luzes são setadas nas suas posições iniciais.
   b) O mapa que está até vinte quadrados de distancia do jogador é processado e construído de acordo com os valores dos **Tiles**. Isso inclui as texturas e moedas.
   c) Os inimigos são renderizados.
   d) É renderizado o mini-mapa 2d sobre uma projeção ortho2d.

Esse processo é repetido a cada frame do programa.

### 2.1.5 Execução

2.1.5.1 **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*.

2.1.5.2 **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...
```

2.1.5.3 **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:

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

*Saída do terminal - Mac OS*

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

2.1.5.4 **Valgrind/Callgrind**: No intuito de melhor observar como o programa se comportava durante sua execução, utilizamos da ferramenta do *Valgrind* para visualizar a sequencia de chamadas efetuadas no programa. Para isso foi incorporado no *Makefile* a chamada para o Valgrind, onde uma nova compilação ocorre sem as chamadas de otimização e verificação de erros seguida da chamada do Valgrind para a geração de um arquivo *Callgrind.out*. Este arquivo pode ser utilizado para gerar um gráfico com as chamadas realizadas pelo programa *KCachegrind* semelhante ao gerado na imagem 2. É valido lembrar que o Valgrind roda com memoria limitada. Por este motivo, ele não permite realizar o monitoramento do programa por períodos muito extensos. O gráfico apresentado na figura 2 foi gerado disponibilizando apenas 16MB para captura de dados no Valgrind [6], como pode ser observado no trecho do manual:

> *By default, Valgrind uses the current "ulimit" value for the stack size, or 16 MB, whichever is lower. In many cases this gives a stack size in the range 8 to 16 MB, which almost never overflows for most applications.* [6]

Normalmente, faríamos da seguinte forma para usar o Valgrind:

*Gerando arquivo callgrind.out*

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

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

*Gerando uma biblioteca dinâmica*

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

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

*Compilando com a biblioteca dinâmica*

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

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

*Chamada personalizada do Valgrind*

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

Seguem a lista de alterações passadas para o Valgrind:

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

### *2.1.6 Artefatos*

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

- Sources
  - button.cpp
  - camera.cpp
  - defines.cpp
  - entidade.cpp
  - eventos.cpp
  - framerate.cpp
  - gamemanager.cpp
  - map.cpp
  - minimap.cpp
  - model_obj.cpp
  - player.cpp
  - soundAL.cpp
  - text.cpp
  - textureloader.cpp
  - tile.cpp
- headers
  - button.h
  - camera.h
  - defines.h
  - entidade.h
  - eventos.h
  - framerate.h
  - gamemanager.h
  - map.h
  - minimap.h
  - model_obj.h
  - player.h
  - soundAL.h
  - text.h
  - textureloader.h
  - tile.h

2. Atualizado em 24 de Setembro de 2012

– vetor3d.h
– vetor.h

2.1.6.2 **README**: O arquivo README pode ser localizado dentre os arquivos fontes, em B.2.17. Nele há algumas informações sobre como o programa foi desenvolvido e uma breve instrução de como construir o jogo a partir do código fonte. O README foi implementado com base na linguagem *textile*, devido ao fato de todo o código estar sediado em um servidor SVN que incorpora o arquivo leia-me de um projeto na página principal, oferecendo assim uma recepção visual prazerosa para aqueles que acessam o projeto na rede. Mais sobre *textile* pode ser encontrado em [7].

#### *2.1.7 Problemas Técnicos*

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

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

## 3 Caso de Teste

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

### 3.1 Sistema de derrota

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

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

### 3.2 Sistema de movimento

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

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

### 3.3 Sistema de colisão

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

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

## 4 Conclusão

### 4.1 Aprendizagem

A matéria de Introdução a Computação Gráfica trouxe grande aprendizagem para o grupo quanto a questões de representação de figuras tridimensionais em programas, porém o jogo serviu não apenas para lapidar os conhecimentos ofertados na matéria, mas também para aumentar ainda mais a gama de ferramentas que poderiam ser utilizadas em conjunto. Foi graças ao jogo que o grupo teve contato com OpenAL, uma biblioteca de áudio *open-source* que tornou possível a simplificação e cross-compilação entre os distintos sistemas operacionais. Não o bastante, este projeto serviu também para fixar o uso de ferramentas como o Apache Subversion (também conhecido por SVN), um sistema de controle de versão que permitiu com que

o grupo trabalhasse de forma independente do desenvolvimento dos demais membros, acelerando assim a produção final não apenas do código fonte, mas os relatórios e apresentações. Por ultimo, e não menos importante esta o uso de diversos sistemas operacionais para um mesmo código. Este desafio foi fundamental para policiar a forma de programação que o grupo praticaria, obrigando a todos os membros a respeitar alguns tópicos fundamentais, que é exposto por muitas as empresas de jogos, como a Valve, em seu fórum de desenvolvedores [8], onde recomenda não apenas o uso de ferramentas como o SVN ou Git, mas que seja implementado um controle do código, com estruturas como $\#ifndef$ ou $\#ifdef$ que são parâmetros fundamentais para que em momento de compilação, o compilador possa tomar decisões de quais bibliotecas serão incluídas. Infelizmente, muitas dicas uteis citadas no site da Valve [8] não foram implementadas no jogo, como o uso de multi-threads ou sinais de controle, e ficarão marcadas como sugestões para futuras implementações. Mesmo assim, o grupo entrega o jogo como um produto finalizado, pois estamos satisfeitos com os resultados obtidos ate então, e deixamos as futuras implementações como um convite para que os membros não se distancie deste nível de programação que nos trouce tanto prazer.

### 4.2 Dificuldades encontradas

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

### 4.3 Sugestões

- Multiplayer para 2 jogadores Alternados.
- Registro de nome para usuários que concluírem um nível com sua respectiva pontuação.
- Uso de multi-threads no código no intuito de conseguir melhor desempenho.
- Inclusão de verificação de sinais, com o intuito de que o código possa ter melhor controle de si mesmo, e que erros inesperados não sejam motivo de acumulo de lixo na memoria após uma quebra forçada do programa.

## REFERÊNCIAS


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

[2] D. Hodges. Why do pinky and inky have different behaviors when pac-man is facing up? [Online]. Available: http://donhodges.com/pacman_pinky_explanation.htm

[3] W. D. Pullen. Perfect maze creation algorithms. [Online]. Available: http://www.astrolog.org/labyrnth/algrithm.htm

[4] J. Buck. An implementation of the growing tree algorithm for maze generation. [Online]. Available: https://gist.github.com/760749

[5] Wikipedia and N. Johnston. Maze generation algorithm. [Online]. Available: http://en.wikipedia.org/wiki/Maze_generation_algorithm

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

[7] M. Type. Textile 2 - documentation. [Online]. Available: http://www.movabletype.org/documentation/author/textile-2-syntax.html

[8] Valve. Valve - developer community. [Online]. Available: https://developer.valvesoftware.com/wiki/Category:Programming



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


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

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

# APÊNDICE A
# FIGURAS
## A.1 Valgrind

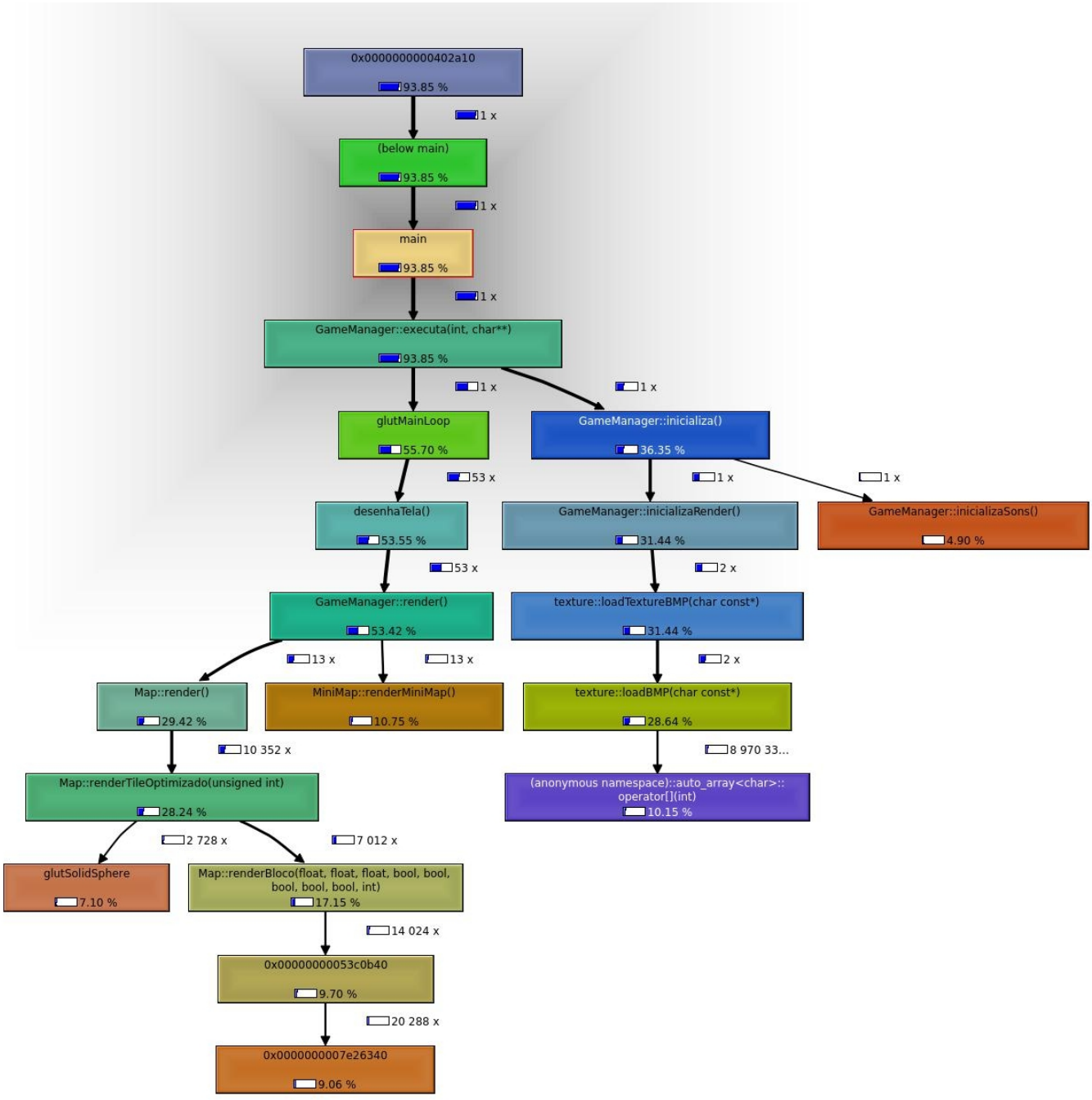



**Figura 2:** Saída gerada pelo Valgrind

## A.2 Diagrama de Classes

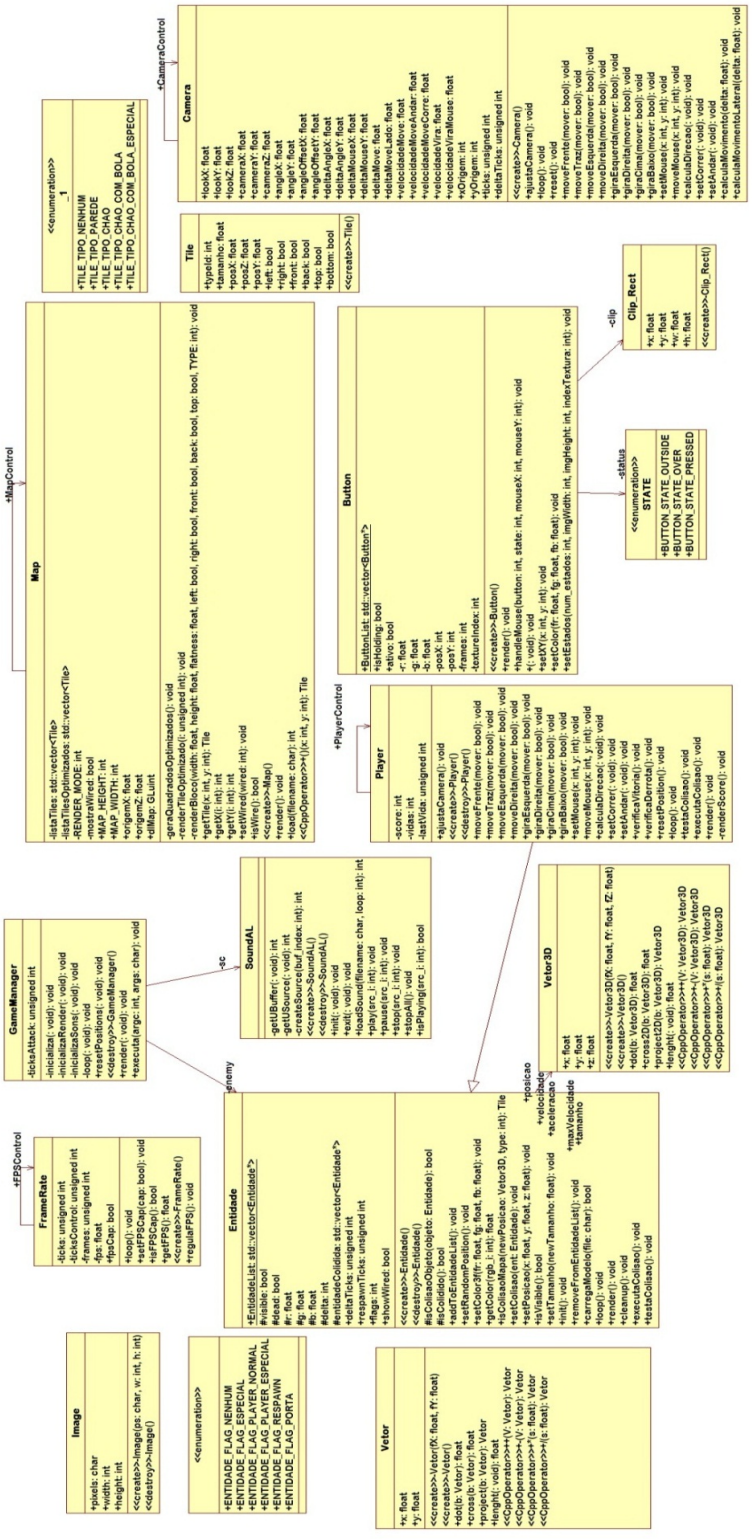


**Figura 3:** Diagrama de classes

# APÊNDICE B
# CÓDIGOS FONTES

## B.1 Headers

### *B.1.1 Button*

```
/*********************************************************************

Button class that inherits from Drawable,

It has a function pointer that points to the action of clicking on the
button.

*********************************************************************/


#ifndef _BUTTON_H__
#define _BUTTON_H__

#include "defines.h"
#include "text.h"
#include <vector>

enum STATE {
    BUTTON_STATE_OUTSIDE = 0,
    BUTTON_STATE_OVER = 1,
    BUTTON_STATE_PRESSED = 2
};

class Clip_Rect
{
public:
    Clip_Rect(){x = y = w = h = 0;}
    float x, y, w, h;
};

class Button
{
    public:

        static std::vector<Button*> ButtonList;

        bool isHolding;

        Button()
        {
            posX = posY = frames = 0;
            ativo = true;
            isHolding = false;
            textureIndex = 0;
            r = g = b = 1.0f;
            ClickAction = 0;
        }
        //Drows
        void render();

        //Loops
        void handleMouse(int button, int state, int mouseX, int mouseY);

        //Action Button
        void (*ClickAction)(void);
        //void (Drawable::*ClickActionBattle)(void);

        //Hold position, COORDINATE SDL
        void setXY(int x, int y);

        void setColor(float fr, float fg, float fb);

        //Sets the number of states and the size of the texture used for the buttons
        void setEstados(int num_estados, int imgWidth, int imgHeight, int indexTextura = 0);

        //if ativo is false, does not run click
        bool ativo;

    private:

        float r,g,b;
        int posX;
        int posY;
        int frames;
        STATE status;
        Clip_Rect clip;

        int textureIndex;

```

```
80};
81
82
83#endif
```

### B.1.2 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(); //Set position and direction of the camera
37        void loop();  //set 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 as public
53        void calculaDirecao(void);
54
55        //Turns run
56        void setCorrer(void);
57        void setAndar(void);
58
59
60        void calculaMovimento(float delta);
61        void calculaMovimentoLateral(float delta);
62
63};
64#endif
```

### B.1.3 Defines

```
 1#ifndef __DEFINESS__H_
 2#define __DEFINESS__H_
 3
 4
 5#if defined (__APPLE__) || defined (MACOSX) /*MAC OS*/
 6    #include <GLUT/glut.h>
 7    #include <OpenAL/alut.h>
 8    #include <OpenAL/al.h>
 9    #include <OpenAL/alc.h>
10
11#else
12    #ifdef _WIN32                          /* Windows */
```

```
13          #define WIN32_LEAN_AND_MEAN
14          #include <glee.h>
15          #include <gl/gl.h>
16          #include <gl/glut.h>
17          #include <windows.h>
18          #include <AL/al.h>
19          #include <AL/alc.h>
20          #include <AL/alut.h>
21
22          #define sleep(x) Sleep(x)
23      #else                                   /*Linux*/
24          #include <cstdarg>
25          #include <unistd.h>
26          #include <GL/gl.h>
27          #include <GL/glut.h>
28          #include <GL/glu.h>
29          #include <AL/al.h>
30          #include <AL/alc.h>
31          #include <AL/alut.h>
32
33          #define Sleep(x) usleep(x<1000000?10000+300*x:x)
34      #endif
35  #endif
36
37  #include <stdio.h>
38  #include <stdlib.h>
39
40
41  #define SCREEN_WIDTH            800
42  #define SCREEN_HEIGHT           600
43
44  #define FRAMES_PER_SECOND       60.0f
45
46  #define TAMANHO_BLOCO           12
47  #define COR_PAREDE              1.0f, 1.0f, 1.0f
48  #define COR_CHAO                1.0f, 1.0f, 1.0f
49  #define COR_COIN                1.0f, 1.0f, 1.0f
50  #define COR_BIG_COIN            0.6f, 0.9f, 0.5f
51  #define GAME_FOV                28
52
53  #define PONTOS_BOLA             10
54  #define PONTOS_BOLA_ESPECIAL    50
55
56  #define TAMANHO_INIMIGO         5
57
58
59
60  //Size of the current screen
61  extern float wScreen;
62  extern float hScreen;
63  //textures
64  extern GLuint wallTexture;
65  extern GLuint floorTexture;
66  //Menu
67  extern bool menuPrincipal;
68  extern int status;
69
70  //Sounds
71  extern int SOUND_main;
72  extern int SOUND_inter1;
73  extern int SOUND_inter2;
74  extern int SOUND_inter3;
75  extern int SOUND_attack;
76  extern int SFX_die;
77  extern int SFX_eat;
78  extern int SFX_eat2;
79  extern int SFX_alert;
80  //Global from gameplay
81  extern int attack_mode;
82
83  #define STATUS_NORMAL 0
84  #define STATUS_VITORIA 1
85  #define STATUS_DERROTA 2
86
87
88
89  #endif
```

### *B.1.4 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#include "soundAL.h"
11#include "model_obj.h"
12
13enum
14{
15    ENTIDADE_FLAG_NENHUM            =   0,
16    ENTIDADE_FLAG_ESPECIAL          =   0x00000001,
17    ENTIDADE_FLAG_PLAYER_NORMAL     =   0x00000002,
18    ENTIDADE_FLAG_PLAYER_ESPECIAL   =   0x00000004,
19    ENTIDADE_FLAG_RESPAWN           =   0x00000008,
20        //not used
21    ENTIDADE_FLAG_PORTA             =   0x00000016
22};
23
24
25class Entidade
26{
27    public:
28        static std::vector<Entidade*> EntidadeList;
29        Entidade();
30        virtual ~Entidade();
31    protected:
32        bool isColisaoObjeto(Entidade* objeto);
33        bool isColidido();
34        bool visible;
35        bool dead;
36
37        float r,g,b;
38
39        int delta;
40        std::vector<Entidade*> entidadeColidida;
41
42
43
44
45
46
47    public:
48        Model_OBJ obj;
49        void createModel(char* filename){obj.Load(filename);}
50        void addToEntidadeList();
51        void setRandomPosition();
52        void setColor3f(float fr, float fg, float fb);
53        float getColor(int rgb_i);
54        Tile* isColisaoMapa(Vetor3D newPosicao, int type = TILE_TIPO_PAREDE);
55        void setColisao(Entidade* ent);
56        void setPosicao(float x, float y, float z);
57        //Ex: int delta = getTicks() - deltaTicks;
58        //Ex: posicao = posicao + (velocidade * (delta/1000.f ) );
59        unsigned int deltaTicks; //quantos ms desde a ultima vez
60        unsigned int respawnTicks;// o tempo em q ele morreu
61        Vetor3D posicao;
62        Vetor3D velocidade;
63        Vetor3D aceleracao;
64        Vetor3D maxVelocidade;
65        Vetor3D tamanho;
66        int flags;
67        bool showWired;
68    public:
69        bool isVisible();
70        void setTamanho(float newTamanho);
71    public:
72        void init();
73        void removeFromEntidadeList();
74
75
76        virtual bool carregaModelo(char* file);
77        virtual void loop();
78        virtual void render();
79        virtual void cleanup();
80        virtual void executaColisao();
81        virtual void testaColisao();
82
83
84};
85
86
87#endif
```

### *B.1.5 Eventos*

```
1#ifndef EVENTOS_H_
2#define EVENTOS_H_
```

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

### B.1.6 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
```

### B.1.7 Game Maneger

```
1//============================================================================
2/*
3    Classe que contera o metodo main e gerenciara o jogo.
4    Class that will have the main method and care the game
5*/
6//============================================================================
7#ifndef _GAME_MANAGER_H_
8#define _GAME_MANAGER_H_
9#include <cstdlib>
10#include "defines.h"
11#include "camera.h"
12#include "framerate.h"
13#include "map.h"
14#include "text.h"
15#include "entidade.h"
16#include "player.h"
17#include "minimap.h"
18#include "button.h"
19#include "soundAL.h"
20#include "textureloader.h"
21
22#define MAX_ENEMY 8
23
24///Note: the cleanup .cpp is called by atExit() in stdlib
25class GameManager
26{
27    private:
28        void inicializa(void);
29        void inicializaRender(void);
30        void inicializaSons(void);
31        void loop(void);
32
33        Entidade* enemy[MAX_ENEMY];
34        Model_OBJ coin;
35
36        //SoundController ... Controls sound
37        SoundAL sc;
```

```
38
39          unsigned int ticksAttack;
40      public:
41          void resetPositions(void);
42          ~GameManager();
43          void render(void);
44          void executa(int argc, char* args[]);
45          void Testes();
46};
47
48
49#endif
```

### B.1.8 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#include "model_obj.h"
12
13
14class Map
15{
16    private:
17        std::vector<Tile> listaTiles;
18        std::vector<Tile> listaTilesOptimizados;
19        void geraQuadradosOptimizados();
20
21        int RENDER_MODE;
22
23
24        //void renderTile(unsigned int i);
25        void renderTileOptimizado(unsigned int i);
26        void renderBloco(float width, float height, float flatness, bool left,
27                         bool right, bool front, bool back, bool top, int TYPE);
28
29
30        bool mostraWired;
31    public:
32        Tile* getTile(int x, int y);
33        inline int getX(int i);
34        inline int getY(int i);
35
36        void setWired(int wired);
37        bool isWire();
38
39        Map();
40
41        //void render();
42        void render();
43        int load(char* filename);
44
45        //Used to others classes to get info about the map
46        static Map MapControl;
47        //Operator overload
48        inline Tile* operator ()  (const int x, const int y)
49        {
50            return this->getTile(x,y);
51        }
52        //Propriedades publicas
53    public:
54        int MAP_HEIGHT;
55        int MAP_WIDTH;
56
57        float origemX; // Where the map start to render
58        float origemZ; //Tile 0,0, growls on right-down
59
60        GLuint dlMap;
61
62        Model_OBJ coin;
63        Model_OBJ bigCoin;
64
65        float coinRotate;
66        float coinVelocidade;
67        //Usa pra calcular rotate
68        unsigned int deltaTicks;
69
70
71};
```

```
72
73
74#endif
```

### B.1.9 Minimap

```
 1#ifndef __MINIMAP_H_
 2#define __MINIMAP_H_
 3
 4#include "map.h"
 5#include "player.h"
 6
 7namespace MiniMap
 8{
 9    extern void renderMiniMap();
10
11}
12
13#endif
```

### B.1.10 Modelo de objetos

```
 1#ifndef MODELS__OBJ_H_
 2#define MODELS__OBJ_H_
 3
 4#include <iostream>
 5#include <fstream>
 6#include <cmath>
 7#include <stdlib.h>
 8#include "defines.h"
 9
10#define POINTS_PER_VERTEX 3
11#define TOTAL_FLOATS_IN_TRIANGLE 9
12
13
14class Model_OBJ
15{
16  public:
17    Model_OBJ();
18    float* calculateNormal(float* coord1,float* coord2,float* coord3 );
19    int Load(char *filename);   // Loads the model
20    void Draw();                    // Draws the model on the screen
21    void Release();             // Release the model
22
23    float* normals;                         // Stores the normals
24    float* Faces_Triangles;                 // Stores the triangles
25    float* vertexBuffer;                    // Stores the points which make the object
26    long TotalConnectedPoints;              // Stores the total number of connected verteces
27    long TotalConnectedTriangles;           // Stores the total number of connected triangles
28
29};
30
31
32
33
34
35#endif
```

### B.1.11 Player

```
 1#ifndef __PLAYER__H_
 2#define __PLAYER__H_
 3
 4#include "entidade.h"
 5#include "text.h"
 6
 7class Player : public Entidade {
 8
 9    public:
10        void ajustaCamera();
11    public:
12
13        Player();
14        ~Player();
15
16        void moveFrente(bool mover);
17        void moveTraz(bool mover);
18        void moveEsquerda(bool mover);
19        void moveDireita(bool mover);
20
21        void giraEsquerda(bool mover);
22        void giraDireita(bool mover);
23        void giraCima(bool mover);
24        void giraBaixo(bool mover);
25
26        void setMouse(int x, int y);
```

```
27          void moveMouse(int x, int y);
28          //temp como public
29          void calculaDirecao(void);
30
31          //Liga ou desliga correr
32          void setCorrer(void);
33          void setAndar(void);
34
35          //Condicoes
36          void verificaVitoria();
37          void verificaDerrota();
38
39          //Virtuais
40          void resetPosition();
41          void loop();
42          void testaColisao();
43          void executaColisao();
44          void render();
45      private:
46
47          int score;
48          void renderScore();
49          int vidas;
50          unsigned int lastVida;
51
52      public:
53          static Player* PlayerControl;
54
55};
56
57
58#endif
```

### B.1.12 Sound

```
1
2/*****************************************************************************
3
4     Versao inicial simplificada. Do post de
5     http://www.gamedev.net/topic/373295-nice-way-to-add-sound-to-my-glut-project-also-icons/
6     Simplesmente carregue o som e toque, nao precisa mexer no listener
7     nem velocidy, position e orientacao
8
9     NOTA: Nao esquecer de chamar init e exit ou havera vazamento de memoria.
10
11*****************************************************************************/
12#ifndef _SOUND_AL_
13#define _SOUND_AL_
14
15#include "defines.h"
16
17#define BUFFER_SIZE_AL 256
18#define SOURCE_SIZE_AL 256
19
20class SoundAL
21{
22      private:
23      typedef struct
24      {
25          ALuint buffer;
26          ALboolean loop;
27          int active;
28      } buf;
29
30      typedef struct
31      {
32          ALuint source;
33          int active;
34      } src;
35
36          //Pega primeiro buffer/source nao utilizado
37          int getUBuffer(void);
38          int getUSource(void);
39          int createSource(int buf_index);
40      public:
41          SoundAL();
42          ~SoundAL();
43
44          static buf buffer[BUFFER_SIZE_AL];
45          static src source[SOURCE_SIZE_AL];
46
47          void init(void);
48          void exit(void);
49
50          int loadSound(const char* filename, int loop);
51
```

```
52          void play(int src_i);
53          void pause(int src_i);
54          void stop(int src_i);
55          void stopAll();
56
57          bool isPlaying(int src_i);
58/*
59          void deleteBuffer(unsigned int b);
60
61          void setListener(float position[3], float vel[3], float orientation[6]);
62          void setSoundPos(unsigned int s, float x, float y, float z, float vx, float vy, float vz);
63*/
64};
65
66#endif
```

### B.1.13 Texto

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

### B.1.14 Carregamento de textura

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

### B.1.15 Tile

```
1//=============================================================================
2/*
```

```
3     Notas:
4     TileID representa
5
6
7*/
8//===========================================================================
9#ifndef __TILE_H_
10#define __TILE_H_
11
12#include "defines.h"
13
14enum
15{
16    TILE_TIPO_NENHUM = 0,
17    TILE_TIPO_PAREDE = 1,
18    TILE_TIPO_CHAO = 2,
19    TILE_TIPO_CHAO_COM_BOLA = 3,
20    TILE_TIPO_CHAO_COM_BOLA_ESPECIAL = 4
21};
22
23class Tile
24{
25    public:
26        int typeId;
27
28        float tamanho;
29
30        float posX, posZ;
31
32        float posY;
33
34        bool left, right, front, back, top, bottom;
35
36        Tile();
37
38};
39
40#endif
```

### *B.1.16 Vetor 3D*

```
1//===========================================================================
2/*
3     A classe Vetor que Coordenada que armazena uma posicao float x e y
4
5     A classe define as operacoes comuns de vetores como produto escalar,
6     vetorial, comprimento, etc.
7
8*/
9//===========================================================================
10#ifndef _VETOR3D_H_
11#define _VETOR3D_H_
12
13#include <math.h>
14
15//
16// NOTA: perproduct = Normal do vetor
17//
18//           /  left hand perproduct,  x = -a.y, y = a.x;
19//          /
20//          ------>
21//         /
22//        /     left hand perproduct,  x = a.y, y = -a.x;
23
24class Vetor3D
25{
26    public:
27
28        float x;
29        float y;
30        float z;
31
32        Vetor3D(float fX, float fY, float fZ) : x(fX), y(fY), z(fZ) {}
33        Vetor3D() {x=0;y=0;z=0;}
34
35        // Produtos
36        // dot = escalar
37        inline float dot(const Vetor3D b) const
38        {
39            return x*b.x + y*b.y + z*b.z;
40        }
41
42        // cross = vetorial
43        inline float cross2D( const Vetor3D b) const
44        {
45            return x*b.y - y*b.x;
```

```
46          }
47
48          // Projecao desse vetor em B
49          //Proj = (U.V) . vetor V
50          //          |V|.|V|
51          Vetor3D project2D(const Vetor3D b)
52          {
53              Vetor3D proj;
54              float dp = dot(b);
55              proj.x = (dp/b.dot(b)) * b.x;
56              proj.y = (dp/b.dot(b)) * b.y;
57
58              return proj;
59          }
60
61          //Tamanho do Vetor
62          inline float lenght(void) const
63          {
64              return (float) sqrt(x*x + y*y + z*z);
65          }
66
67          //Operadores
68
69          inline Vetor3D operator + (const Vetor3D &V)   const
70          { return Vetor3D(x+V.x, y+V.y, z+V.z); }
71
72          inline Vetor3D operator - (const Vetor3D &V)   const
73          { return Vetor3D(x-V.x, y-V.y, z-V.z); }
74
75          inline Vetor3D operator * (float s)           const
76          { return Vetor3D(x*s, y*s, z*s); }
77
78          inline Vetor3D operator / (float s)           const
79          { return Vetor3D(x/s, y/s, z/s); }
80
81
82};
83
84#endif
```

### B.1.17 Vetor

```
1//=========================================================================
2/*
3    A classe Vetor que Coordenada que armazena uma posicao float x e y
4
5    A classe define as operacoes comuns de vetores como produto escalar,
6    vetorial, comprimento, etc.
7
8*/
9//=========================================================================
10#ifndef _VETOR_H_
11#define _VETOR_H_
12
13#include <math.h>
14
15//
16// NOTA: perproduct = Normal do vetor
17//
18//           /  left hand perproduct,  x = -a.y, y = a.x;
19//          /
20//          ------>
21//         /
22//        /     left hand perproduct,  x = a.y, y = -a.x;
23
24class Vetor
25{
26    public:
27
28        float x;
29        float y;
30
31        Vetor(float fX, float fY) : x(fX), y(fY) {}
32        Vetor() {x=0;y=0;}
33
34        // Produtos
35        // dot = escalar
36        inline float dot(const Vetor b) const
37        {
38            return x*b.x + y*b.y;
39        }
40
41        // cross = vetorial
42        inline float cross( const Vetor b) const
43        {
44            return x*b.y - y*b.x;
```

```
45         }
46
47         // Projecao desse vetor em B
48         //Proj = (U.V) . vetor V
49         //         |V|.|V|
50         Vetor project(const Vetor b)
51         {
52             Vetor proj;
53             float dp = dot(b);
54             proj.x = (dp/b.dot(b)) * b.x;
55             proj.y = (dp/b.dot(b)) * b.y;
56
57             return proj;
58         }
59
60         //Tamanho do Vetor
61         inline float lenght(void) const
62         {
63             return (float) sqrt(x*x + y*y);
64         }
65
66         //Operadores
67
68         inline Vetor operator + (const Vetor &V)   const
69         { return Vetor(x+V.x, y+V.y); }
70
71         inline Vetor operator - (const Vetor &V)   const
72         { return Vetor(x-V.x, y-V.y); }
73
74         inline Vetor operator * (float s)           const
75         { return Vetor(x*s, y*s); }
76
77         inline Vetor operator / (float s)           const
78         { return Vetor(x/s, y/s); }
79
80
81};
82
83#endif
```

## B.2 Sources

### *B.2.1 Button*

```
1/*********************************************************************
2
3Classe Button,
4
5It has a function pointer that points to the action of clicking on the
6button.
7
8*********************************************************************/
9
10#include "button.h"
11
12
13std::vector<Button*> Button::ButtonList;
14
15//Draw on the screen
16void Button::render()
17{
18    switch(status)
19    {
20        case BUTTON_STATE_OUTSIDE:
21            clip.x = 0;
22            break;
23        case BUTTON_STATE_OVER:
24            clip.x = clip.w;
25            break;
26        case BUTTON_STATE_PRESSED:
27            clip.x = clip.w*2;
28            break;
29    }
30
31    ///Draw texture on the screen
32    if(textureIndex) // if !=0
33    {
34        glEnable(GL_TEXTURE_2D);
35        glBindTexture(GL_TEXTURE_2D, textureIndex);
36    }
37
38
39
40    txt::setProjecaoOrto();
41    glPushMatrix();
42        glColor3f(r,g,b);
```

```
43          glTranslatef(posX, posY, 0.0f);
44          glBegin(GL_QUADS);
45                  glTexCoord2f(clip.x, clip.y);
46              glVertex2f(0.0f, 0.0f);
47
48                  glTexCoord2f(clip.x, clip.y+clip.h);
49              glVertex2f(0.0f, clip.h);
50
51                  glTexCoord2f(clip.x+clip.h, clip.y+clip.h);
52              glVertex2f(clip.w, clip.h);
53
54                  glTexCoord2f(0.0f, 0.0f);
55              glVertex2f(clip.w, 0.0f);
56          glEnd();
57
58      glPopMatrix();
59      txt::restauraProjecaoPerspectiva();
60
61      glDisable(GL_TEXTURE_2D);
62
63  }
64
65  //Loops
66  void Button::handleMouse(int button, int state, int mouseX, int mouseY)
67  {
68      //release the left button
69      if (button == GLUT_LEFT_BUTTON && state == GLUT_UP)
70          isHolding = false;
71
72      if (mouseX > posX && mouseY > posY
73          && mouseX < posX + clip.w
74          && mouseY < posY + clip.h)
75      {/// Within the button
76          //pressing and holding the left button
77          if(button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
78              isHolding = true;
79
80
81          /// holding
82              if(isHolding && frames > 2) // If the left button
83              {
84                  status = BUTTON_STATE_PRESSED;
85              }
86              else
87              {
88                  if (frames > 1)
89                      status = BUTTON_STATE_OVER; //is not holding or has only two states
90                  else
91                      status =  BUTTON_STATE_OUTSIDE; //has only one state
92              }
93              /// Clicked
94              if (state == GLUT_DOWN)
95              {
96                  if (ativo)
97                  {
98                      if (ClickAction != 0)
99                          (*ClickAction)();
100                 }
101
102             }
103
104     }
105     else
106     {
107         status = BUTTON_STATE_OUTSIDE;
108     }
109
110 }
111
112 //Push button image, and the number of states and texture
113 void Button::setEstados(int num_estados, int imgWidth, int imgHeight, int indexTextura)
114 {
115     if (indexTextura >= 0)
116         textureIndex = indexTextura;
117
118     if ( num_estados < 1 || num_estados > 3)
119     {
120         frames = 1;
121         return;
122     }
123     frames = num_estados;
124
125
126     clip.w = imgWidth/frames;
127     clip.h = imgHeight;
128     clip.y = 0;
```

```
129     clip.x = 0;
130
131
132 }
133
134 // Hold position
135 void Button::setXY(int x, int y)
136 {
137     if ( x >= 0 )
138     {
139         this->posX = x;
140     }
141
142     if ( y >= 0 )
143     {
144         this->posY = y;
145     }
146 }
147
148 void Button::setColor(float fr, float fg, float fb)
149 {
150     r = fr;
151     g = fg;
152     b = fb;
153 }
```

### *B.2.2 Camera*

```
1 #include "camera.h"
2
3 #include <math.h>
4 Camera Camera::CameraControl;
5 Camera::Camera()
6 {
7     angleX = 90.0f;
8     angleY = 0.0f;
9     angleOffsetX = angleOffsetY = 0;
10
11     lookX = 0.5f;
12     lookY = 0.0f;
13     lookZ = -1.0f;
14
15     cameraX = (TAMANHO_BLOCO*1) + TAMANHO_BLOCO/2;
16     cameraY = 5.0f;
17     cameraZ = (TAMANHO_BLOCO*1) + TAMANHO_BLOCO/2;
18     //tests
19
20     //tests
21     deltaAngleX = deltaAngleY = 0.0f; //Angle of rotation of the horizontal and vertical direction
22
23     deltaMouseX = deltaMouseY = 0.0f;
24
25     deltaMove = deltaMoveLado = 0.0f;
26
27
28     velocidadeMoveAndar = CAMERA_ANDA;
29     velocidadeMoveCorre = CAMERA_CORRE;
30     velocidadeMove = velocidadeMoveAndar;
31     velocidadeVira = 45.f;
32     velocidadeViraMouse = 0.1f;
33
34     xOrigem = -1;
35     yOrigem = -1;
36     ticks = 0;
37
38     calculaDirecao();
39 }
40
41 void Camera::reset()
42 {
43     angleX = 90.0f;
44     angleY = 0.0f;
45     angleOffsetX = angleOffsetY = 0;
46
47     lookX = 0.5f;
48     lookY = 0.0f;
49     lookZ = -1.0f;
50
51     cameraX = (TAMANHO_BLOCO*1) + TAMANHO_BLOCO/2;
52     cameraY = 5.0f;
53     cameraZ = (TAMANHO_BLOCO*1) + TAMANHO_BLOCO/2;
54     //tests
55
56     //tests
57     deltaAngleX = deltaAngleY = 0.0f; //Angle of rotation of the horizontal and vertical direction
58
```

```
59      deltaMouseX = deltaMouseY = 0.0f;
60
61      deltaMove = deltaMoveLado = 0.0f;
62
63
64      velocidadeMoveAndar = CAMERA_ANDA;
65      velocidadeMoveCorre = CAMERA_CORRE;
66      velocidadeMove = velocidadeMoveAndar;
67      velocidadeVira = 45.f;
68      velocidadeViraMouse = 0.1f;
69
70      xOrigem = -1;
71      yOrigem = -1;
72      ticks = 0;
73
74      calculaDirecao();
75      ticks = glutGet(GLUT_ELAPSED_TIME);
76  }
77
78
79  //Called internally by Player.
80  void Camera::ajustaCamera()
81  {
82
83      if (deltaAngleX || deltaAngleY)
84          calculaDirecao();
85
86      gluLookAt(  cameraX      , cameraY      , cameraZ,
87                  cameraX+lookX, cameraY+lookY, cameraZ+lookZ,
88                  0.0f   , 1.0f,    0.0f);
89
90      ticks = glutGet(GLUT_ELAPSED_TIME);
91  }
92
93  void Camera::loop()
94  {
95      deltaTicks = glutGet(GLUT_ELAPSED_TIME) - ticks;
96  }
97
98  void Camera::calculaDirecao(void)
99  {
100     float fator = deltaTicks/1000.f;
101     angleX += deltaAngleX*fator;
102     angleY += deltaAngleY*fator;
103
104     //correct angle
105     if ( angleX+angleOffsetX >= 360 )
106         angleX -= 360;
107     if ( angleX+angleOffsetX < 0)
108         angleX += 360;
109
110     //Only allows to rotate 180 degrees Y
111     if ( angleY+angleOffsetY >= 90 )
112         angleY = 90-angleOffsetY;
113     if ( angleY+angleOffsetY <= -90)
114         angleY = -(90+angleOffsetY);
115
116
117     lookX = sin( (angleX+angleOffsetX)*M_PI/180);
118     lookZ = cos( (angleX+angleOffsetX)*M_PI/180);
119
120     lookY = sin( (angleY+angleOffsetY)*M_PI/180);
121 }
122 void Camera::calculaMovimento(float delta)
123 {
124     //Add the movement
125     float fator = deltaTicks/1000.f;
126
127     //Factor delta times direction. 0.1f to adjust speed.
128     cameraX += (delta*fator) * lookX;
129     cameraZ += (delta*fator) * lookZ;
130 }
131 void Camera::calculaMovimentoLateral(float delta)
132 {
133     float fator = deltaTicks/1000.f;
134
135     float lateralX = sin( (angleX-90)*M_PI/180);
136     float lateralZ = cos( (angleX-90)*M_PI/180);
137     //Add the movement
138     //Factor delta times direction. 0.1f to adjust speed.
139     cameraX += (delta*fator) * (lateralX);
140     cameraZ += (delta*fator) * (lateralZ);
141 }
142
143
144 void Camera::moveFrente(bool mover)
```

```
145{
146    if(mover)
147        deltaMove = velocidadeMove;
148    else
149        deltaMove = 0.0f;
150}
151void Camera::moveTraz(bool mover)
152{
153    if(mover)
154        deltaMove = -velocidadeMove;
155    else
156        deltaMove = 0.0f;
157
158}
159void Camera::moveEsquerda(bool mover)
160{
161    if(mover)
162        deltaMoveLado = -velocidadeMove;
163    else
164        deltaMoveLado = 0.0f;
165}
166void Camera::moveDireita(bool mover)
167{
168    if(mover)
169        deltaMoveLado = velocidadeMove;
170    else
171        deltaMoveLado = 0.0f;
172}
173
174void Camera::giraEsquerda(bool mover)
175{
176    if(mover)
177        deltaAngleX = velocidadeVira;
178    else
179        deltaAngleX = 0.0f;
180}
181void Camera::giraDireita(bool mover)
182{
183    if(mover)
184        deltaAngleX = -velocidadeVira;
185    else
186        deltaAngleX = 0.0f;
187}
188void Camera::giraCima(bool mover)
189{
190    if(mover)
191        deltaAngleY = velocidadeVira;
192    else
193        deltaAngleY = 0.0f;
194}
195void Camera::giraBaixo(bool mover)
196{
197    if(mover)
198        deltaAngleY = -velocidadeVira;
199    else
200        deltaAngleY = 0.0f;
201}
202
203void Camera::setMouse(int x, int y)
204{
205    xOrigem = x;
206    yOrigem = y;
207
208    if (xOrigem == -1) //Both will be necessarily -1
209    {
210        angleX +=angleOffsetX;
211        angleY +=angleOffsetY;
212        angleOffsetX = 0;
213        angleOffsetY = 0;
214    }
215}
216void Camera::moveMouse(int x, int y)
217{
218    deltaMouseX = deltaMouseY = 0;
219    //If there was displacement
220    if (xOrigem>0)
221    {
222        angleOffsetX = (xOrigem-x) * 0.1f;
223    }
224    if (yOrigem>0)
225    {
226        angleOffsetY = (yOrigem-y) * 0.1f;
227    }
228    calculaDirecao();
229}
230
```

```
231 void Camera::setCorrer(void)
232 {
233     velocidadeMove = velocidadeMoveCorre;
234 }
235 void Camera::setAndar(void)
236 {
237     velocidadeMove = velocidadeMoveAndar;
238 }
```

### B.2.3 Defines

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

### B.2.4 Entidade

```
 1 #include "entidade.h"
 2
 3 #include <stdlib.h>
 4
 5
 6
 7
 8 //============================================================================
 9 // static variables
10 //============================================================================
11 std::vector<Entidade*> Entidade::EntidadeList;
12
13 //============================================================================
14 // constructors
15 //============================================================================
16 Entidade::Entidade()
17 {
18     flags = ENTIDADE_FLAG_NENHUM;
19     entidadeColidida.clear();
20     deltaTicks = 9999999;
21     deltaTicks = 0;
22     tamanho.x = tamanho.y = tamanho.z = 10;
23     visible = true;
24     dead = false;
25     showWired = false;
26
27     r = 1.0f;
28     g = b = 0.0f;
29
30     maxVelocidade.x = maxVelocidade.y = maxVelocidade.z = 50.f;
31     entidadeColidida.clear();
32
33 }
34
35 void Entidade::init()
36 {
37     deltaTicks = glutGet(GLUT_ELAPSED_TIME);
38 }
39 Entidade::~Entidade()
40 {
41
42 }
43 void Entidade::cleanup()
44 {
45 }
46 bool Entidade::isColisaoObjeto(Entidade* objeto)
47 {
48     //Note: The point marks position 0 .... ex: position 0 beginning of the block end of the block in the x, y, z
49     //Such that y lower = y ; y highest = y+tamanhoY
```

```
50      int baixo1 = this->posicao.y;
51      int cima1 = this->posicao.y + this->tamanho.y;
52      int esquerda1 = this->posicao.x;
53      int direita1 = this->posicao.x + this->tamanho.x;
54      int frente1 = this->posicao.z;
55      int traz1 = this->posicao.z + this->tamanho.z;
56
57      int baixo2 = objeto->posicao.y;
58      int esquerda2 = objeto->posicao.x;
59      int frente2 = objeto->posicao.z;
60      int direita2 = objeto->posicao.x + objeto->tamanho.x;
61      int cima2 = objeto->posicao.y + objeto->tamanho.y;
62      int traz2 = objeto->posicao.z + objeto->tamanho.z;
63
64      if (
65          !(baixo1 > cima2) &&
66          !(cima1  < baixo2) &&
67          !(esquerda1 > direita2) &&
68          !(direita1 < esquerda2) &&
69          !(frente1 > traz2) &&
70          !(traz1 < frente2)
71          )
72          {
73              return true;
74          }
75
76      return false;
77
78  }
79  //=============================================================================
80  // Returns true if colliding with the map
81  //=============================================================================
82  Tile* Entidade::isColisaoMapa(Vetor3D newPosicao, int type)
83  {
84      //Calculates Id tile to be tested
85      //Ex: X = 5 Such that startX = 0,41 = 0 endX = 1,3 = 1
86      int startX = (newPosicao.x) / TAMANHO_BLOCO;
87      int startZ = (newPosicao.z) / TAMANHO_BLOCO;
88      int endX = (newPosicao.x + (tamanho.x)) / TAMANHO_BLOCO;
89      int endZ = (newPosicao.z + (tamanho.z)) / TAMANHO_BLOCO;
90
91      //Check collisions with tiles
92      for(int iZ = startZ; iZ <= endZ; iZ++) {
93          for(int iX = startX; iX <= endX; iX++) {
94              Tile* bloco = Map::MapControl(iX, iZ);
95
96              if(
97                 (bloco->typeId == type) &&
98                 (posicao.y < (bloco->posY+bloco->tamanho) ) &&
99                 ((posicao.y+tamanho.y) > bloco->posY)
100                )
101                 return bloco;
102             }
103     }
104     return 0;
105 }
106
107 void Entidade::removeFromEntidadeList()
108 {
109     for(unsigned int i = 0; i < EntidadeList.size(); i++)
110     {
111         if (EntidadeList[i] == this)
112             EntidadeList.erase(EntidadeList.begin()+i);
113     }
114 }
115 void Entidade::addToEntidadeList()
116 {
117
118
119     for(unsigned int i = 0; i < EntidadeList.size(); i++)
120     {
121         if (EntidadeList[i] == this)
122             return; //Se ja estiver na lista, retorna
123     }
124
125     EntidadeList.push_back(this);
126 }
127
128 bool Entidade::carregaModelo(char* file){return true;}
129 //=============================================================================
130 // Performs actions of the loop, acceleration, speed.
131 //=============================================================================
132 void Entidade::loop()
133 {
134     //3 seconds has the spawn
135     if ( (flags == ENTIDADE_FLAG_RESPAWN) && ( (glutGet(GLUT_ELAPSED_TIME) - respawnTicks) > 3000) )
```

```
136     {
137         dead = false;
138         visible = true;
139         setRandomPosition();
140         flags = ENTIDADE_FLAG_NENHUM;
141     }
142 
143     if(dead) return;
144     //deltaTicks reset the surrender
145     delta = glutGet(GLUT_ELAPSED_TIME) - deltaTicks;
146     float fator = delta/1000.f;
147 
148     //calculates accelerations
149     if ( velocidade.x + aceleracao.x <= maxVelocidade.x)
150         velocidade.x += (aceleracao.x * fator);
151     if ( velocidade.y + aceleracao.y <= maxVelocidade.y)
152         velocidade.y += (aceleracao.y * fator);
153     if ( velocidade.z + aceleracao.z <= maxVelocidade.z)
154         velocidade.z += (aceleracao.z * fator);
155 
156     Vetor3D newPosicao = posicao + (velocidade * fator );
157 
158     if (isColisaoMapa(newPosicao) == false)
159         posicao = newPosicao;
160     else
161     {
162         velocidade.x = 0;
163         velocidade.z = 0;
164         aceleracao.x = 0;
165         aceleracao.z = 0;
166         int pos = (int)(rand() % 4);
167         switch(pos)
168         {
169             case 0:
170                 aceleracao.x = 20;break;
171             case 1:
172                 aceleracao.x = -20;break;
173             case 2:
174                 aceleracao.z = 20;break;
175             case 3:
176                 aceleracao.z = -20;break;
177             default:;
178         }
179 
180     }
181 
182     deltaTicks = glutGet(GLUT_ELAPSED_TIME);
183 }
184 void Entidade::render()
185 {
186     if (!isVisible())
187         return;
188 
189     int tamanhoCubo = tamanho.x; //Temp while using glutCube
190     glPushMatrix();
191     //Centers due to GLUT
192     if (flags == ENTIDADE_FLAG_ESPECIAL)
193         glColor3f( getColor(1), getColor(2), getColor(3) );
194     else
195         glColor3f(r,g,b);
196     glTranslated(posicao.x+tamanho.x/2,
197                  posicao.y+tamanho.y/2,
198                  posicao.z+tamanho.z/2);
199     if (showWired)
200         glutWireCube(tamanhoCubo);
201     else
202     {
203         //glutSolidCube(tamanhoCubo);
204         glScaled(2.0f, 2.4f, 2.0f);
205         obj.Draw();
206 
207     }
208 
209     glPopMatrix();
210 
211 
212 }
213 void Entidade::testaColisao()
214 {
215     if(dead) return;
216 
217     unsigned int thisID = -1;
218     for (unsigned int i = 0; i < EntidadeList.size(); i++)
219         if (EntidadeList[i] == this)
220         {
221             thisID = i;
```

```
222             break;
223         }
224     //Tests with all the entities of this forward.
225     //Ex:    lista: 1 2 3 4
226     // thisID =1,  tests with 2, 3 , 4
227     // thisID = 2  tests with 3, 4      this way, thisID = 2 no collisions with 1 as has already been tested previously.
228     for (unsigned int i = thisID+1; i < EntidadeList.size(); i++)
229     {
230         if (EntidadeList[i] != this && !EntidadeList[i]->dead)
231         {
232             if(isColisaoObjeto(EntidadeList[i]) )
233             {   //adds this element collisions so as tested in
234                 setColisao(EntidadeList[i]);
235                 EntidadeList[i]->setColisao(this);
236             }
237         }
238     }
239 }
240 //Set collision through the public method
241 void Entidade::setColisao(Entidade* ent)
242 {
243     entidadeColidida.push_back(ent);
244 }
245 bool Entidade::isColidido()
246 {
247     if (entidadeColidida.size() == 0)
248         return false;
249     else
250         return true;
251 }
252 void Entidade::executaColisao()
253 {
254     if ( !isColidido() )
255         return; // no collisions
256
257
258 /*
259
260     //Back what had moved.
261     float fator = delta/1000.f;
262     posicao = posicao - (velocidade * fator );
263     //For, and go in the opposite direction
264     velocidade.x = 0;
265     velocidade.z = 0;
266     aceleracao.x = -aceleracao.x;
267     aceleracao.z = -aceleracao.z;
268 */
269     if ( (flags == ENTIDADE_FLAG_ESPECIAL) && (entidadeColidida[0]->flags == ENTIDADE_FLAG_PLAYER_ESPECIAL) )
270     {
271         flags = ENTIDADE_FLAG_RESPAWN;
272         respawnTicks = glutGet(GLUT_ELAPSED_TIME);
273         dead = true;
274         visible = false;
275         SoundAL sc;
276         sc.play(SFX_eat2);
277     }
278
279     entidadeColidida.clear();
280 }
281
282 void Entidade::setRandomPosition()
283 {
284     bool isOK = false;
285         while(!isOK) {
286             int posX = rand() % Map::MapControl.MAP_WIDTH;
287             int posZ = rand() % Map::MapControl.MAP_HEIGHT;
288
289             //If the position is different from the wall, then ground .... put cube
290             if (Map::MapControl.getTile(posX, posZ)->typeId != TILE_TIPO_PAREDE) {
291                 //Note: (TAMANHO_BLOCO/2 - tamanho.x/2)  is used to find the center of the floor
292                 posicao.x = (TAMANHO_BLOCO/2 - tamanho.x/2) + TAMANHO_BLOCO*posX;
293                 posicao.y = 0;
294                 posicao.z = (TAMANHO_BLOCO/2 - tamanho.z/2) + TAMANHO_BLOCO*posZ;
295                 //1 to 10
296                 aceleracao.x = 1 + rand() % 10;
297                 aceleracao.z = 1 + rand() % 10;
298                 init();
299                 isOK = true;
300                 ///Possible to add verification that the entity was not in the same place using
301                 //isColisao and clear() from list of collisions
302             }
303         }
304 }
305
306 bool Entidade::isVisible()
307 {
```

```
308     return visible;
309 }
310 void Entidade::setTamanho(float newTamanho)
311 {
312     tamanho.x = tamanho.y = tamanho.z = newTamanho;
313 }
314 void Entidade::setPosicao(float x, float y, float z)
315 {
316     posicao.x = x;
317     posicao.y = y;
318     posicao.z = z;
319 }
320 void Entidade::setColor3f(float fr, float fg, float fb)
321 {
322     r = fr;
323     g = fg;
324     b = fb;
325 }
326 float Entidade::getColor(int rgb_i)
327 {
328     float color = 0.0f;
329     switch(rgb_i)
330     {
331         case 1:
332             color = r;
333             if (flags == ENTIDADE_FLAG_ESPECIAL)
334                 color -= 0.55f;
335             break;
336         case 2:
337             color = g;
338             if (flags == ENTIDADE_FLAG_ESPECIAL)
339                 color += 1;
340             break;
341         case 3:
342             color = b;
343             if (flags == ENTIDADE_FLAG_ESPECIAL)
344                 color += 0.95f;
345             break;
346     }
347     return color;
348 }
```

### *B.2.5 Eventos*

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

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

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

```
215             Player::PlayerControl->setMouse(x,y);
216         }
217     }
218}
219
220void moveMouse(int x, int y)
221{
222     if (menuPrincipal)
223         return; /// IGNORA ABAIXO
224
225     Player::PlayerControl->moveMouse(x,y);
226}
```

### B.2.6 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}
```

### B.2.7 Game Maneger

```
 1#include "gamemanager.h"
 2#include "eventos.h"
 3#include <time.h>
 4GameManager game;
 5
 6void startButtonAction()
 7{
 8     menuPrincipal = false;
 9
10     game.resetPositions();
11
12     SoundAL sc;
13     sc.stopAll();
14     sc.play(SOUND_inter2);
```

```
15}
16void changeSize(int w, int h)
17{
18    //Prevents division by zero
19    if ( h == 0)
20        h = 1;
21
22    float ratio = w*1.0 / h;
23
24    //Uses projection matrix
25    glMatrixMode(GL_PROJECTION);
26    //Reseta matriz
27    glLoadIdentity();
28
29    //Arranges viewport to entire window
30    glViewport(0,0,w,h);
31
32    //Arranges the right perspective
33    gluPerspective(45.0f, ratio, 1, GAME_FOV*TAMANHO_BLOCO);
34
35    //Back to modelView
36    glMatrixMode(GL_MODELVIEW);
37
38    wScreen = w;
39    hScreen = h;
40}
41void GameManager::inicializaRender(void)
42{
43    //transparency
44    glBlendFunc(GL_SRC_ALPHA,GL_ONE);
45
46    glEnable(GL_LIGHTING); //enables light
47    glEnable(GL_LIGHT0); //enables light #0
48    glEnable(GL_LIGHT1); //enables lightz #0
49    glEnable(GL_NORMALIZE); //Automatically normalize normals
50    glEnable(GL_COLOR_MATERIAL);
51    //glEnable(GL_LIGHT1); //enables light #1
52
53    glEnable(GL_DEPTH_TEST);
54    glShadeModel(GL_SMOOTH); //Shading
55
56    glEnable(GL_CULL_FACE); //Reduces the amount of triangles drawn.
57    glCullFace(GL_CW);
58
59    wallTexture = texture::loadTextureBMP("data/wall.bmp");
60    floorTexture = texture::loadTextureBMP("data/floor.bmp");
61
62
63}
64void GameManager::inicializa(void)
65{
66    inicializaRender();
67    inicializaSons();
68
69//---Testes
70
71    coin.Load((char*)"suzane.obj");
72
73    Map::MapControl.coin.Load((char*)"suzane.obj");
74    Map::MapControl.bigCoin.Load((char*)"suzane.obj");
75//----------------------------
76    //Specifies the background color
77    glClearColor(0.3f,0.3f,0.9f,1.0f);
78
79    GLfloat fog_color[4] = {0.0f,0.0f,0.0f,1.0};
80    glFogfv(GL_FOG_COLOR, fog_color);
81    glFogf(GL_FOG_DENSITY, 0.35f);
82
83    glFogi(GL_FOG_MODE, GL_LINEAR);
84    glHint(GL_FOG_HINT, GL_DONT_CARE);
85    glFogf(GL_FOG_START, TAMANHO_BLOCO*4.0f);
86    glFogf(GL_FOG_END, TAMANHO_BLOCO*10.0f);
87    glEnable(GL_FOG);
88
89    //Tests menu
90    menuPrincipal = true;
91
92    Button* start = new Button();
93
94    start->setXY(220, 200);
95    start->setEstados(1, 350, 60, 0);
96
97    start->ClickAction = startButtonAction;
98
99    Button::ButtonList.push_back(start);
100
```

```
101     for(unsigned int i = 0; i < MAX_ENEMY; i++) {
102         enemy[i] = new Entidade();
103         enemy[i]->addToEntidadeList();
104         enemy[i]->setTamanho(5);
105         enemy[i]->createModel((char*)"ghost.obj");
106     }
107 
108     Player::PlayerControl = new Player();
109     Player::PlayerControl->addToEntidadeList();
110 
111 }
112 
113 void GameManager::inicializaSons(void)
114 {
115     sc.init();
116 
117     SOUND_main = sc.loadSound("data/mus/main.wav", 1);
118     SOUND_inter1 = sc.loadSound("data/mus/M1.WAV", 1);  //Linux & MAC are sensitive case
119     SOUND_inter2 = sc.loadSound("data/mus/M2.WAV", 1);
120     SOUND_inter3 = sc.loadSound("data/mus/M3.WAV", 1);
121     SOUND_attack = sc.loadSound("data/mus/atk.wav", 1);
122 
123     SFX_die = sc.loadSound("data/sfx/die.wav", 0);
124     SFX_eat = sc.loadSound("data/sfx/eat.wav", 0);
125     SFX_eat2 = sc.loadSound("data/sfx/eat2.wav", 0);
126     SFX_alert = sc.loadSound("data/sfx/alert.wav", 0);
127 
128 
129     sc.play(SOUND_inter1);
130 
131 
132 }
133 void GameManager::resetPositions(void)
134 {
135     printf("Posicoes resetadas: %u\n", Entidade::EntidadeList.size());
136 
137     Map::MapControl.load((char*) "map_pacman_new.txt");
138 
139     srand( time(NULL) );
140 
141     for(int i = 0; i < MAX_ENEMY; i++) {
142         enemy[i]->setRandomPosition();
143     }
144 
145     Player::PlayerControl->init();
146     Player::PlayerControl->resetPosition();
147 }
148 
149 void GameManager::Testes()
150 {
151     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
152 
153     glMatrixMode(GL_MODELVIEW);
154     glLoadIdentity();
155 
156     glPushMatrix();
157         glColor3f(1.0f, 1.0f, 1.0f);
158         glTranslatef(2.0f, 0.0f, -10.0f);
159         glRotated(120, 0,1,0);
160         //glutSolidSphere(10.0f, 18.0f, 18.0f);
161         glDisable(GL_CULL_FACE);
162         coin.Draw();
163         glEnable(GL_CULL_FACE);
164     glPopMatrix();
165 }
166 void desenhaTela(void)
167 {
168 
169     game.render();
170 
171     //game.Testes();
172 
173     glutSwapBuffers();
174 }
175 
176 void GameManager::loop(void)
177 {
178 
179     FrameRate::FPSControl.loop();
180     for(unsigned int i = 0; i < Entidade::EntidadeList.size(); i++)
181     {
182         Entidade::EntidadeList[i]->loop();
183     }
184     for(unsigned int i = 0; i < Entidade::EntidadeList.size(); i++)
185     {
186         Entidade::EntidadeList[i]->testaColisao();
```

```
187     }
188     for(unsigned int i = 0; i < Entidade::EntidadeList.size(); i++)
189     {
190         Entidade::EntidadeList[i]->executaColisao();
191     }
192 
193 
194     //Verifies change of states on the special ball
195     if(attack_mode == 1) //notified change and play music
196     {
197         //Ste SPECIAL flag active for all entities. Even the player
198         for(unsigned int i = 0; i < Entidade::EntidadeList.size(); i++)
199         {
200             Entidade::EntidadeList[i]->flags = ENTIDADE_FLAG_ESPECIAL;
201         }
202         Player::PlayerControl->flags = ENTIDADE_FLAG_PLAYER_ESPECIAL; // resets the player's flag
203         ticksAttack = glutGet(GLUT_ELAPSED_TIME);
204         sc.stopAll();
205         sc.play(SFX_alert);
206         attack_mode = 2;
207     } else
208     if (attack_mode == 2)
209     {
210         //after 3 seconds
211         if( (glutGet(GLUT_ELAPSED_TIME) - ticksAttack) > 3000 )
212         {
213             sc.stopAll();
214             sc.play(SOUND_attack);
215             attack_mode = 3;
216             ticksAttack = glutGet(GLUT_ELAPSED_TIME);
217         }
218     } else
219     if (attack_mode == 3)
220     {
221         //over the end of the ball efects 10 seconds + 3 the preceding sfx
222         if( (glutGet(GLUT_ELAPSED_TIME) - ticksAttack) > 10000)
223         {
224             sc.stopAll();
225             sc.play(SOUND_inter2);
226             attack_mode = 0;
227             for(unsigned int i = 0; i < Entidade::EntidadeList.size(); i++)
228             {
229                 Entidade::EntidadeList[i]->flags = ENTIDADE_FLAG_NENHUM;
230             }
231             Player::PlayerControl->flags = ENTIDADE_FLAG_PLAYER_NORMAL; // resets the player's flag
232         }
233     }
234 
235 }
236 void GameManager::render(void)
237 {
238 
239     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
240 
241     glMatrixMode(GL_MODELVIEW);
242     glLoadIdentity();
243 
244     if (menuPrincipal)
245     {
246         for(unsigned int i = 0; i < Button::ButtonList.size();i++)
247             Button::ButtonList[i]->render();
248 
249         txt::renderText2dOrtho(30,150,8,"Aperte o grande quadrado branco para comecar!!!");
250 
251         switch(status)
252         {
253             case STATUS_DERROTA:
254                 txt::renderText2dOrtho(30,130,8,"Derrota!!!");
255                 break;
256             case STATUS_NORMAL:
257                 txt::renderText2dOrtho(30,130,8,"Novo jogo!!!");
258                 break;
259             case STATUS_VITORIA:
260                 txt::renderText2dOrtho(30,130,8,"Vitoria!!!");
261                 break;
262                 default:;
263         }
264 
265         return;
266     }
267 
268 
269     //Lighting
270     GLfloat ambientLight[] = {0.1f, 0.1f, 0.1f, 1.0f};
271     glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientLight);
272     GLfloat directedLight[] = {0.7f, 0.7f, 0.7f, 0.0f};
```

```
273     GLfloat directedLightPos[] = {0.0f, 20.0f, -20.0f, 1.0f};
274     GLfloat light[] = {0.9f, 0.9f, 0.9f, 1.0f};
275     GLfloat lightPos[] = {100.0f, 30.0f, -10.0f, 1.0f};
276     glLightfv(GL_LIGHT0, GL_DIFFUSE, directedLight);
277     glLightfv(GL_LIGHT0, GL_POSITION, directedLightPos);
278
279     glLightfv(GL_LIGHT1, GL_DIFFUSE, light);
280     glLightfv(GL_LIGHT1, GL_POSITION, lightPos);
281     //end of lighting
282
283
284     //calculates iterations
285     this->loop();
286
287     //Print SOL's
288     glPushMatrix();
289         glColor3f(1.0f, 1.0f, 1.0f);
290         glTranslatef(directedLightPos[0],directedLightPos[1],directedLightPos[2]);
291         glutSolidSphere(10.0f, 18.0f, 18.0f);
292     glPopMatrix();
293     glPushMatrix();
294         glColor3f(1.0f, 0.0f, 0.0f);
295         glTranslatef(lightPos[0],lightPos[1],lightPos[2]);
296         glutSolidSphere(10.0f, 18.0f, 18.0f);
297     glPopMatrix();
298
299     Map::MapControl.render();
300     //unsigned int temp = Entidade::EntidadeList.size();
301     for(unsigned int i = 0; i < Entidade::EntidadeList.size(); i++)
302     {
303         if (Entidade::EntidadeList[i]->isVisible())
304             Entidade::EntidadeList[i]->render();
305     }
306
307     txt::renderText2dOrtho(10,15,0,"FPS: %.2f",FrameRate::FPSControl.getFPS());
308
309
310
311
312     MiniMap::renderMiniMap();
313
314 }
315
316
317 // when called during cleanup destructor,
318 // segmentation fault occurs only delete the Entity
319 GameManager::~GameManager()
320 {
321     sc.stopAll();
322     sc.exit();
323     coin.Release();
324     Map::MapControl.coin.Release();
325     Map::MapControl.bigCoin.Release();
326 }
327 void cleanup(void)
328 {
329     unsigned int sizeEnt = Entidade::EntidadeList.size();
330     unsigned int sizeBtn = Button::ButtonList.size();
331     printf("Entidade cleanup size: %u\n", sizeEnt);
332     for(unsigned int i = 0; i < sizeEnt; i++)
333     {
334         delete Entidade::EntidadeList[i];
335     }
336
337     printf("Button cleanup size: %u\n", sizeBtn);
338     for(unsigned int i = 0; i < sizeBtn; i++)
339         delete Button::ButtonList[i];
340     printf("EXIT\n");
341 }
342 void testOpenAL()
343 {
344     unsigned int g_buf = -1;
345     unsigned int g_src = -1;
346
347     if(!alutInit(NULL, NULL))
348     {
349         printf("%s",alutGetErrorString(alutGetError()));
350         return;
351     }
352     alGetError();
353     alutGetError();
354
355     g_buf = alutCreateBufferFromFile("testing.wav");
356
357     if (alutGetError() != ALUT_ERROR_NO_ERROR)
358      {
```

```
359         alDeleteBuffers(1, &g_buf);
360         alutExit();
361         return;
362     }
363
364     alGenSources(1, &g_src);
365
366     if(alGetError() != AL_NO_ERROR)
367     {
368         alDeleteBuffers(1, &g_buf);
369         alDeleteSources(1, &g_buf);
370         alutExit();
371         return;
372     }
373
374     alSourcei(g_src, AL_BUFFER, g_buf);
375
376     alSourcePlay(g_src);
377     alutSleep(4.0f);
378
379     alutExit();
380}
381void testSoundALClass()
382{
383    SoundAL sn;
384    sn.init();
385
386    int m_i = sn.loadSound("testing.wav", 1);
387    sn.play(m_i);
388
389    alutSleep(4.0f);
390
391    sn.exit();
392}
393int main(int argc, char* args[])
394{
395
396    //testOpenAL();
397    //testSoundALClass();
398
399    game.executa(argc, args);
400    return 0;
401}
402void GameManager::executa(int argc, char* args[])
403{
404    glutInit(&argc, args);
405    glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
406    glutInitWindowPosition(100,100);
407    glutInitWindowSize(SCREEN_WIDTH, SCREEN_HEIGHT);
408    glutCreateWindow("Labirinth");
409
410    inicializa();
411
412    glutDisplayFunc(desenhaTela);
413    glutReshapeFunc(changeSize);
414    glutIdleFunc(desenhaTela);
415
416    glutKeyboardFunc(teclasNormais);
417    glutKeyboardUpFunc(teclasNormaisUp);
418    glutSpecialFunc(teclasEspeciais);
419    glutSpecialUpFunc(teclasEspeciaisSoltar);
420    glutMotionFunc(moveMouse);
421    glutMouseFunc(mouseButton);
422
423    atexit(cleanup);
424
425    glutIgnoreKeyRepeat(0);
426    //Get in the loop processing events
427    glutMainLoop();
428}
```

### *B.2.8 Map*

```
1#include "map.h"
2
3//Used by others classes to get info about the map
4Map Map::MapControl;
5
6//Take the Title in position x,y of the map
7//Ex: Map 1 2 3   vector sera 1 2 3 4 5 6
8//        4 5 6
9Tile* Map::getTile(int x, int y)
10{
11    unsigned int ID = 0;
12
13    ID = (y * MAP_WIDTH) + x;
```

```
14
15      return &listaTilesOptimizados[ID];
16  }
17  inline int Map::getX(int i)
18  {
19      return i % MAP_WIDTH;
20  }
21  inline int Map::getY(int i)
22  {
23      return (int) i/MAP_WIDTH;
24  }
25
26  Map::Map()
27  {
28      origemX = -TAMANHO_BLOCO;
29      origemZ = -TAMANHO_BLOCO;
30      mostraWired = false;
31      RENDER_MODE = 0x0007; //GL_QUADS
32      coinRotate = 0;
33      coinVelocidade = 180;
34  }
35
36  void Map::renderBloco(float width, float height, float flatness, bool left,
37          bool right, bool front, bool back, bool top, int TYPE = GL_QUADS)
38  {
39      float w = width/2;
40      float h = height/2;
41      float f = flatness/2;
42
43      float xTexNumber = width/TAMANHO_BLOCO;
44
45      glEnable(GL_TEXTURE_2D);
46      glBindTexture(GL_TEXTURE_2D, wallTexture);
47      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
48      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
49
50
51      glBegin(TYPE);
52      //Front
53      if(front)
54      {
55           glNormal3f(0.0f, 0.0f, 1.0f);
56              //glNormal3f(-1.0f, 0.0f, 1.0f);
57              glTexCoord2f(0.0f, 0.0f);
58          glVertex3f(-w, -h, f);
59              //glNormal3f(1.0f, 0.0f, 1.0f);
60              glTexCoord2f(xTexNumber, 0.0f);
61          glVertex3f(w, -h, f);
62              //glNormal3f(1.0f, 0.0f, 1.0f);
63              glTexCoord2f(xTexNumber, 1.0f);
64          glVertex3f(w, h, f);
65              //glNormal3f(-1.0f, 0.0f, 1.0f);
66              glTexCoord2f(0.0f, 1.0f);
67          glVertex3f(-w, h, f);
68      }
69
70      //Right
71      if(right)
72      {
73           glNormal3f(1.0f, 0.0f, 0.0f);
74              //glNormal3f(1.0f, 0.0f, -1.0f);
75              glTexCoord2f(0.0f, 0.0f);
76          glVertex3f(w, -h, -f);
77              //glNormal3f(1.0f, 0.0f, -1.0f);
78              glTexCoord2f(0.0f, 1.0f);
79          glVertex3f(w, h, -f);
80              glTexCoord2f(1.0f, 1.0f);
81              //glNormal3f(1.0f, 0.0f, 1.0f);
82          glVertex3f(w, h, f);
83              glTexCoord2f(1.f, 0.0f);
84              //glNormal3f(1.0f, 0.0f, 1.0f);
85          glVertex3f(w, -h, f);
86      }
87
88      //Back
89      if(back)
90      {
91              glNormal3f(0.0f, 0.0f, -1.0f);
92              //glNormal3f(-1.0f, 0.0f, -1.0f);
93              glTexCoord2f(0.0f, 0.0f);
94          glVertex3f(-w, -h, -f);
95              //glNormal3f(-1.0f, 0.0f, -1.0f);
96              glTexCoord2f(0.0f, 1.0f);
97          glVertex3f(-w, h, -f);
98              //glNormal3f(1.0f, 0.0f, -1.0f);
99              glTexCoord2f(xTexNumber, 1.0f);
```

```
100         glVertex3f(w, h, -f);
101             //glNormal3f(1.0f, 0.0f, -1.0f);
102             glTexCoord2f(xTexNumber, 0.0f);
103         glVertex3f(w, -h, -f);
104     }
105
106
107     //Left
108     if(left)
109     {
110         glNormal3f(-1.0f, 0.0f, 0.0f);
111             //glNormal3f(-1.0f, 0.0f, -1.0f);
112             glTexCoord2f(0.0f, 0.0f);
113         glVertex3f(-w, -h, -f);
114             //glNormal3f(-1.0f, 0.0f, 1.0f);
115             glTexCoord2f(1.0f, 0.0f);
116         glVertex3f(-w, -h, f);
117             //glNormal3f(-1.0f, 0.0f, 1.0f);
118             glTexCoord2f(1.0f, 1.0f);
119         glVertex3f(-w, h, f);
120             //glNormal3f(-1.0f, 0.0f, -1.0f);
121             glTexCoord2f(0.0f, 1.0f);
122         glVertex3f(-w, h, -f);
123     }
124     glEnd();
125 glDisable(GL_TEXTURE_2D);
126     glBegin(TYPE);
127     //Top
128     if(top)
129     {
130         glNormal3f(0.0f, 1.0f, 0.0f);
131         //glNormal3f(-1.0f, 1.0f, -1.0f);
132         glVertex3f(-w, h, -f);
133             //glNormal3f(-1.0f, 1.0f, 1.0f);
134         glVertex3f(-w, h, f);
135             //glNormal3f(1.0f, 1.0f, 1.0f);
136         glVertex3f(w, h, f);
137             //glNormal3f(1.0f, 1.0f, -1.0f);
138         glVertex3f(w, h, -f);
139     }
140
141     // Don't need background
142     /*
143     //Bottom
144     glNormal3f(0.0f, -1.0f, 0.0f);
145         //glNormal3f(-1.0f, -1.0f, -1.0f);
146     glVertex3f(-w, -h, -f);
147         //glNormal3f(-1.0f, -1.0f, 1.0f);
148     glVertex3f(-w, -h, f);
149         //glNormal3f(1.0f, -1.0f, 1.0f);
150     glVertex3f(w, -h, f);
151         //glNormal3f(1.0f, -1.0f, -1.0f);
152     glVertex3f(w, -h, -f);
153     */
154     glEnd();
155 }
156
157 void Map::render()
158 {
159     glPushMatrix();
160     float offset = (float)TAMANHO_BLOCO/2.0f;
161
162     // Glut start printing starting from the center
163     glTranslated(offset, offset, offset);
164     glColor3f(COR_PAREDE);
165
166     int indexX = (Camera::CameraControl.cameraX / TAMANHO_BLOCO);
167     int indexY = (Camera::CameraControl.cameraZ / TAMANHO_BLOCO);
168
169     int beginX = indexX - GAME_FOV;
170     int beginY = indexY - GAME_FOV;
171     int endX = indexX + GAME_FOV;
172     int endY = indexY + GAME_FOV;
173     if(endX > MAP_WIDTH)
174             endX = MAP_WIDTH;
175     if(endY > MAP_HEIGHT)
176         endY = MAP_HEIGHT;
177     if(beginX < 0)
178         beginX = 0;
179     if(beginY < 0)
180         beginY = 0;
181
182
183     for(int i = beginY; i < endY; i++)
184     {
185         for(int j = beginX; j < endX; j++)
```

```
186         {
187             glPushMatrix();
188                 renderTileOptimizado(j+i*MAP_WIDTH);
189             glPopMatrix();
190         }
191     }
192 
193     //Desenha chao
194     glPopMatrix();
195 
196 }
197 void Map::renderTileOptimizado(unsigned int i)
198 {
199     //Gera fator para a rotacao da moeda
200     int delta = glutGet(GLUT_ELAPSED_TIME) - deltaTicks;
201     float fator = delta/1000.f;
202     deltaTicks = glutGet(GLUT_ELAPSED_TIME);
203 
204     coinRotate += coinVelocidade*fator;
205 
206     if ( coinRotate > 360 )
207         coinRotate-=360;
208 
209 
210     //Camera centra em 0,0,0
211     glTranslated(listaTilesOptimizados[i].posX * TAMANHO_BLOCO,
212                  listaTilesOptimizados[i].posY * TAMANHO_BLOCO,
213                  listaTilesOptimizados[i].posZ * TAMANHO_BLOCO);
214 
215 
216     if(listaTilesOptimizados[i].typeId == TILE_TIPO_PAREDE )
217     {
218         glColor3f(COR_PAREDE);
219         renderBloco(listaTilesOptimizados[i].tamanho, listaTilesOptimizados[i].tamanho, listaTilesOptimizados[i].tamanho,
220                     listaTilesOptimizados[i].left,listaTilesOptimizados[i].right,listaTilesOptimizados[i].front,
221                     listaTilesOptimizados[i].back,listaTilesOptimizados[i].top,
222                     RENDER_MODE);
223 
224     }
225     else //Print ground
226     {
227         glEnable(GL_TEXTURE_2D);
228         glBindTexture(GL_TEXTURE_2D, floorTexture);
229         glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
230         glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
231 
232         float offset = (float)TAMANHO_BLOCO/2.0f;
233         glColor3f(COR_CHAO);
234         glBegin(RENDER_MODE);
235             glNormal3f(0.0f, 1.0f, 0.0f);
236                 glTexCoord2f(0.0f, 0.0f);
237             glVertex3f(-offset, -offset, -offset);
238                 glTexCoord2f(0.0f, 1.0f);
239             glVertex3f(-offset, -offset, offset);
240                 glTexCoord2f(1.0f, 1.0f);
241             glVertex3f(offset, -offset, offset);
242                 glTexCoord2f(1.0f, 0.0f);
243             glVertex3f(offset, -offset, -offset);
244         glEnd();
245 
246         glDisable(GL_TEXTURE_2D);
247         if (listaTilesOptimizados[i].typeId == TILE_TIPO_CHAO_COM_BOLA)
248         {
249             glTranslated(0,-2,0);
250             //glutSolidSphere(1,8,8);
251             glRotatef(coinRotate, 0, 1, 0);
252             glColor3f(COR_COIN);
253             coin.Draw();
254         }
255         else
256         if (listaTilesOptimizados[i].typeId == TILE_TIPO_CHAO_COM_BOLA_ESPECIAL)
257         {
258             glTranslated(0,-2,0);
259             //glutSolidSphere(3,8,8);
260             glRotatef(coinRotate, 0, 1, 0);
261             glColor3f(COR_BIG_COIN);
262             bigCoin.Draw();
263         }
264 
265     }
266 }
267 
268 
269 int Map::load(char* filename)
270 {
271     listaTiles.clear();
```

```
272
273     FILE* file = fopen(filename, "r");
274
275     if(file == NULL)
276         return -1;
277
278     MAP_HEIGHT = MAP_WIDTH = 0;
279
280     // Take the map size (blocks)
281     int error = fscanf(file, "%d-%d\n", &MAP_WIDTH, &MAP_HEIGHT);
282
283     for (int y = 0; y < MAP_HEIGHT; y++)
284     {
285         for (int x = 0; x < MAP_WIDTH; x++)
286         {
287             Tile tempTile;
288             error = fscanf(file, "[%d] ",&tempTile.typeId);
289
290             listaTiles.push_back(tempTile);
291         }
292         error = fscanf(file, "\n");
293     }
294     fclose(file);
295     ///TEST
296     geraQuadradosOptimizados();
297     return error;
298 }
299
300 void Map::geraQuadradosOptimizados()
301 {
302     listaTilesOptimizados.clear();
303
304     for(int iY = 0; iY < MAP_HEIGHT; iY++)
305     {
306        for(int iX = 0; iX < MAP_WIDTH; iX++) //Test all the blocks after this one in X
307        {
308             Tile retangulo;
309             int index = iX + MAP_WIDTH*iY;
310             if (listaTiles[index].typeId != TILE_TIPO_PAREDE)
311             {
312                 retangulo.typeId = listaTiles[index].typeId;
313                 retangulo.posX = iX;
314                 retangulo.posZ = iY;
315                 listaTilesOptimizados.push_back(retangulo);
316                 continue;
317             }
318
319              retangulo.top = true;
320              //If wall, check out of the boards
321              if (index-1 < 0)
322                  retangulo.left = true;
323              else // If ground, than have any wall in this direction
324                  if (listaTiles[index-1].typeId != TILE_TIPO_PAREDE)
325                      retangulo.left = true;
326              if (index - MAP_WIDTH < 0)
327                  retangulo.back = true;
328              else // If ground, than have any wall in this direction
329                  if (listaTiles[index - MAP_WIDTH].typeId != TILE_TIPO_PAREDE)
330                      retangulo.back = true;
331              if (index +1 >= (int)listaTiles.size())
332                  retangulo.right = true;
333              else // If ground, than have any wall in this direction
334                  if (listaTiles[index +1].typeId != TILE_TIPO_PAREDE)
335                      retangulo.right = true;
336              if (index + MAP_WIDTH >= (int)listaTiles.size())
337                  retangulo.front = true;
338              else // If ground, than have any wall in this direction
339                  if (listaTiles[index + MAP_WIDTH].typeId != TILE_TIPO_PAREDE)
340                      retangulo.front = true;
341
342              retangulo.posX = iX;
343              retangulo.posZ = iY;
344              retangulo.typeId = listaTiles[index].typeId;
345
346              listaTilesOptimizados.push_back(retangulo);
347
348         }
349     }
350 }
351
352
353
354 void Map::setWired(int wired)
355 {
356     if (wired)
357     {
```

```
358         mostraWired = true;
359         RENDER_MODE = GL_LINES;
360     }
361     else
362     {
363         mostraWired = false;
364         RENDER_MODE = GL_QUADS;
365     }
366
367}
368bool Map::isWire()
369{
370     return mostraWired;
371}
```

### B.2.9 Minimap

```
 1#include "minimap.h"
 2
 3
 4namespace MiniMap
 5{
 6     void renderMiniMap()
 7     {
 8     glEnable(GL_BLEND);
 9     txt::setProjecaoOrto();
10
11     int tileMiniSize = 10;
12     int posXInit = wScreen-300;
13     int posYInit = 20;
14     for(int y = 0; y < Map::MapControl.MAP_HEIGHT;y++) {
15         for(int x = 0; x < Map::MapControl.MAP_WIDTH;x++) {
16             glPushMatrix();
17
18                 Tile* bloco = Map::MapControl.getTile(x,y);
19
20                 if (bloco->typeId == TILE_TIPO_PAREDE)
21                     glColor4f(1.0f,1.0f,1.0f,0.5f);
22                 else
23                     glColor4f(0.0f,0.0f,5.0f,0.5f);
24
25
26                     glTranslatef(posXInit + x*tileMiniSize, posYInit + y*tileMiniSize, 0.0f);
27                 glBegin(GL_QUADS);
28                     glVertex2f(0.0f, 0.0f);
29                     glVertex2f(0.0f, tileMiniSize);
30                     glVertex2f(tileMiniSize, tileMiniSize);
31                     glVertex2f(tileMiniSize, 0.0f);
32                 glEnd();
33
34                 if (bloco->typeId == TILE_TIPO_CHAO_COM_BOLA) {
35                     glTranslatef(tileMiniSize/2, tileMiniSize/2, 0.0f);
36                     glColor4f(0.0f,1.0f, 0.0f, 0.5f);
37                     float raio = 2;
38                     glBegin(GL_TRIANGLE_FAN);
39                         glVertex2f(0, 0);
40                         for (int angle = 0; angle < 360; angle+=5)
41                             glVertex2f(sin(M_PI*angle/180) * raio, cos(M_PI*angle/180) * raio);
42                     glEnd();
43                 }
44                 if (bloco->typeId == TILE_TIPO_CHAO_COM_BOLA_ESPECIAL) {
45                     glTranslatef(tileMiniSize/2, tileMiniSize/2, 0.0f);
46                     glColor4f(0.0f,1.0f, 5.0f, 0.5f);
47                     float raio = 3.5;
48                     glBegin(GL_TRIANGLE_FAN);
49                         glVertex2f(0, 0);
50                         for (int angle = 0; angle < 360; angle+=35)
51                             glVertex2f(sin(M_PI*angle/180) * raio, cos(M_PI*angle/180) * raio);
52                     glEnd();
53                 }
54
55
56             glPopMatrix();
57         }
58     }
59     ///Desenha entidades do mapa, inimigos, jogador.
60     //Fator para adequar as posicoes do real para o mini-mapa
61     float fator = float(tileMiniSize)/float(TAMANHO_BLOCO);
62     float tamanhoEntidadeX = float(TAMANHO_INIMIGO)*fator;
63     float tamanhoEntidadeY = float(TAMANHO_INIMIGO)*fator;
64     //Desenha inimigos e jogador
65     for(unsigned int i = 0; i < Entidade::EntidadeList.size(); i++) {
66
67         //se entidade nao visivel... ignora
68         if (Entidade::EntidadeList[i]->isVisible() == false)
69             continue;
```

```
70
71          tamanhoEntidadeX = Entidade::EntidadeList[i]->tamanho.x * fator;
72          tamanhoEntidadeY = Entidade::EntidadeList[i]->tamanho.z * fator;
73
74          glPushMatrix();
75              glColor4f(Entidade::EntidadeList[i]->getColor(1),
76                        Entidade::EntidadeList[i]->getColor(2),
77                        Entidade::EntidadeList[i]->getColor(3), 1.0f);
78                          //Inicio do mapa, posicao fatorada no mapa da entidade
79              glTranslatef(posXInit + Entidade::EntidadeList[i]->posicao.x*fator,
80                           posYInit + Entidade::EntidadeList[i]->posicao.z*fator, 0.0f);
81
82              //Se for o jogador
83              if(Entidade::EntidadeList[i] == Player::PlayerControl){
84                  glColor4f(1.0f,1.0f,1.0f,1.0f);
85                  glBegin(GL_QUADS);
86                      glVertex2f(-1.0f, -1.0f);
87                      glVertex2f(-1.0f, tamanhoEntidadeY+1);
88                      glVertex2f(tamanhoEntidadeX+1, tamanhoEntidadeY+1);
89                      glVertex2f(tamanhoEntidadeX+1, -1.0f);
90                  glEnd();
91
92              }
93              else //Se inimigos
94              {
95                  glBegin(GL_QUADS);
96                      glVertex2f(0.0f, 0.0f);
97                      glVertex2f(0.0f, tamanhoEntidadeY);
98                      glVertex2f(tamanhoEntidadeX, tamanhoEntidadeY);
99                      glVertex2f(tamanhoEntidadeX, 0.0f);
100                 glEnd();
101             }
102
103         glPopMatrix();
104     }
105
106
107     txt::restauraProjecaoPerspectiva();
108     glDisable(GL_BLEND);
109     }
110
111 }
```

### *B.2.10 Modelo de objetos*

```
1 #include "model_obj.h"
2
3 using namespace std;
4
5 Model_OBJ::Model_OBJ()
6 {
7     this->TotalConnectedTriangles = 0;
8     this->TotalConnectedPoints = 0;
9 }
10
11 float* Model_OBJ::calculateNormal( float *coord1, float *coord2, float *coord3 )
12 {
13    /* calculate Vector1 and Vector2 */
14    float va[3], vb[3], vr[3], val;
15    va[0] = coord1[0] - coord2[0];
16    va[1] = coord1[1] - coord2[1];
17    va[2] = coord1[2] - coord2[2];
18
19    vb[0] = coord1[0] - coord3[0];
20    vb[1] = coord1[1] - coord3[1];
21    vb[2] = coord1[2] - coord3[2];
22
23    /* cross product */
24    vr[0] = va[1] * vb[2] - vb[1] * va[2];
25    vr[1] = vb[0] * va[2] - va[0] * vb[2];
26    vr[2] = va[0] * vb[1] - vb[0] * va[1];
27
28    /* normalization factor */
29    val = sqrt( vr[0]*vr[0] + vr[1]*vr[1] + vr[2]*vr[2] );
30
31     float norm[3];
32     norm[0] = vr[0]/val;
33     norm[1] = vr[1]/val;
34     norm[2] = vr[2]/val;
35
36     float* pointer = norm;
37     return pointer;
38 }
39
40
41 int Model_OBJ::Load(char* filename)
```

```
42{
43    string line;
44    ifstream objFile (filename);
45    if (objFile.is_open())                                          // If obj file is open, continue
46    {
47        objFile.seekg (0, ios::end);                                // Go to end of the file,
48        long fileSize = objFile.tellg();                            // get file size
49        objFile.seekg (0, ios::beg);                                // we'll use this to register memory for our 3d mod
50
51        vertexBuffer = (float*) malloc (fileSize);                  // Allocate memory for the verteces
52        Faces_Triangles = (float*) malloc(fileSize*sizeof(float));  // Allocate memory for the triangles
53        normals  = (float*) malloc(fileSize*sizeof(float));         // Allocate memory for the normals
54
55        int triangle_index = 0;                                     // Set triangle index to zero
56        int normal_index = 0;                                       // Set normal index to zero
57
58        while (! objFile.eof() )                                    // Start reading file data
59        {
60            getline (objFile,line);                                 // Get line from file
61
62            if (line.c_str()[0] == 'v')                             // The first character is a v: on this line is a ve
63            {
64                line[0] = ' ';                                      // Set first character to 0. This will allow us to
65
66                sscanf(line.c_str(),"%f %f %f ",                    // Read floats from the line: v X Y Z
67                    &vertexBuffer[TotalConnectedPoints],
68                    &vertexBuffer[TotalConnectedPoints+1],
69                    &vertexBuffer[TotalConnectedPoints+2]);
70
71                TotalConnectedPoints += POINTS_PER_VERTEX;          // Add 3 to the total connected points
72            }
73            if (line.c_str()[0] == 'f')                             // The first character is an 'f': on this line is a
74            {
75                line[0] = ' ';                                      // Set first character to 0. This will allow us to
76
77                int vertexNumber[4] = { 0, 0, 0 };
78                sscanf(line.c_str(),"%i%i%i",                       // Read integers from the line:
  f 1 2 3
79                    &vertexNumber[0],                               // First point of our triangle. This is an
80                    &vertexNumber[1],                               // pointer to our vertexBuffer list
81                    &vertexNumber[2] );                             // each point represents an X,Y,Z.
82
83                vertexNumber[0] -= 1;                               // OBJ file starts counting from 1
84                vertexNumber[1] -= 1;                               // OBJ file starts counting from 1
85                vertexNumber[2] -= 1;                               // OBJ file starts counting from 1
86
87
88                /*********************************************************************
89                 * Create triangles (f 1 2 3) from points: (v X Y Z) (v X Y Z) (v X Y Z).
90                 * The vertexBuffer contains all verteces
91                 * The triangles will be created using the verteces we read previously
92                 */
93
94                int tCounter = 0;
95                for (int i = 0; i < POINTS_PER_VERTEX; i++)
96                {
97                    Faces_Triangles[triangle_index + tCounter   ] = vertexBuffer[3*vertexNumber[i] ];
98                    Faces_Triangles[triangle_index + tCounter +1 ] = vertexBuffer[3*vertexNumber[i]+1 ];
99                    Faces_Triangles[triangle_index + tCounter +2 ] = vertexBuffer[3*vertexNumber[i]+2 ];
100                   tCounter += POINTS_PER_VERTEX;
101               }
102
103               /*********************************************************************
104                * Calculate all normals, used for lighting
105                */
106               float coord1[3] = { Faces_Triangles[triangle_index], Faces_Triangles[triangle_index+1],Faces_Triangles[triangle
107               float coord2[3] = {Faces_Triangles[triangle_index+3],Faces_Triangles[triangle_index+4],Faces_Triangles[triangle
108               float coord3[3] = {Faces_Triangles[triangle_index+6],Faces_Triangles[triangle_index+7],Faces_Triangles[triangle
109               float *norm = this->calculateNormal( coord1, coord2, coord3 );
110
111               tCounter = 0;
112               for (int i = 0; i < POINTS_PER_VERTEX; i++)
113               {
114                   normals[normal_index + tCounter ] = norm[0];
115                   normals[normal_index + tCounter +1] = norm[1];
116                   normals[normal_index + tCounter +2] = norm[2];
117                   tCounter += POINTS_PER_VERTEX;
118//                 printf("%f\t",normals[normal_index + tCounter ]);
119               }
120
121               triangle_index += TOTAL_FLOATS_IN_TRIANGLE;
122               normal_index += TOTAL_FLOATS_IN_TRIANGLE;
123               TotalConnectedTriangles += TOTAL_FLOATS_IN_TRIANGLE;
124           }
125       }
126       objFile.close();                                                // Close OBJ file
```

```
127     }
128     else
129     {
130         cout << "Unable to open file";
131     }
132     return 0;
133 }
134 
135 void Model_OBJ::Release()
136 {
137     //Se objeto nao criado
138     if(this->TotalConnectedTriangles == 0)
139         return;
140     free(this->Faces_Triangles);
141     free(this->normals);
142     free(this->vertexBuffer);
143 }
144 
145 void Model_OBJ::Draw()
146 {
147     glDisable(GL_CULL_FACE);                        //Custom, estava cortando a face de traz quando objeto virado
148     glEnableClientState(GL_VERTEX_ARRAY);                       // Enable vertex arrays
149     glEnableClientState(GL_NORMAL_ARRAY);                       // Enable normal arrays
150     glVertexPointer(3,GL_FLOAT, 0,Faces_Triangles);             // Vertex Pointer to triangle array
151     glNormalPointer(GL_FLOAT, 0, normals);                      // Normal pointer to normal array
152     glDrawArrays(GL_TRIANGLES, 0, TotalConnectedTriangles);     // Draw the triangles
153     glDisableClientState(GL_VERTEX_ARRAY);                      // Disable vertex arrays
154     glDisableClientState(GL_NORMAL_ARRAY);                      // Disable normal arrays
155     glEnable(GL_CULL_FACE);
156 }
```

### B.2.11 Player

```
 1 #include "player.h"
 2 
 3 Player* Player::PlayerControl;
 4 
 5 //Chame init() para inicializar e coloca-lo na lista
 6 Player::Player()
 7 {
 8     Entidade();
 9     resetPosition();
10     flags = ENTIDADE_FLAG_PLAYER_NORMAL;
11 }
12 void Player::resetPosition()
13 {
14     Camera::CameraControl.reset();
15     lastVida = 0;
16 
17     setTamanho(5);
18     posicao.x = ((TAMANHO_BLOCO*1) + TAMANHO_BLOCO/2) - (tamanho.x/2);
19     posicao.y = tamanho.y/2;
20     posicao.z = ((TAMANHO_BLOCO*1) + TAMANHO_BLOCO/2) - (tamanho.z/2);
21     showWired = true;
22     score = 0;
23     vidas = 1;
24 }
25 
26 Player::~Player()
27 {
28 }
29 void Player::ajustaCamera()
30 {
31     Camera::CameraControl.loop(); //Ajusta timer.
32 
33     float deltaMove = Camera::CameraControl.deltaMove;
34     //Se estiver movendo (frente ou traz)
35     if (deltaMove) //Se estiver colidido, nem calcula o movimento
36     {   //Efetua movimento
37         Camera::CameraControl.calculaMovimento(deltaMove);                      //Calcula posicao da camera
38         Vetor3D pos;
39         // menos tamanho.x para pegar o centro do cubo.
40         pos.x = Camera::CameraControl.cameraX-(tamanho.x/2);
41         pos.y = Camera::CameraControl.cameraY-(tamanho.y/2);
42         pos.z = Camera::CameraControl.cameraZ-(tamanho.z/2);
43         if( isColisaoMapa(pos) == false)  //Verifica se colidiu
44             setPosicao(pos.x, pos.y, pos.z); // e setado para poder calcular colisoes com entidades no futuro
45         else
46             //Se colidir, entao desfaz o movimento
47             Camera::CameraControl.calculaMovimento(-deltaMove);                 //Recalcula para posicao anterior se colidiu
48     }
49     float deltaMoveLado = Camera::CameraControl.deltaMoveLado;
50     //Se moviento lateral
51     if (deltaMoveLado)
52     {
53         Camera::CameraControl.calculaMovimentoLateral(deltaMoveLado);
```

```
54          Vetor3D pos;
55          pos.x = Camera::CameraControl.cameraX-(tamanho.x/2);
56          pos.y = Camera::CameraControl.cameraY-(tamanho.y/2);
57          pos.z = Camera::CameraControl.cameraZ-(tamanho.z/2);
58          if (isColisaoMapa(pos) == false)
59              setPosicao(pos.x, pos.y, pos.z);
60          else
61              Camera::CameraControl.calculaMovimentoLateral(-deltaMoveLado);
62      }
63
64      Camera::CameraControl.ajustaCamera();
65
66      render();
67  }
68  void Player::moveFrente(bool mover){
69      Camera::CameraControl.moveFrente(mover);
70  }
71  void Player::moveTraz(bool mover){
72      Camera::CameraControl.moveTraz(mover);
73  }
74  void Player::moveEsquerda(bool mover){
75      Camera::CameraControl.moveEsquerda(mover);
76  }
77  void Player::moveDireita(bool mover){
78      Camera::CameraControl.moveDireita(mover);
79  }
80  void Player::giraEsquerda(bool mover){
81      Camera::CameraControl.giraEsquerda(mover);
82  }
83  void Player::giraDireita(bool mover){
84      Camera::CameraControl.giraDireita(mover);
85  }
86  void Player::giraCima(bool mover){
87      Camera::CameraControl.giraCima(mover);
88  }
89  void Player::giraBaixo(bool mover){
90      Camera::CameraControl.giraBaixo(mover);
91  }
92  void Player::setMouse(int x, int y){
93      Camera::CameraControl.setMouse(x,y);
94  }
95  void Player::moveMouse(int x, int y){
96      Camera::CameraControl.moveMouse(x,y);
97  }
98  //temp como public
99  void Player::calculaDirecao(void){
100     Camera::CameraControl.calculaDirecao();
101 }
102 //Liga ou desliga correr
103 void Player::setCorrer(void){
104     Camera::CameraControl.setCorrer();
105 }
106 void Player::setAndar(void){
107     Camera::CameraControl.setAndar();
108     deltaTicks = glutGet(GLUT_ELAPSED_TIME);
109 }
110
111 //
112 void Player::loop()
113 {
114     //int delta = glutGet(GLUT_ELAPSED_TIME) - deltaTicks;
115     //float fator = delta/1000.f;
116     //deltaTicks = glutGet(GLUT_ELAPSED_TIME);
117     //testaColisao();
118     ajustaCamera();
119     verificaVitoria();
120
121 }
122 void Player::testaColisao()
123 {
124     Tile* tileBall = 0;
125
126     //Se for diferente de 0, isto e, NULL.
127     if ( (tileBall = isColisaoMapa(posicao, TILE_TIPO_CHAO_COM_BOLA)) ) {
128         //BOLA NORMAL, ACUMULA PONTOS
129         score += PONTOS_BOLA;
130         tileBall->typeId = TILE_TIPO_CHAO;
131         ///TOCA SOM!!!!
132         SoundAL sc;
133         sc.play(SFX_eat);
134
135     } else if ((tileBall = isColisaoMapa(posicao, TILE_TIPO_CHAO_COM_BOLA_ESPECIAL)) ){
136         score += PONTOS_BOLA_ESPECIAL;
137         tileBall->typeId = TILE_TIPO_CHAO;
138         ///TOCA SOM!!!!
139         SoundAL sc;
```

```
140         sc.play(SFX_eat2);
141         //ativa especial
142         attack_mode = 1;
143     }
144 
145     //Executa
146 
147 }
148 
149 void Player::renderScore()
150 {
151    txt::renderText2dOrtho(wScreen -100, 10,0,"Pontos:%d",score);
152    txt::renderText2dOrtho(wScreen -160, 10,0,"Vidas:%d",vidas);
153 }
154 void Player::render()
155 {
156     Entidade::render();
157     renderScore();
158 }
159 void Player::executaColisao()
160 {
161     if (!isColidido())
162         return;
163 
164     //so reajusta posicao e verifica derrota se nao estiver com o efeito do especial
165     if (entidadeColidida[0]->flags == ENTIDADE_FLAG_NENHUM)
166     {
167         Camera::CameraControl.calculaMovimento(-Camera::CameraControl.deltaMove);
168         Camera::CameraControl.calculaMovimentoLateral(-Camera::CameraControl.deltaMoveLado);
169         setPosicao(Camera::CameraControl.cameraX-(tamanho.x/2), Camera::CameraControl.cameraY-(tamanho.x/2), Camera::CameraCont
170 
171         verificaDerrota();
172     }
173 
174 
175 
176 
177 
178     entidadeColidida.clear();
179 }
180 
181 void Player::verificaVitoria(){
182     int bolas = 0;
183     for(int iy = 0; iy < Map::MapControl.MAP_HEIGHT;iy++)
184         for(int ix = 0; ix < Map::MapControl.MAP_WIDTH;ix++) {
185             if (Map::MapControl.getTile(ix,iy)->typeId == TILE_TIPO_CHAO_COM_BOLA)
186                 bolas++;
187         }
188 
189     if (bolas <= 0) {
190         menuPrincipal = true;
191         status = STATUS_VITORIA;
192         SoundAL sc;
193         sc.stopAll();
194         sc.play(SOUND_inter3);
195         alutSleep(4.0f);
196         sc.stopAll();
197 
198         //Reseta especial
199         attack_mode = 0;
200         for(unsigned int i = 0; i < Entidade::EntidadeList.size(); i++)
201         {
202             Entidade::EntidadeList[i]->flags = ENTIDADE_FLAG_NENHUM;
203         }
204         Player::PlayerControl->flags = ENTIDADE_FLAG_PLAYER_NORMAL; // reseta a flag player
205     }
206 }
207 void Player::verificaDerrota(){
208     //se tiver passado 1.5 segundos desde a ultima vez que tomou dano.
209     if ( (glutGet(GLUT_ELAPSED_TIME) - lastVida ) > 1500 ) {
210         vidas--;
211         lastVida = glutGet(GLUT_ELAPSED_TIME);
212     }
213 
214 
215     if (vidas <= 0) {
216         menuPrincipal = true;
217         status = STATUS_DERROTA;
218         SoundAL sc;
219         sc.stopAll();
220         sc.play(SFX_die);
221         alutSleep(2.0f);
222         sc.play(SOUND_inter1);
223 
224         //Reseta especial
225         attack_mode = 0;
```

```
226         for(unsigned int i = 0; i < Entidade::EntidadeList.size(); i++)
227         {
228             Entidade::EntidadeList[i]->flags = ENTIDADE_FLAG_NENHUM;
229         }
230         Player::PlayerControl->flags = ENTIDADE_FLAG_PLAYER_NORMAL; // reseta a flag player
231     }
232}
```

### B.2.12 Sound

```
 1#include "soundAL.h"
 2
 3SoundAL::buf SoundAL::buffer[BUFFER_SIZE_AL];
 4SoundAL::src SoundAL::source[SOURCE_SIZE_AL];
 5
 6SoundAL::SoundAL()
 7{
 8}
 9SoundAL::~SoundAL()
10{
11}
12
13void SoundAL::init()
14{
15     alutInit(NULL, 0);
16
17
18     for(int i = 0; i < BUFFER_SIZE_AL;i++)
19         buffer[i].active=0;
20
21         for(int i = 0; i < SOURCE_SIZE_AL;i++)
22         source[i].active=0;
23     //limpa erros
24     alGetError();
25     alutGetError();
26}
27
28int SoundAL::loadSound(const char* filename, int loop = 0)
29{
30
31     //AO RODAR COM alutCreateBufferFromFile, gera erro de renderizacao quando executar. Porem ao usar a versao depreciada, func
32     ALuint buf_value = 0;//alutCreateBufferFromFile(filename);
33
34     /** VERSAO DEPRECIADA... FUNCIONA **/
35     ALenum format;
36     ALbyte * data;
37     ALsizei size, freq;
38     ALuint name;
39
40     alutLoadWAVFile((ALbyte*)filename, &format, (ALvoid **)&data, &size, &freq, NULL);
41     #ifdef DEBUG
42     printf("DEBUG{%s >> format: %d; size: %d; freq: %d}\n",
43            filename, format, size, freq);
44         #endif
45
46     alGenBuffers(1, &name);
47     alBufferData(name, format, data, size, freq);
48
49     buf_value = name;
50     /** FIM DA VERSAO **/
51     //if couldn't load the file
52     if(!buf_value)
53     {
54         printf("Erro ao carregar arquivo de som(%s): %s\n",filename, alutGetErrorString(alutGetError()));
55         return -1;
56     }
57
58     int buf_index = getUBuffer();
59
60     if (buf_index == -1)
61         return -1;
62
63     buffer[buf_index].buffer = buf_value;
64     buffer[buf_index].loop = loop;
65     buffer[buf_index].active = 1;
66
67     int src_index = createSource(buf_index);
68
69     //Erro ao carregar source, printf na funcao createSource
70     if (src_index == -1)
71         return -1;
72
73
74     printf("Som carregado: %s\n",filename);
75
76     return src_index;
```

```
77}
78
79int SoundAL::createSource(int buf_index)
80{
81    int src_index = getUSource();
82
83    alGenSources(1,&source[src_index].source);
84
85    if(alGetError() != AL_NO_ERROR)
86    {
87        alDeleteBuffers(1,&buffer[buf_index].buffer);
88        alDeleteSources(1,& source[src_index].source);
89        printf("Erro ao carregar arquivo de som: %d", alGetError());
90        return -1;
91    }
92    //Liga a source ao buffer
93    alSourcei(source[src_index].source, AL_BUFFER, buffer[buf_index].buffer);
94    alSourcei (source[src_index].source, AL_LOOPING,  buffer[buf_index].loop);
95
96    source[src_index].active = 1;
97
98    return src_index;
99
100}
101
102void SoundAL::exit()
103{
104    for(int i = 0; i < BUFFER_SIZE_AL; i++)
105        if(buffer[i].active)
106            alDeleteBuffers(1,&buffer[i].buffer);
107
108    for(int i = 0; i < SOURCE_SIZE_AL; i++)
109        if(source[i].active)
110            alDeleteBuffers(1,&source[i].source);
111
112    alutExit();
113}
114
115void SoundAL::play(int src_i)
116{
117    alSourcePlay(source[src_i].source);
118}
119
120void SoundAL::pause(int src_i)
121{
122    alSourcePause(source[src_i].source);
123}
124void SoundAL::stop(int src_i)
125{
126    alSourceStop(source[src_i].source);
127}
128void SoundAL::stopAll()
129{
130    for(int i = 0; i < SOURCE_SIZE_AL; i++)
131        if(source[i].active)
132            alSourceStop(source[i].source);
133}
134
135int SoundAL::getUBuffer()
136{
137    for(int i = 0; i < BUFFER_SIZE_AL; i++)
138        if(!buffer[i].active)
139            return i;
140
141    return -1;
142}
143
144int SoundAL::getUSource()
145{
146    for(int i = 0; i < SOURCE_SIZE_AL; i++)
147        if(!source[i].active)
148            return i;
149
150    return -1;
151}
152
153bool SoundAL::isPlaying(int src_i)
154{
155    ALenum state;
156
157    alGetSourcei(source[src_i].source, AL_SOURCE_STATE, &state);
158
159    return (state == AL_PLAYING);
160}
```

### *B.2.13 Texto*

```
#include "text.h"

namespace txt
{
    void renderBitmapString(
            float x,
            float y,
            int spacing,
            void *font,
            char *string) {

      char *c;
      int x1 = x; //Guarda posicao rasterizada para computar espaco

      for (c=string; *c != '\0'; c++) {
        glRasterPos2d(x1,y);
        glutBitmapCharacter(font, *c);
        x1 = x1 + glutBitmapWidth(font, *c) + spacing;
      }
    }

    void* font_glut = GLUT_BITMAP_8_BY_13;

    ///ARRUMA PROJECOES
    extern void setProjecaoOrto()
    {
        glDisable(GL_DEPTH_TEST);
        glDisable(GL_LIGHTING);
        glMatrixMode(GL_PROJECTION);
        glPushMatrix(); //nao fecha
        glLoadIdentity();

        // coloca projecao ortografica 2d
        gluOrtho2D(0, wScreen, hScreen, 0);
        glMatrixMode(GL_MODELVIEW);

        glPushMatrix();
        glLoadIdentity();
    }
    extern void restauraProjecaoPerspectiva()
    {
        glPopMatrix();
        glMatrixMode(GL_PROJECTION);
        glPopMatrix(); // fecha o pushMatrix do projecaoOrtho
        glEnable(GL_DEPTH_TEST);
        glEnable(GL_LIGHTING);
        glMatrixMode(GL_MODELVIEW);
    }

    extern void renderText2dOrtho(float x, float y, int spacing, const char*pStr, ...)
    {
        char string[128];
        va_list valist; //info das variaveis
        va_start(valist, pStr); //inicia lista de argumentos das variaveis
        vsprintf(string, pStr, valist); // joga string formatado para string
        va_end(valist); // realiza operacoes de fato

        glDisable(GL_LIGHTING);
        setProjecaoOrto();
            renderBitmapString(x,y, spacing, font_glut, string);
        restauraProjecaoPerspectiva();
        glEnable(GL_LIGHTING);

    }
}
```

### *B.2.14 Carregamento de textura*

```
#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
```

```
19  int toInt(const char* bytes) {
20      return (int)(((unsigned char)bytes[3] << 24) |
21                   ((unsigned char)bytes[2] << 16) |
22                   ((unsigned char)bytes[1] << 8) |
23                   (unsigned char)bytes[0]);
24  }
25
26  //Converts a two-character array to a short, using little-endian form
27  short toShort(const char* bytes) {
28      return (short)(((unsigned char)bytes[1] << 8) |
29                     (unsigned char)bytes[0]);
30  }
31
32  //Reads the next four bytes as an integer, using little-endian form
33  int readInt(ifstream &input) {
34      char buffer[4];
35      input.read(buffer, 4);
36      return toInt(buffer);
37  }
38
39  //Reads the next two bytes as a short, using little-endian form
40  short readShort(ifstream &input) {
41      char buffer[2];
42      input.read(buffer, 2);
43      return toShort(buffer);
44  }
45
46  //Just like auto_ptr, but for arrays
47  template<class T>
48  class auto_array {
49      private:
50          T* array;
51          mutable bool isReleased;
52      public:
53          explicit auto_array(T* array_ = NULL) :
54              array(array_), isReleased(false) {
55          }
56
57          auto_array(const auto_array<T> &aarray) {
58              array = aarray.array;
59              isReleased = aarray.isReleased;
60              aarray.isReleased = true;
61          }
62
63          ~auto_array() {
64              if (!isReleased && array != NULL) {
65                  delete[] array;
66              }
67          }
68
69          T* get() const {
70              return array;
71          }
72
73          T &operator*() const {
74              return *array;
75          }
76
77          void operator=(const auto_array<T> &aarray) {
78              if (!isReleased && array != NULL) {
79                  delete[] array;
80              }
81              array = aarray.array;
82              isReleased = aarray.isReleased;
83              aarray.isReleased = true;
84          }
85
86          T* operator->() const {
87              return array;
88          }
89
90          T* release() {
91              isReleased = true;
92              return array;
93          }
94
95          void reset(T* array_ = NULL) {
96              if (!isReleased && array != NULL) {
97                  delete[] array;
98              }
99              array = array_;
100         }
101
102         T* operator+(int i) {
103             return array + i;
104         }
```

```
105
106             T &operator[](int i) {
107                 return array[i];
108             }
109     };
110 }
111
112 namespace texture {
113     GLuint loadTextureBMP(const char* filename)
114     {
115         Image* image = loadBMP(filename);
116
117         GLuint textureId;
118         glGenTextures(1, &textureId); //Make room for our texture
119         glBindTexture(GL_TEXTURE_2D, textureId); //Tell OpenGL which texture to edit
120         //Map the image to the texture
121         glTexImage2D(GL_TEXTURE_2D,                 //Always GL_TEXTURE_2D
122                      0,                             //0 for now
123                      GL_RGB,                        //Format OpenGL uses for image
124                      image->width, image->height,   //Width and height
125                      0,                             //The border of the image
126                      GL_RGB, //GL_RGB, because pixels are stored in RGB format
127                      GL_UNSIGNED_BYTE, //GL_UNSIGNED_BYTE, because pixels are stored
128                                        //as unsigned numbers
129                      image->pixels);                //The actual pixel data
130
131         delete image;
132         return textureId; //Retorna id da textura
133
134     }
135
136     Image* loadBMP(const char* filename) {
137         ifstream input;
138         input.open(filename, ifstream::binary);
139         assert(!input.fail() || !"Could not find file");
140         char buffer[2];
141         input.read(buffer, 2);
142         assert( (buffer[0] == 'B' && buffer[1] == 'M' ) || !"Not a bitmap file");
143         input.ignore(8);
144         int dataOffset = readInt(input);
145
146         //Read the header
147         int headerSize = readInt(input);
148         int width;
149         int height;
150         switch(headerSize) {
151             case 40:
152                 //V3
153                 width = readInt(input);
154                 height = readInt(input);
155                 input.ignore(2);
156                 assert(readShort(input) == 24 || !"Image is not 24 bits per pixel");
157                 assert(readShort(input) == 0 || !"Image is compressed");
158                 break;
159             case 12:
160                 //OS/2 V1
161                 width = readShort(input);
162                 height = readShort(input);
163                 input.ignore(2);
164                 assert(readShort(input) == 24 || !"Image is not 24 bits per pixel");
165                 break;
166             case 64:
167                 //OS/2 V2
168                 assert(!"Can't load OS/2 V2 bitmaps");
169                 break;
170             case 108:
171                 //Windows V4
172                 assert(!"Can't load Windows V4 bitmaps");
173                 break;
174             case 124:
175                 //Windows V5
176                 assert(!"Can't load Windows V5 bitmaps");
177                 break;
178             default:
179                 assert(!"Unknown bitmap format");
180         }
181
182         //Read the data
183         int bytesPerRow = ((width * 3 + 3) / 4) * 4 - (width * 3 % 4);
184         int size = bytesPerRow * height;
185         auto_array<char> pixels(new char[size]);
186         input.seekg(dataOffset, ios_base::beg);
187         input.read(pixels.get(), size);
188
189         //Get the data into the right format
190         auto_array<char> pixels2(new char[width * height * 3]);
```

```
191         for(int y = 0; y < height; y++) {
192             for(int x = 0; x < width; x++) {
193                 for(int c = 0; c < 3; c++) {
194                     pixels2[3 * (width * y + x) + c] =
195                         pixels[bytesPerRow * y + 3 * x + (2 - c)];
196                 }
197             }
198         }
199 
200         input.close();
201         return new Image(pixels2.release(), width, height);
202     }
203 }
```

### B.2.15 Tile

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

### B.2.16 Makefile

```
1  ############################################################
2  #                       Makefile
3  #                Friday 17 August 2012
4  ############################################################
5  CC = g++
6  CFLAGS =$(GLFLAGS) -I./ -O3 -Os -g $(PROBLENS)
7  CC_WINDOWS = x86_64-linux-gnu-g++
8  
9  PROBLENS=-Wall -pedantic -fpermissive
10 UNAME = $(shell uname)
11 OUTPUT = Amaze.out
12 
13 SRC =   button.cpp  \
14         defines.cpp   \
15         eventos.cpp    \
16         minimap.cpp    \
17         player.cpp   \
18         text.cpp           \
19         tile.cpp \
20         camera.cpp  \
21         entidade.cpp  \
22         framerate.cpp  \
23         map.cpp          \
24         model_obj.cpp  \
25         soundAL.cpp  \
26         textureloader.cpp
27 
28 OBJS = ${SRC:.cpp=.o}
29 
30 .SUFFIXES:.c.o
31 ###
32 #   libghc-opengl-dev
33 #   libghc-openal-dev
34 #   freeglut3-dev
35 #   libglui-dev
36 #   libalut-dev
37 #   glee-dev
38 ####
39 
40 
41 
42 define PROGRAM_template
43 $(1): $(addsuffix .o,$(1))
44 endef
45 $(foreach t,$(compiling),$(eval $(call PROGRAM_template,$(t))))
46 
47 
48 ifeq ($(UNAME),Linux) # Linux OS
49     GLFLAGS = -lglut -lglui -lGLU -lGL -lalut  -lopenal
50     SEARCH = dpkg -l | grep -iq
51     else
52     ifeq ($(UNAME),Darwin) # MAC OS X
53         GLFLAGS = -framework OpenGL -framework GLUT -framework OpenAL
54         SEARCH = ls /System/Library/Frameworks | grep -i
55     else #Windows
56         GLFLAGS = -lopengl32 -lglu32 -lglut32 -lglee -lalut
57         SEARCH=
58     endif
```

```
59 endif
60
61 all: system out
62
63 debug: config
64
65 system:
66     echo "System: "$(UNAME) "OS"
67     echo -n "Compiling..."
68
69 system_debug:
70     echo "System: "$(UNAME) "OS"
71     if rm *.o;\
72     then \
73         echo -n "Cleaning && ";\
74     fi;
75     echo -n "Compiling..."
76
77 config: system_debug
78     if $(MAKE) out ;\
79     then \
80         echo " " ;\
81     else \
82     echo  "Error on compiling! Probably some package is missing"; \
83         $(MAKE) check;\
84     fi;
85
86 .c.o:
87     echo -n "compiling..." $<
88     $(CC) $< -c -g $(CFLAGS) $(GLFLAGS)
89     echo  "Done"
90
91 compiling: $(OBJS)
92
93 lib: compiling
94     echo  "Done"
95     echo -n "Making lib..."
96     ar rcs libAmaze.a *.o
97
98 out: lib
99     $(CC) gamemanager.cpp -o $(OUTPUT) $(CFLAGS) -L./ -lAmaze $(GLFLAGS)
100    echo "Done.\nRun "$(OUTPUT) ;\
101
102 clean:
103    echo "Cleaning all..."
104    rm -rfv $(OUTPUT) *.o *.d *.a
105
106 run: out
107    echo "Running..."
108    ./$(OUTPUT)
109
110 valgrind: *.cpp
111    $(CC) -g -c $(SRC) $(CFLAGS) $(GLFLAGS)
112    ar rc libAmaze.a *.o
113    $(CC) -g gamemanager.cpp -o ToGring $(GLFLAGS) -L./ -lAmaze
114 #   valgrind --tool=callgrind --dsymutil=yes --trace-jump=yes ./ToGring -q --fullpath-after=string
   --show-possibly-lost=yes --trace-children=yes -v --main-stacksize=512MB
115    echo "Valgrind files available: (newer first)"
116    ls -t| egrep -i grind
117
118 windows: *.cpp
119    echo "Cross compiling to" $@
120    $(CC_WINDOWS) *.cpp -c $(CFLAGS)
121    $(CC_WINDOWS) *.o -o ./bin/x86-x64-Amaze.exe $(CFLAGS)
122    echo "done.\nRun " x86-x64-Amaze.exe "on bin directory"
123
124 check:
125    echo "Checking if all dev packages are installed"
126 #   OPENGL
127    echo -n "opengl "
128    if $(SEARCH) "opengl.*dev" ;\
129    then \
130        echo "[OK]";\
131    else \
132        echo "[MISSING!] - Install libghc-opengl-dev" ;\
133    fi;
134 #   OPENAL
135    echo -n "openal "
136    if $(SEARCH) "openal.*dev" ;\
137    then \
138        echo "[OK]";\
139    else \
140        echo "[MISSING!] - Install libghc-openal-dev" ;\
141    fi;
142 #   GLUT
143    echo -n "glut "
```

```
144     if $(SEARCH)  "glut.*dev" ;\
145     then \
146         echo "[OK]";\
147     else \
148         echo "[MISSING!] - Install freeglut3-dev" ;\
149     fi;
150#    GLUI
151     echo -n "glui "
152     if $(SEARCH)  "glui.*dev" ;\
153     then \
154         echo "[OK]";\
155     else \
156         echo "[MISSING!] - Install libglui-dev" ;\
157     fi;
158#    ALUT
159     echo -n "alut "
160#Como deveria de ser pra buscar por suporte para desenvolvedores
161#    if $(SEARCH) | grep -qi "alut.*dev" ;\
162     if $(SEARCH)  "alut.*dev" ;\
163     then \
164         echo "[OK]";\
165     else \
166         echo "[MISSING!] - Install libalut-dev" ;\
167     fi;
168#    GLEE
169     echo -n "glee "
170     if $(SEARCH)  "glee.*dev" ;\
171     then \
172         echo "[OK]";\
173     else \
174         echo "[MISSING!] - Install glee-dev" ;\
175     fi;
176
177.SILENT:
178
179#Obs
180#
181#    Bibliotecas incluidas:
182#
183#    alut-dev
184#    openal-dev
185#
186#    Descobrindo pacotes instalados:
187# $ dpkg -l | grep alut
188#
189#    No MacOS os Frameworks ficam no diretorio/System/Library/Frameworks
190# e possuem a nomeclatura semelhante a:
191# OpenAL.framework
```

### *B.2.17 README*

```
h1. README

# Windows

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

 # Mac OS

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

    Not supported yet.

# Linux

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

* Example of compiling

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

    $ make check
    Checking if all dev packages are installed
```

```
opengl [OK]
openal [OK]
glut [OK]
glui [OK]
alut [OK]
glee [OK]
```

## SUMÁRIO

## Lista de Figuras