GLPong
Pong em OpenGL

Francisco Ferreira <fmsf@student.dei.uc.pt>
Gustavo Felisberto <gustavo@student.dei.uc.pt>
Vasco Gomes <vlgomes@student.dei.uc.pt>

30 de Maio de 2004

Resumo

No natal de 1972 a Atari lança a sua primeira consola de jogos com o jogo PONG, 33 anos mais tarde 3 estudantes de comunicações e multimédia re-inventam o jogo em OpenGL.

Capítulo 1

Estrutura funcional

1.1 Flow do programa

O Pong é um jogo conceptualmente bastante simples, mas a tentativa de o implementar em OpenGL seguindo uma boa estruturação revelou-se bastante mais complexa do que o esperado inicialmente.

O jogo foi programado em C e segue uma estrutura linear.

1. Inicializar o sistema de vídeo
2. Inicializar o sistema OpenGL
3. Inicializar a bola (carrega texturas, define a estrutura da esfera, valores iniciais das coordenadas, etc)
4. Inicializar os pad's
5. Inicializar o sistema de som (pré-carregamento dos wavs a ser usados)
6. Entrar no loop principal
7. Verificar se ouve interacção com o sistema (teclado, tamanho da janela modificado etc)
8. Actualizar a bola
   - A bola envia para a parede as suas coordenadas "virtuais"
   - A parede verifica se a bola está dentro do domínio de jogo e caso tenha saído, devolve à bola o numero de choques com cada parede
   - A bola corrige as suas coordenadas
9. Desenhar as paredes
10. Desenhar os pads

    Nesta fase são executadas uma série de funções relativas aos pads:

    (a) O "AI" verifica a posição do pad controlado pelo computador e da bola e muda a sua direcção de movimento caso seja necessário
    (b) É actualizada a posição do pad do computador
    (c) Desenha o pad do computador
    (d) Verifica se é necessário actualizar a posição do pad do jogador (depende de o jogador estar a manter o teclado pressionado)
    (e) Desenhar o pad do jogador
11. Actualizar a nova imagem no ecran
12. Voltar ao ponto 6

1.2 Funções exportadas[1]

Na figura 1.1 é apresentada uma imagem com a estrutura do programa, este diagrama é um pouco complexo, seria bom apresentar de forma mais compreensível.

[1] O código integral está disponível através de ViewCVS `http://www.felisberto.net/viewcvs/practica/`

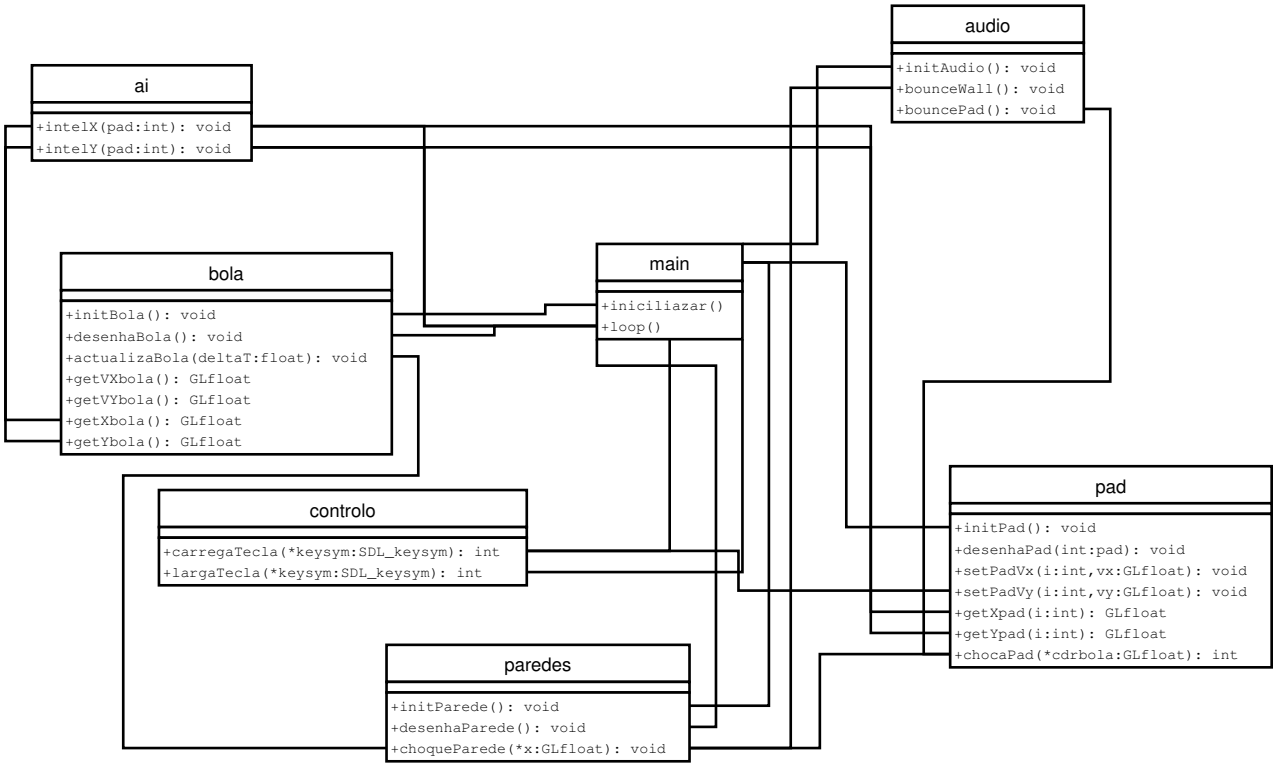

Figura 1.1: Diagrama

1.2.1 ai.h

```
void intelX(int pad);
void intelY(int pad);
```

O .c contém as funções relativas à "inteligência". As duas funções que exporta servem para fazer com que o pad indicado corrija a sua direcção em relação à bola.

1.2.2 audio.h

```
void initAudio();
void bounceWall();
void bouncePad();
```

O *initAudio()* trata da inicialização do sistema de som, pré-carregamento dos wavs e por o som de background; é chamado pelo programa no arranque.

O *bounceWall()* e *bouncePad()* são responsáveis por tocar um som quando a bola bate nas paredes ou nos pads.

1.2.3 bola.h

```
struct bola {
    GLUquadricObj *esfera;
    GLfloat posBola[3];
    GLfloat velBola[3];
    GLuint texture[1];
};
void initBola(void);
void desenhaBola(void);
void actualizaBola(float deltaT);
GLfloat getVXbola();
GLfloat getVYbola();
GLfloat getXbola();
GLfloat getYbola();
```

Aqui encontram-se as funções responsáveis pelo movimento da bola bem como a estrutura que define a bola. A função *initBola()* é chamada no arranque para criar o objecto da bola e lhe aplicar a textura; *desenhaBola()* é usada para desenhar a bola no ecran e finalmente *actualizaBola()* é chamado para actualizar a posição da bola em função do intervalo de tempo entre cada *frame*, esta função recorre a outras exportadas pela *parede.h*.

As quatro funções restantes são exportadas para que as outras partes do programa possam aceder à velocidade e posição da bola em cada momento.

1.2.4 controlo.h

```
int carregaTeclado(SDL_keysym *keysym);
int largaTeclado(SDL_keysym *keysym);
```

Estas duas funções servem para tratar do teclado sendo chamadas quando uma tecla é pressionada ou libertada. Internamente elas acedem a várias funções de outras partes do código, nomeadamente a funções exportadas pelo *pad* para controlar a direcção.

1.2.5 pad.h

```
struct pad {
    GLfloat posPad[3][2];
    GLuint texture[1];
    GLfloat velPad[3][2];
};
void initPad();
void desenhaPad(int i);
void setPadVx(int i, GLfloat vx);
void setPadVy(int i, GLfloat vy);
GLfloat getXpad(int i);
GLfloat getYpad(int i);
int chocaPad(GLfloat *crdbola);
void actualizaPad(int i);
```

Nesta parte do programa encontra-se a estrutura que contem os dados sobre os *pads*, esta parte é ainda responsável pela inicialização dos pads (transparências, eventuais texturas etc), por desenhar o *pad* e exporta ainda as funções *setPadVx()* e *setPadVy()* usadas pelo *ai.c* e pelo *controlo.c* para alterar a direcção dos *pads*. *getXpad()* e *getYpad()* devolvem a posição do *pad* e *actualizaPad()* altera a sua posição em função da sua "velocidade"[2]. A função *chocaPad()* recebe um ponteiro para um array que contém uma série de informações sobre a bola e que serve para que o *pad* possa detectar se a bola chocou com o *pad*.

1.2.6 paredes.h

```
void initParede(void);
void desenhaParede(void);
void choqueParede(GLfloat *x);
```

As funções *initParede()* e *desenhaParede()* são usadas para inicializar as paredes (bem como o fundo do cenário) e as desenhar em cada ciclo.

A função *choqueParede()* recebe um pnteiro para um array de informações sobre a bola e calcula o número de choques com cada uma das paredes. Esta função chama ainda a função *chocaPad()* para determinar se o pad se encontrava na posição certa para interceptar a bola no momento em esta iria sair do jogo.

[2] O *pad* não tem neste momento uma velocidade no sentido de $\frac{l}{t}$ mas apenas uma distância que ele percorre entre cada frame.

Capítulo 2

Estado do programa na segunda meta

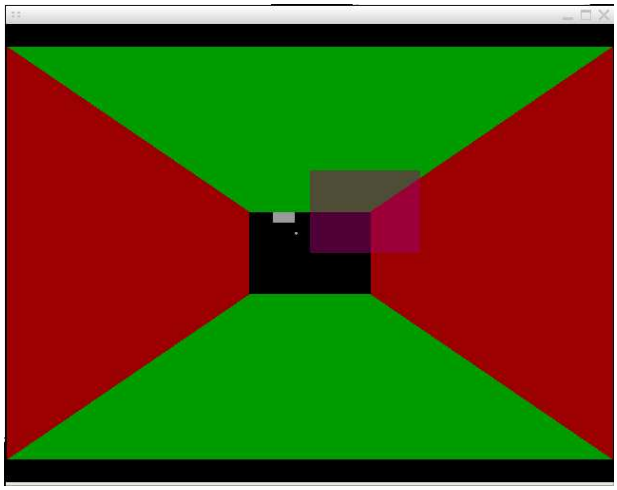

Figura 2.1: Screen Shot

A figura 2.1 mostra as actuais capacidades do sistema. Falta ainda bastante trabalho, nomeadamente nas questões gráficas de efeitos de luz.

Ao realizar este relatório ficou patente a necessidade de uma boa estruturação do programa antes de se começar a "bater código". Muitos pequenos problemas com o trabalho previamente executado foram facilmente resolvidos após a melhor estruturação das ideias, mas a reorganização permitiu que eles fossem rapidamente resolvidos.

Capítulo 3

Versão final

Na versão final do jogo temos um menu inicial onde podemos ver o computador a jogar em modo 1 on 1. Daqui podemos iniciar o jogo ou ver o menu de resultados.

Em relação ás versões anteriores o jogo ganhou um melhor sistema de som (música ambiente e efeitos sonoros), melhores efeitos gráficos (transparências e texturas) e um melhor sistema de interacção com o utilizador.

De salientar também que a nova versão corre em *BSD, Linux e Microsoft Windows (aqui esta com o problema de não mostrar o texto).