FootVR – Visualização de um Jogo de Futebol num Ambiente de Realidade Virtual

Gustavo Corrente, Ana Cantanhede, Paulo Dias

***Abstract* – This paper describes the project FootVR, carried out within the 3D Modeling and Visualization course. FootVR allows to visualize football games from log files in a virtual reality environment. For visualization had been used *Visualization Toolkit* (VTK).**

***Resumo* – Este artigo descreve o projecto FootVR realizado no âmbito da disciplina Modelação e Visualização 3D. O FootVR permite-nos visualizar jogos de Futebol a partir de ficheiros de *log* num ambiente de realidade virtual. Para a visualização foi usado o *Visualization Toolkit (VTK)*.**

I. INTRODUÇÃO

O projecto FootVR foi elaborado no âmbito da disciplina de Modelação e Visualização 3D, leccionada, como disciplina de opção, ao 5^o ano da licenciatura em Engenharia de Computadores e Telemática, da Universidade de Aveiro. Este projecto tem como objectivo visualizar um jogo de futebol num ambiente de realidade virtual, utilizando o VTK [1]. Esta visualização pode ser global (idêntica a uma câmara de TV) ou apresentar o campo tal como este é visto por um dos jogadores.

A. Objectivos

Pretende-se visualizar a evolução da posição de vários jogadores em campo (posição fornecida através de um ficheiro em ASCII, cujo formato é descrito mais à frente) num ambiente para aplicações de Realidade Virtual e Aumentada, existente no IEETA, no âmbito de um projecto de final de licenciatura no ano lectivo 2004-2005 [2]. De forma a atingir o objectivo principal, este foi dividido nos seguintes sub-objectivos:

- Especificação do formato do ficheiro: definição de um formato ASCII que contenha toda a informação relativa à posição dos jogadores e da bola ao longo de várias *frames*. Geração de alguns ficheiros para teste de acordo com este formato.
- Desenvolvimento de um modelo de um campo de futebol no ambiente VTK.
- Criação de uma aplicação em VTK que permita visualizar a evolução da posição dos jogadores e da bola ao longo do tempo, de acordo com o ficheiro fornecido.
- Estudo do *software* do ambiente virtual existente e adaptação/integração da aplicação desenvolvida permitindo utilizar o sensor de orientação para actualizar o ponto de observação do utilizador de acordo com a orientação da cabeça.

B. Descrição

Neste projecto utilizam-se os óculos *HMD i-glasses SVGA Pro* [3] (Fig.1) e o sensor de orientação *Intertraxx 2* [4] (Fig.2). Com estes dois equipamentos é possível montar um sistema que permite criar um ambiente de realidade virtual.

Fig. 1 - Óculos HMD i-glasses SVGA Pro

Fig. 2 - Sensor de orientação Intertraxx 2

II. ORGANIZAÇÃO DA SOLUÇÃO

A. Geração de jogos

Os jogos que são visualizados pelo FootVR são criados a partir de *logs* de jogos da liga de simulação 2D do Robocup [5]. Coloca-se de imediato uma questão pertinente: porque não os logs de simulação 3D? O porquê de utilizar os jogos de 2D e não de 3D prende-se com o facto de os jogos 2D serem muito mais realistas e naturais do que os jogos 3D. Assim, a ferramenta *rcg2xml* foi alterada de modo a ser incluída no pacote de *software* de desenvolvimento da liga de simulação 2D, *rcssbase* e *rcsslogplayer*. Esta ferramenta permite converter um *log* para o formato *xml*. Depois

desta modificação é possível gerar um ficheiro de texto de acordo com o formato especificado pelo utilizador.

De forma a produzir um ficheiro, alterou-se a ferramenta *rcg2xml*. Este passou a gerar os ficheiros de *log* cuja estrutura é apresentada de seguida. Assim, para cada frame é definido o tempo, a posição da bola e a posição dos 22 jogadores:

- tempo: definiu-se *Time t*, em que t é um inteiro que representa a frame que vai descrita;
- posição da bola: definiu-se *Ball x z*, em que x e z são reais que representam as coordenadas x e z da bola nessa frame (de notar que a coordenada y é sempre nula, pelo que não é necessário lê-la do ficheiro);
- posição e para a orientação do corpo e da cabeça de cada jogador: foi definido o formato *p x z bo ho*, em que p é um inteiro que representa o número do jogador (0-10 representam os jogadores da equipa A e 11-21 os jogadores da equipa B), x e z são reais que representam as coordenadas x e z desse jogador nessa frame (mais uma vez, a coordenada y é sempre nula) e bo e ho são reais que representam, em radianos, a orientação do corpo e da cabeça, respectivamente.

B. Estrutura global

Começou-se por criar a classe Player, que ”representa”um jogador. Assim, esta classe possui informação sobre a posição e a orientação do corpo e da cabeça do jogador e sobre a sua forma (neste caso, um jogador é representado por dois prismas triangulares (Fig.3), sendo que um representa o corpo e outro a cabeça.

A classe Player possui os seguintes atributos:

```
//Coordenada X da posição
float posX;
//Coordenada Y da posição
float posY;
//Coordenada Z da posição
float posZ;
// Orientação do corpo
float bodyOrientation;
// Orientação da cabeça
float headOrientation;
// Cabeça
vtkCylinderSource *head;
//Corpo
vtkCylinderSource *body;
//Actor da cabeça
vtkActor *headActor;
//Actor do corpo
vtkActor *bodyActor;
//Mapper da cabeça
vtkPolyDataMapper *headMapper;
//Mapper do corpo
vtkPolyDataMapper *bodyMapper;
```

Código 1 - *Classe Player : atributos*

e os seguintes métodos:

```
Player(void); // Construtor
~Player(void); // Destrutor
Player* New (float px, float py, float pz,
    float bodyOrient, float headOrient);
// Alterar a posição do jogador
void SetPosition(float x, float y, float z);
// Alterar a cor do corpo do jogador
void SetBodyColor(float r, float g, float b)
    ;
// Alterar a cor da cabeça do jogador
void SetHeadColor(float r, float g, float b)
    ;
// Alterar a orientação do corpo
void SetBodyOrientation(float bo);
// Alterar a orientação da cabeça.
void SetHeadOrientation(float ho);
// Retornar o GetOutput da cabeça
vtkPolyData* GetHeadOutput(void);
// Retornar o GetOutput do corpo
vtkPolyData* GetBodyOutput(void);
// Retornar o actor da cabeça
vtkActor* GetBodyActor(void);
// Retornar o actor do corpo
vtkActor* GetHeadActor(void);
```

Código 2 - *Classe Player : métodos*

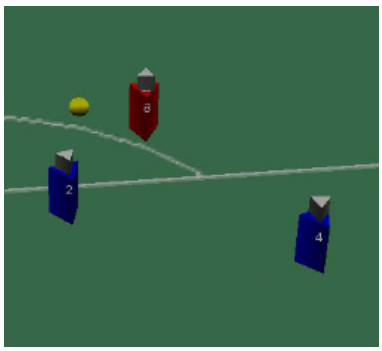

Fig. 3 - ”Corpo”dos Jogadores

De forma a armazenar a informação lida do ficheiro de *log*, foi criada a classe Frame, que armazena toda a informação de uma frame.

Cada frame (*Fig.4*) contém informação sobre o instante de tempo (*Time ID*), a posição da bola (*Ball x z*), em que *x* e *z* representam as coordenadas *x* e *z* da bola (em metros) nessa frame. De notar que a coordenada *y* é sempre nula, pelo que não é necessário constar no ficheiro. Possui ainda informação relativa a cada jogador (linha a linha) onde cada linha tem o formato *id x z bo ho* - *id* é a identificação do jogador, *x* e *z* são as coordenadas (em metros) do jogador (mais uma vez, como a coordenada *y* é sempre zero, não é necessário constar no ficheiro). Os campos *bo* e *ho* são os valores da orientação do corpo e da cabeça, respectivamente.

Deste modo, a classe Frame possui os seguintes atributos:

```
// numero da frame / tempo
int time;
// a bola
vtkSphereSource *ball;
// os jogadores
Player player[NUMBER_OF_PLAYERS_PER_TEAM*2];
// o actor da bola
vtkActor *ballActor;
// o mapper da bola
vtkPolyDataMapper *ballMapper;
```

Código 3 - *Classe Frame : atributos*

e os seguintes métodos:

```
Frame(void); // Construtor
~Frame(void); // Destrutor
// Alterar a posição do jogador numPlayer
void SetPlayerPosition(int numPlayer, float
    x, float z);
// Alterar a orientação da cabeça e do corpo
     do jogador numPlayer
void SetPlayerOrientation(int numPlayer,
    float bo, float ho);
// Alterar a posição da bola
void SetBallPosition(float x, float z);
// Alterar o tempo / numero da frame
void SetTime(int t);
// Retorna um ponteiro para a posição da
    bola
float* GetBallPos(void);
// Retorna o valor do tempo / numero da
    frame
int GetTime(void);
// Devolve um ponteiro para o actor do corpo
vtkActor* GetBodyActor(int jog);
// Devolve um ponteiro para o actor da
    cabeça
vtkActor* GetHeadActor(int jog);
// Devolve um ponteiro para o actor da bola
vtkActor* GetBallActor(void);
```

Código 4 - *Classe Frame : métodos*

```
Time 1
Ball 0.0000 0.0000
0 -47.620 0.000 -1.739 0.000
1 -15.320 15.070 2.047 0.000
2 -15.320 4.710 1.266 0.000
3 -15.320 -4.710 1.893 0.000
4 -15.320 -15.070 -2.612 0.000
5 -9.320 0.380 -0.224 0.000
6 -3.250 15.000 2.108 0.000
7 -1.250 -15.000 -0.070 0.000
8 -1.500 0.000 2.233 0.000
9 -1.500 25.000 -3.050 0.000
10 -1.500 -25.000 -0.674 0.000
11 46.500 0.000 3.008 0.131
12 21.500 20.000 -2.622 0.782
13 26.500 10.000 -2.653 -0.183
14 26.500 0.000 -2.993 -0.148
15 26.500 -10.000 2.835 0.052
16 21.500 -20.000 2.382 -0.096
17 13.500 10.000 3.140 0.658
18 13.500 -10.000 3.125 -0.658
19 1.000 20.000 0.000 -1.038
20 10.200 0.000 3.047 -0.087
21 1.000 -20.000 0.000 1.410
```

Fig. 4 - Exemplo de uma frame do ficheiro de dados gerado

A classe Baliza possui a representação VTK das balizas de futebol. O terreno de jogo é um plano coberto com a textura de um campo de futebol. De forma a ter uma boa iluminação do campo, utilizam-se quatro fontes de iluminação, localizadas nos quatro cantos do campo.

O programa principal tem a seguinte sequência de eventos:

1. inicialização de uma *frame*;
2. inicialização dos dois tipos de câmaras a utilizar;
3. desenho do campo de futebol e da sua envolvente;
4. inicialização dos números dos jogadores, a escrever no campo, de forma a tornar mais legível na visualização quais os jogadores que estamos a visualizar (equipa A *vermelha*; equipa B *azul*);
5. inicialização das balizas;
6. inicialização as luzes;
7. inicialização do renderer, definição de propriedades e adicionar os actores ao renderer;
8. abertura o ficheiro de entrada;
9. verificação a existência do tracker;
10. leitura do ficheiro de dados, de acordo com o formato descrito anteriormente.

Para interagir com o programa foram capturados/tratados os seguintes eventos:

1. Se a tecla *q* for pressionada, o programa termina;
2. Se a tecla *t* for pressionada, a câmara passa para o modo *player view* (Fig.7) (se pressionar a tecla *1* selecciona-se a equipa R; se pressionar a tecla *2* selecciona a equipa B; as teclas *F1 a F11* permitem seleccionar o jogador número 1-11, respectivamente da equipa seleccionada);
3. Se a tecla *space* for pressionado, a câmara passa para o modo *follow ball*;
4. Se a tecla *c* for pressionada, a câmara regressa ao modo *top view* (Fig.5), vista de cima do campo;
5. A tecla *r* permite efectuar um reset *Heading* (quando se está a usar os óculos de RV)
6. As teclas direccionais permitem movimentar a cabeça na direcção pressionada, quando se está em *player view mode* e não há tracker.

O último passo foi a integração dos óculos de RV e do sensor de orientação. De forma a facilitar a interação com o sensor, foi usado no projecto o módulo isense. Este fornece um conjunto de funções para interação e controlo.

C. Câmaras

De forma a visualizar os jogos, há vários modos de funcionamento para a câmara. Foram definidos dois modos de obeservação: um *top view* (em que é visto o jogo de cima, com a câmara a seguir a bola) e um *player view* (em que se pode escolher um jogador e visualizar no monitor o que esse jogador está a ver, podendo virar a câmara/cabeça utilizando as teclas direccionais).

Top View

Este modo permite visualizar o jogo visto de cima (Fig.5), idêntico a um jogo visto na televisão. Contém ainda a seguinte particularidade: quando a bola se aproxima da grande área a câmara muda a sua posição para trás da baliza (Fig.6).

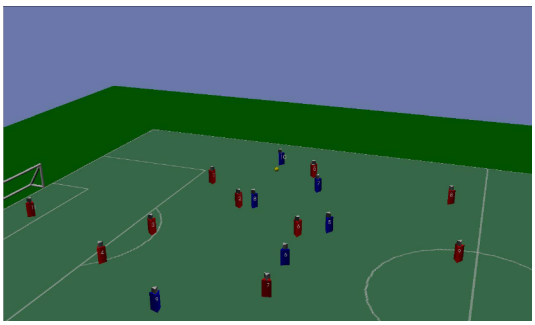

Fig. 5 - Câmara *TopView* – vista lateral (TV)

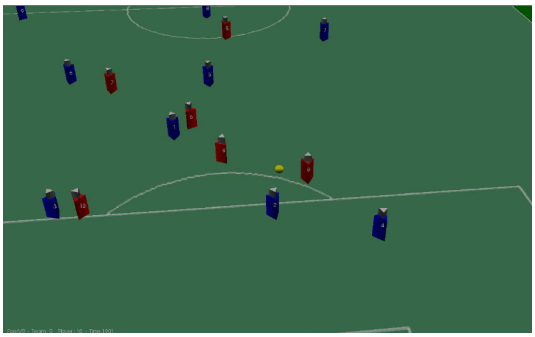

Fig. 6 - Câmara *TopView* – vista da área

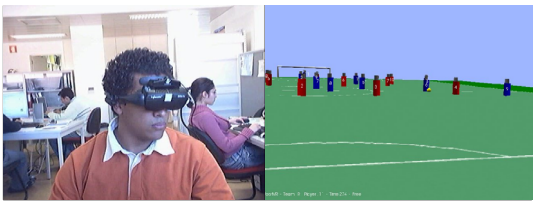

Fig. 7 - *(a)*-utilizador *(b)-Player View*

Player View

Este modo permite visualizar o jogo tal como um dos jogadores em campo, ou seja, a câmara está posicionada na cabeça de um jogador e movimenta-se de acordo com os movimentos do utilizador no que respeita à orientação da cabeça. Há ainda mais duas possibilidades: a primeira permite que se esteja sempre a olhar para a bola. A segunda permite utilizar a orientação da cabeça da pessoa que tem os óculos (Fig.7) colocados e, de acordo com essa orientação, mudar o ângulo de visão com que está a visualizar o campo (devido ao uso do sensor de orientação montado nos óculos de realidade virtual).

III. RESULTADOS

Nas figuras 5 e 6 foram apresentados resultados da vista Top View. Nas figuras 8 e 9 são apresentadas duas frames da vista Player View.

IV. CONCLUSÃO

Todos os objectivos propostos foram atingidos. Com este trabalho uma melhoria obvia, seria melhor visualmente, usando modelos mais complexos, nomeadamente modelos humanos. Porém, no caso de serem mais realistas são modelos difíceis de encontrar gratuitamente. O acesso ao sensor (tracker) revelou-se bastante fácil (usou-se o módulo isense). As principais dificuldades centraram-se no uso do VTK que possui um ajuda bastante confusa e desorganizada. De referir ainda o erro cumulativo existente no sensor (tracker) o que leva a que se tenha de fazer reset de tempos a tempos. Uma evolução futura seria usar todos os equipamentos na sua versão *wireless* bem com um sensor de posição, de modo a o utilizador poder movimentar-se livremente pelo campo de futebol como se fosse um árbitro.

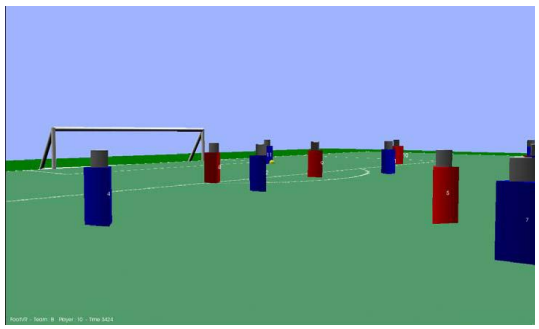

Fig. 8 - Câmara *PlayerView* - avançado da equipa B

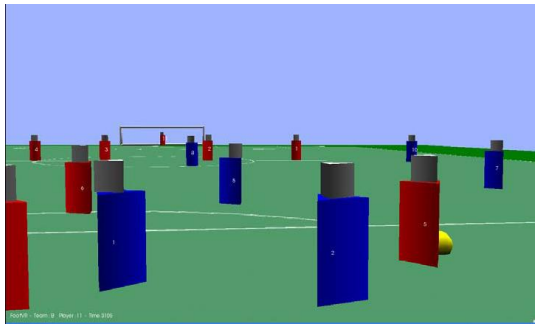

Fig. 9 - Câmara *PlayerView* - guarda redes da equipa B

BIBLIOGRAFIA

[1] *VTK Home Page* : `http://www.vtk.org` *on-line* em Março 2006.

[2] Mário Cruz, Paulo Moreira, Paulo Dias, Joaquim Madeira, Beatriz Sousa Santos,*Desenvolvimento de um ambiente para Realidade Virtual e Aumentada*, Electrónica e Telecomonicações, Vol. 4, nº 5, Setembro 2005, pp.579.

[3] *i-O Display Systems LLC* : `http://www.i-glassesstore.com` *on-line* em Março 2006.

[4] *isense Home Page* : `http://www.isense.com` *on-line* em Março 2006.

[5] *Robocup Home Page* : `http://www.robocup.org` *on-line* em Março 2006.