Artigo

SIBGRAPI'91
IV Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens

SA3D, UM SISTEMA DE ANIMAÇÃO 3D PARA AMBIENTES COMPUTACIONAIS DE RECURSOS LIMITADOS

Carlos Gerardo de São Paulo
(INESC e ISCTE - Lisboa, Portugal)

José Miguel Salles Dias
(INESC e ISCTE - Lisboa, Portugal)

Jorge Vitória
(INESC e ISCTE - Lisboa, Portugal)

Manoel Gamito
(INESC e ISCTE - Lisboa, Portugal)

Pedro Faria Lopes
(INESC e ISCTE - Lisboa, Portugal)

Mário Rui Gomes
(INESC e ISCTE - Lisboa, Portugal)

Página em branco na versão original impressa.

SA3D, um Sistema de Animação 3D para Ambientes Computacionais de Recursos Limitados

Carlos Gerardo de São Paulo *

José Miguel Salles Dias *,**

Jorge Vitória *

Manuel Gamito *

Pedro Faria Lopes *, **

Mário Rui Gomes *

* INESC, Instituto de Engenharia de Sistemas e Computadores

R. Alves Redol, 9, 1000. Lisboa, Fax 351 1 525843

** ISCTE, Instituto Superior de Ciências do Trabalho e da Empresa

Av. das Forças Armadas, Edifício ISCTE 1600 Lisboa, Fax 351 1 7935300

Sumário

Descreve-se a arquitectura do Sistema de Animação 3D (SA3D), um Sistema de Animação Modelada por Computador desenvolvido no projecto CAD/CAM no INESC, tendo em conta as limitações impostas por um ambiente de microcomputador. Referem-se sob o ponto de vista da funcionalidade os diferentes módulos constituintes : Modelação Geométrica 3D, Síntese de Movimento (Animação), Síntese de Imagem com Elevado Grau de Realismo e Edição 2D Interactiva, incluindo Digitalização. Referem-se ainda os diversos ambientes informáticos, para os quais o SA3D foi transportado.

Palavras Chave : Animação por Computador, Modelação Geométrica 3D, Edição Gráfica 2D, Aproximação e Interpolação Spline, CGI-INESC, VISTA GRAPHICS.

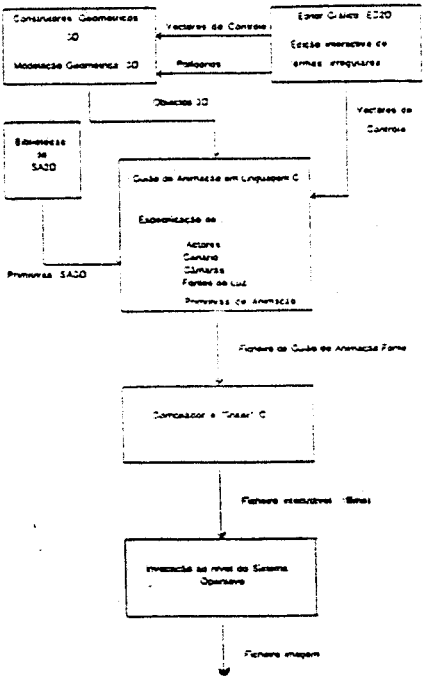
1. Introdução

No âmbito do projecto CAD/CAM do Instituto de Engenharia de Sistemas e Computadores, tem sido desenvolvida uma linha de investigação no domínio da Animação por Computador [Lopes88b].

De facto desde Março de 1987, altura em que se finalizou o desenvolvimento do primeiro sistema de animação tridimensional, o ANIMED [Lopes87], um conjunto de conceitos e tecnologias relacionados relacionados com esta área científica foram consolidados. Um grande investimento foi feito em domínios científicos, tais como a Modelação Geométrica 3D, as Interfaces Homem-Máquina Gráficas, a Edição Gráfica 2D e 3D, a Síntese de Movimento com aproximação cinemática, por interpolação de forma ou uma mistura destas duas aproximações, e a Síntese de Imagem com os modelos e métodos : "wire-frame", constante, Gouraud, Phong e Ray-Tracing.

Por outro lado, em finais de 1988, o INESC lançou, com Carlos Gerardo de São Paulo, um projecto para a realização de um novo Sistema de Animação, capitalizando uma larga experiência, deste último, de mais de 5 anos de trabalho em Animação por Computador, quer ao nível de investigação e desenvolvimento, quer de produção.

O Sistema de Animação que se apresenta neste artigo, potencia assim uma larga experiência de Carlos Gerardo de São Paulo, de cujas ideias surgiu a arquitectura do Sistema, com a experiência de 3 anos de investigação realizada no INESC, em Animação por Computador.



2. Arquitectura do Sistema SA3D

O Sistema SA3D, um sistema de Animação Modelada por Computador [M. Thalmann85], no qual uma realidade complexa tridimensional é representada e animada nas estruturas internas de um sistema computacional, utiliza uma aproximação de especificação da animação que se baseia na aproximação/interpolação de parâmetros da cinemática do movimento, como o sejam ângulos de rotação, factores de escala, valores de translacção ou parâmetros de camaras virtuais e fontes de luz [M. Thalmann85].

O sistema é eminentemente procedimental, só possuindo um componente (o editor 2D, ED2D) que é interactivo. De facto se o utilizador/animador pretende modelar uma serie de objectos 3D, então tem de invocar ao nivel do sistema operativo (o sistema está implementado em Unix ou MS/DOS) um conjunto de construtores geométricos procedimentais, com os quais pode criar primitivas geométricas. Um modelo geometrico mais sofisticado pode ser conseguido à custa da instânciação de diversas primitivas, no espaço do mundo. O utilizador pode visualizar rapidamente o modelo obtido numa projecção ortogonal, atraves de um modelo de arame ou utilizando outros modelos de síntese de imagem ("Flat" e "Phong"). Se o desejar, pode ainda definir as caracteristicas de uma câmara virtual (cujos parâmetros pode "animar"), ao vizualizar o modelo geometrico criado.

Por outro lado se o utilizador pretende estabelecer uma animação hierarquica dos objectos 3D criados, definindo actores e subactores a partir dos mesmos [Lopes88b], deve escrever um guião de animação. Nesse guião, utilizando as primitivas de animação do SA3D, o utilizador especifica procedimentalmente a hierarquia entre actores, estabelece as transformações, para cada quadro da animação, do espaço local dos actores para o espaço do mundo e manipula um numero praticamente ilimitado de câmaras virtuais e um numero finito de fontes de luz, podendo quando o desejar comutar entre elas. O guião é escrito em Linguagem C, devendo ser compilado e ligado as bibliotecas do SA3D de modo a ser produzido um ficheiro executavel que corresponde ao filme criado. Quando tal ficheiro é executado produz-se, para cada quadro, um ficheiro com a descrição da imagem 3D calculada e projectada em 2D (imagem sintetizada). Este ficheiro imagem é visualizado no ecran gráfico mediante um comando do SA3D. Se se desejar produz-se ainda, para cada quadro, um ficheiro (com informação "bitmap") com um formato compatível com um Sistema a jusante que conecta com um Sistema de Video profissional.

A arquitectura do Sistema SA3D está descrita no diagrama da Fig. 1. Como se vê pelo diagrama o SA3D introduz o conceito de vector de controlo uni ou bidimensional, que será desenvolvido em pormenor de seguida e que "controla" a variação, entre imagens chave, dos diversos parâmetros da animação.

Vectores de Controlo

No SA3D a avaliação do valor instantâneo de qualquer grandeza, por exemplo a componente *x* da posição de uma câmara virtual, ou um dos ângulos de rotação de um braço de robot, é dependente de um vector de controlo. Este ultimo corresponde a um ficheiro de texto com o seguinte formato :

< código do algoritmo >

P_1 P_2 P_3 P_4 ..... P_n

... em que P_1 ... P_n são os valores de n pontos de controlo da grandeza e *< código do algoritmo >* codifica o modelo matematico parametrico (que depende de um parâmetro $0 <= u <= 1$) de interpolação ou aproximação utilizado na avaliação da grandeza.

Têm-se as seguintes leis :

Código	Modelo
1	Interpolação Linear;
2	Splines interpolantes de Kochanek-Bartels;
3	Aproximação B spline com interpolação dos pontos extremos do vector utilizando pontos fantasma;
4	Aproximação B spline com interpolação dos ponto extremos do vector por triplicação dos mesmos;
5	Aproximação B spline, sendo avaliada uma curva fechada.

O modelo de spline interpolante Kochanek-Bartels [Barsky87] no caso bidimensional, interpola (por definição) os pontos de controle. Este modelo garante a continuidade da tangente a curva entre cada dois troços consecutivos da curva (cada troço é dependente de dois pontos consecutivos de controle). O modelo utiliza os polinomios cúbicos de Hermite, e permite controlar localmente, por manipulações nas derivadas de entrada e de saída em cada troço, o grau de continuidade, de tensão e de assimetria ('bias') do troço de curva .

No sistema SA3D, estabeleceu-se que P_1, P_2 e P_3 correspondem respectivamente aos valores da tensão, continuidade e 'bias', globais para toda a curva o que garante um grau elevado de controlo da respectiva curvatura. Os valores da tensão, continuidade e bias estão compreendidos entre -1 e 1, como se pode observar na Fig.2 .

Por seu lado o modelo de aproximação B spline utilizando polinomios cúbicos [Barsky87] e [Watt89], que neste Sistema interpola os pontos extremos do vector de controlo, garante a continuidade das derivadas parametricas de 1ª ordem e de 2ª ordem, entre cada dois troços de curva consecutivos (cada troço é controlado por quatro pontos de controle). Este facto é importante para a especificação cinematica do movimento, dado que a continuidade da 1ª e 2ª derivadas parametricas correspondem respectivamente à continuidade da velocidade e aceleração o que garante um movimento sintetizado suave, Fig.2 . O modelo B spline ainda garante que modificações num ponto do vector de controle apenas corresponde a uma modificação local da curva (dos quatro troços que são influenciados por esse ponto).

Por estas razões, no SA3D, o modelo B spline é o mais utilizado em síntese de movimento, sendo os restantes, nomeadamente as splines de interpolação e o modelo B spline fechado, mais utilizados em modelação geometrica 3D.

3 Edição Interactiva 2D

O Sistema SA3D, possui um editor interactivo 2D, **ED2D** com capacidade para digitalizar e editar formas irregulares bidimensionais (introduzir, apagar, mover e alinhar pontos e ainda duplicar e triplicar pontos de formas). Tais formas vão constituir essencialmente vectores de controlo que serão posteriormente utilizados na especificação e controlo da animação de actores, de câmaras virtuais e de fontes de luz e ainda na modelação geométrica 3D. Assim o editor produz e adquire ficheiros compatíveis com a organização dos vectores de controlo já referida anteriormente (suportando os modelos linear, Kochanek Bartels Spline e B spline).

Este editor foi desenvolvido utilizando o sistema grafico CGI-INESC [INESC88], que se mostrou muito versatil para a implementação de um editor gráfico de formas irregulares 2D, já que ofereçe nomeadamente, a manipulação de segmentos e a independência dos dispositivos tais como a mesa digitalizadora e o terminal grafico. Este facto veio a mostrar-se importante pois existiram uma multiplicidade de ambientes de implementação do **ED2D** (terminal gráfico com a placa Vista Graphics ou terminal gráfico Tektronics).

4 Modelação Geométrica 3D

O sistema SA3D suporta a descrição de modelos geometricos tridimensionais, sob a forma de malhas de polígonos [Foley84]. Cada polígono possui a seguinte descrição:

- lista vértices (com as coordenadas dos pontos cartesianos e coordenadas dos vectores normais nos pontos) percorridos no sentido anti-horario.

- vector normal ao polígono

- cor, RGB do polígono

- índice de uma tabela de materais (mármore, cobre, plástico, etc.)

A criação de modelos 3D é feita à custa de construtores geométricos procedimentais não interactivos, invocados do nivel do sistema operativo que produzem nomeadamente as seguintes primitivas geométricas: polígonos regulares convexos, prismas regulares, pirâmides regulares, esferas, superfícies de revolução, malhas de polígonos criadas pela propagação de uma curva ao longo de um caminho. Quaisquer das primitvas geometricas anteriores pode ser instanciada no mundo, podendo-se, deste modo, definir modelos geometricos complexos :

Polígonos regulares convexos : **$ poligono** arq_saida raio n_lados

Prismas regulares: **$ prisma** arq_saida raio_base raio_topo altura n_lados

Pirâmides regulares: **$ piramide** outfile raio_base altura n_lados

Esferas: **$ esfera** arq_saida raio n_paralelos n_meridianos

Superfícies de revolução: **$ revol** perfil.x perfil.y arq_saida ângulo n_intevalos

Dados os nomes dos ficheiros de entrada e de saída, o ângulo de rotação e o número de intervalos, são gerados os polígonos correspondentes às faces constituintes de uma superfície de revolução. Os ficheiros de entrada correspondem a um vector de controle bidimensional (perfil.x e perfil.y) que definem o perfil que é revolucionado. A revolução é feita entre zero graus e o ângulo dado como argumento, em torno do eixo dos yy'.

Propagação de uma curva ao longo de um caminho:

$ propag secção.x secção.y resol_u caminho.x caminho.y resol_v modelo.m escala

É possivel a definição de uma malha de polígonos, propagando uma secção curva ao longo de um caminho. A secção é descrita no plano XY à custa de um vector de controlo bidimensional, *secção.x* e *secção.y*,

podendo assim corresponder a um segmento de recta, uma secção linear ou uma secção curva aberta ou fechada, dependendo tal facto do código do algoritmo utilizado no vector de controle.

O caminho de propagação da secção corresponde igualmente a uma curva descrita no plano XY. Essa curva é agora controlada pelo vector *caminho.x* e *caminho.y*. Definem-se igualmente o número de pontos avaliados na secção (resolução paramétrica u) e o número de vezes em que a secção é propagada ao longo do caminho (resolução paramétrica v). A propagação é realizada de tal forma que a secção se encontra sempre disposta ortogonalmente em relação a curva de propagação. A malha de polígonos resultante é guardada num ficheiro de saida *modelo.m*, sendo visualizada em projecção ortogonal à medida que é construída.

Por outro lado a medida que a secção se propaga pode-se opcionalmente definir uma transformação de escala, que se aplica à própria secção. Essa transformação define-se através de um vector de controlo uni-dimensional.

A generalidade deste comando permite por ex. propagar uma spline fechada ao longo de um segmento de recta, obtendo-se um tubo. Aplicando-se uma transformação de escala a secção, por intermédio de uma outra spline, ao longo da propagação, obtém-se um cálice (ver Fig. 3). Quando a secção e um segmento **de** recta a propagação corresponde a uma extrusão simples (ver Fig. 4). Por outro lado, se a secção e o caminho correspondem a splines fechadas obtêm-se, na malha de polígonos resultante, toros.

Refira-se que as curvas associadas aos vectores de controlo, que podem ser definidas com um vulgar editor de texto, ou, com mais precisão, utilizando o editor **ED2D** (ver secção 3), podem ser rapidamente visualizadas no écran gráfico com os comandos **plot** (vector bidimensional) e **spline** (vector unidimensional). Ex :

$ plot anel.x anel.y -aval 2020 avaliações

$ spline escala -aval 3030 avaliações

Os modelos criados podem ser visualizados graficamente, em projecção ortogonal e modelo de arame, por utilização do comando **projec** :

$ projec calice.m

Instanciação

As primitivas geométricas criadas com os comandos anteriores podem ser sucessivamente instanciadas no mundo, mediante a utilização do comando **instance** :

$ instance arq_primitivas modelo_saida

O ficheiro de texto *arq_primitivas*, possui uma entrada por cada primitiva que se pretende instanciar, com o seguinte formato (linhas precedidas por *corresponem a comentários) :

< nome da primitiva > < tx > < ty > < tz > < sx > < sy > < sz > < rx > < ry > < rz >

As transformações são aplicadas, à primitiva, pela ordem seguinte: 1^{o} as escalas (*sx*, *sy* e *sz*), em 2^{0} as rotações: em torno dos xx', depois em torno dos yy' e finalmente em torno dos zz' e em 3^{0} as translacções (*tx*, *ty* e tz). Um caracter '-' em qualquer dos parâmetros, desactiva a transformação correspondente.

O comando cria um objecto final em *modelo_saida* onde se posicionam todas as primitivas, transformadas de acordo com os parâmetros contidos no ficheiro *arq_primitivas*.

5. Síntese de Movimento

Os modelos geométricos 3D são, como vimos, criados num espaço local de coordenadas e estão descritos num ficheiro, num formato interno do SA3D, nomeadamente em malhas de polígonos. O animador, utilizando primitivas do SA3D, deve instânciá-los no espaço do mundo sujeito à transformação de câmara virtual, transformando-os em actores. Essas primitivas correspondem a aplicação, aos objectos 3D, de transformações de translacção, rotação (nos três eixos coordenados do espaço local) e mudanças de escala, podendo ser concatenadas pela ordem que se desejar. Tudo isto é descrito num guião de animação. Fácil é agora compreender que o animador pode iterar o processo de instânciação de um objecto no espaço do mundo, mediante por exemplo uma transformação de translacção, variando em cada iteração os parâmetros (valores *x*, *y* e *z*) da translacção. O animador associa a cada parâmetro um vector de controlo que especifica pontos chave (de imagens ou posições chave) que os parâmetros devem obedecer ao longo da sua variação [Watt89].

Este processo aplica-se a todos os parâmetros das diversas primitivas de síntese de movimento, definindo-se, por meio de vectores de controlo, a variação dos mesmos ao longo das imagens chave.

Seja então a variação de posição de um actor (translacção) o vector **P** :

$$\mathbf{P} = \mathbf{P}(\mathbf{k}) = [\, x(\mathbf{k})\ y(\mathbf{k})\ z(\mathbf{k}) \,] \qquad (1)$$

... em que **k** é a imagem (ou posição) chave.

Cada componente de [$x(\mathbf{k})$ $y(\mathbf{k})$ $z(\mathbf{k})$], é avaliada em função de um vector de controlo, de imagem-chave para imagem-chave.

A variação temporal cinemática das imagens chave é conseguida a custa de outro vector de controlo (o vector de sistema *time*), de que depende a variação da imagem chave **k** em função do tempo t :

$$\mathbf{k} = \mathbf{k}(t) \qquad (2)$$

Nesta metodologia o animador especifica $\mathbf{P} = \mathbf{P}(\mathbf{k})$ e $\mathbf{k} = \mathbf{k}(t)$, com o auxílio de vectores de controlo e o sistema calcula $\mathbf{P} = \mathbf{P}(t)$ que corresponde à sequência de animação final resultante.

Sejam então **Tx**, **Ty** e **Tz**, vectores de controlo (com a lei **B spline**), dos parâmetros de uma translação e **ALFA** um vector de controle de um ângulo de rotação em torno do eixo dos yy', transformações estas a aplicar a um modelo geométrico descrito no ficheiro **cubo.m**. Numa iteração do guião de animação teríamos:

```
tx = eval("TX", t);/* avalia a variável tx, em função do vector   */
ty = eval("TY", t);/* de controle TX, para o instante de tempo   */
tz = eval("TZ", t);/* normalizado  t  */
alfa = eval("ALFA", t);/* idem, para o ângulo de rotação alfa */
translat(tx, ty, tz);/* calcula uma transformação de transla. */
rotate("y", alfa); /* calcula uma transformação de rotação  */
scale(2.0, 1.0, 2.0);/* calcula uma transformação de escala */
execw("cubo.m");/* aplica  as transformações concatenadas. */
/* 1º a escala em 2º a rotação e em 3º a translacção, ao  */
/* objecto 3D cubo.m e  acrescenta o resultado a um ficheiro */
/* imagem, sujeito  ainda à transformação de câmara  virtual */
```

É possível a animação hierárquica de objectos 3D (constituídos como actores e sub-actores), como por exemplo um braço articulado de um robot (ver Fig. 5)

O sistema SA3D suporta ainda a definição de uma multiplicidade de câmaras virtuais, cada uma possuindo os seguintes parâmetros :

- posição cartesiana no espaço do mundo, *pos*;
- posição de interesse no mesmo espaço, *pin*;
-ângulo de rotação segundo o eixo [*pos pin*], *tilt*
- distância focal, *focal*.

A distância focal, medida em milímetros, permite modelar uma lente teleobjectiva, grande angular, etc.. Cada parâmetro possui o correspondente vector de controlo, que o faz depender da imagem chave. As câmaras estão numeradas de 0 a ... n . Para activar, no guião, uma câmara (por ex. a nº 0) o animador apenas tem de escrever :

```
...
camera(0)
...
```

... e o sistema tenta identificar os seguintes vectores de controlo, respeitantes aos parâmetros da câmara : *pos* -> fromx0, fromy0 e fromz0; *pin* -> atx0, aty0, atz0; *tilt* -> tilt0; *focal* -> focal0, predefinidos pelo animador e que controlam a coreografia da câmara nº 0. Como exemplo, de notar que para uma câmara seguir um determinado actor basta que, a partir de determinado instante, os vectores de controlo atx0, aty0 e atz0 coincidam com os vectores de controlo da translacção do actor.

Na escrita do guião de animação, o animador é livre de utilizar todas as estruturas de controlo e estruturas de dados que a linguagem C suporta, podendo sempre inquirir ao sistema qual o nº da imagem chave corrente (**k**), qual o instante de tempo corrente (**t**) (assumindo o instante 0 no início da animação), quantas imagens devem ser produzidas (**nf**), qual o nº da imagem final (**ff**), etc..

Os valores iniciais para **ff**, **nf**, **k**, **t**, etc. são atribuídos em tempo de execução, por especificação na linha de comando. Assim, assumindo que o ficheiro executável (do filme) se denomina **robot**, têm-se as seguintes opções da linha de comando :

$ robot image **-ff** < nº da imagem final > **-fc** < nº da imagem corrente > **-nf** < nº de imagens a serem produzidas > **-nocam -video -prev -in**

Algumas opções merecem um comentário :

-nocam	desactiva todas as transformações de câmara.
-video	produz cada ficheiro imagem em formato compatível com um equipamento de gravação vídeo profissional.
-prev	(de "preview") mostra até 16 imagens simultâneas no écran.
-in	(de "inbetween") mostra todas as imagens calculadas (não apaga as intermédias).

Um dado ficheiro imagem, por ex. image1, produzido pelo guião de animação pode ser visualizado em modelo de arame com o comando **wire** :

$ wire image1

6. Modelo de Iluminação

O processo global de pintura de objectos de uma cena é constituido por um encadeamento de de sub-processos formando um *pipeline* de visualização:

1 - Remoção de polígonos ocultos

2 - Remoção de partes de polígonos ocultos por polígonos mais próximos do observador;

3 - Preenchimento do interior de cada polígono que passou pelas fases 1) e 2);

4 - Cálculo da intensidade luminosa para cada elemento de imagem (*pixel*) no interior de um polígono.

O cálculo da intensidade luminosa no *pixel*, entra em linha de conta com as influências do meio ambiente tais como fontes de luz e com as caracteristicas da superfície do objecto (rugosidade, brilho, côr, reflectância.....).

No SA3D, tendo em vista a implementação desse *pipeline* de visualização foi desenvolvido um modelo de iluminação que constitui a parte final do processo de pintura dos polígonos. Foi seleccionado o modelo de iluminação de Phong [M.-Thalmann85] [Amanatides87] por ser um algoritmo que proporciona imagens com um bom nivel de realismo e que não dispende muito tempo de processamento.

Segundo este modelo a intensidade luminosa divide-se em três componentes distintas :

$$I = Ia + Id + Is \qquad (3)$$

...em que

Ia - componente **ambiente**: resulta da energia luminosa que se encontra distribuida uniformemente por todo o espaço.

Id - componente **difusa**: resulta da reflexão irregular, no objecto, da energia luminosa vinda das fontes.

Is - componente especular: resulta da reflexão da energia luminosa proveniente das fontes, segundo a direcção de espelho (obedecendo a lei de reflexão de Snell).

Para dissimular a estrutura poligonal dos objectos recorreu-se a uma interpolação Phong que se destinou a interpolar bilinearmente a normal no interior de um polígono em função das normais nos seus vertices. Deste modo conseguiu-se obter um sombreado que varia continuamente ao longo da superfície do objecto aumentando o seu grau de realismo.

De notar que se esta interpolação Phong não for efectuada, obtém-se um sombreamento constante, ao longo da superficie de cada polígono. Existe, assim, a possibilidade de se optar entre um sombreamento constante ou um sombreamento Phong (mais realista mas tambem mais dispendioso).

O modelo foi testado recorrendo a um algoritmo de pintura de polígonos de tipo *scan-line* [M.-Thalmann85] sem entrar em linha de conta com facetas ocultas para maior simplicidade. Os resultados obtidos com imagens de demonstração, corresponderam aos requisitos de bom nivel de realismo especificado.

Como se referiu na secção 2 (Arquitectura do Sistema SA3D), torna-se possivel a visualização de um objecto 3D, nos diversos modelos de iluminação suportados ("wireframe", constante e Phong), incluindo a definição de uma câmara virtual, cujos parâmetros podem eventualmente ser animados, não existindo a necessidade de se escrever um guião de animação especifico para esse efeito. Assim supondo que existem os vectores de controlo da camara 0 (ver secção 5), na directoria corrente, invoca-se ao nivel do Sistema Operativo :

$ face -m < arquivo modelo 3D > **-ff** < nº da imagem final > **-fc** < nº da imagem corrente > **-nf** < nº de imagens a serem produzidas > **-nocam -video -prev -in -bf -phong -const**

Em adição às opções anteriormente descritas, existem mais algumas, que mereçem uma referencia: **-bf** (remove as facetas ocultas), **-phong** (visualiza com interpolação phong) , **-const** (visualiza sem interpolação Phong)

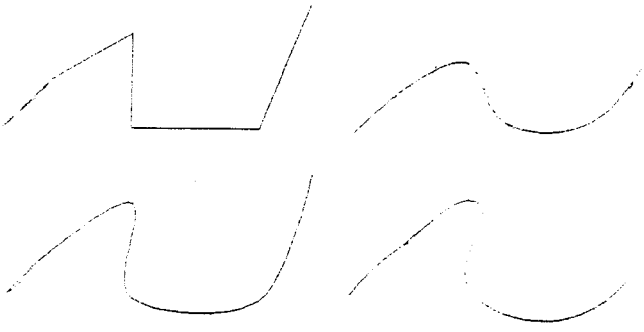


Fig. 2: Da esquerda para a direita e de cima para baixo temos quatro curvas todas com o mesmo vector de controlo: modelo linear, modelo B spline e duas splines modelo Kochanek-Bartels (primeira spline: tensão = 0, continuidade = 0, bias = 0; segunda spline: tensão = -0.6, continuidade = 0, bias = 0)

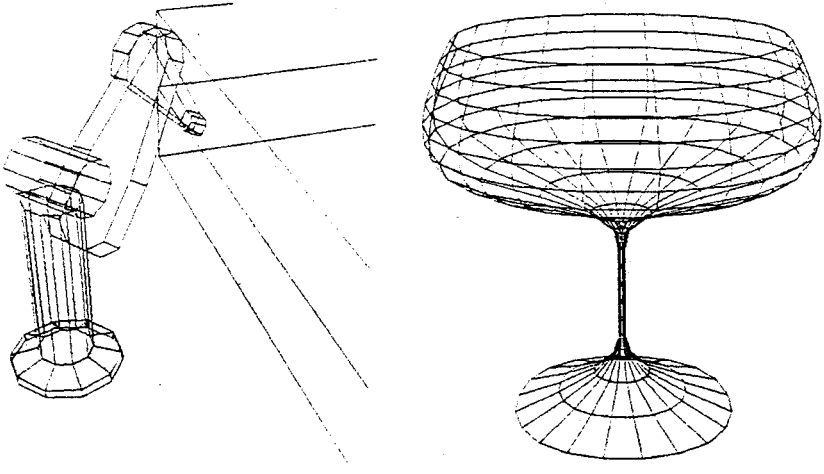


Fig 3: Braço de robot Puma com 4 graus de liberdade.

Fig. 3: Modelo criado com invocação de construtor geométrico propag: propagação de spline fechada ao longo de linha com escalamento controlado por spline.

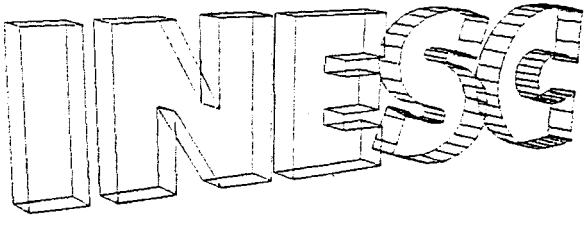


Fig 4: Exemplo de modelos criados através de extrusão utilizando o construtor geométrico propag.

7. Ambiente de Desenvolvimento

O sistema SA3D foi desenvolvido, em linguagem C [Kernighan78] tendo em vista as limitações de um ambiente PC AT com Sistema Operativo MS-DOS. Utilizou-se como suporte gráfico a placa Vista Videographics da AT&T. Apenas algumas rotinas de baixo nível foram escritas em "assembly", nomedamente uma rotina que inicializa a placa com o formato PAL GB, PAL M ou NTCS, uma outra que escreve um vector, com uma cor (definida em RGB) e uma função lógica (cópia ou xor) no "frame buffer", ou uma rotina que apaga todo o "frame buffer" .

O editor ED2D foi desenvolvido inicialmente em ambiente Unix, utilizando um terminal gráfico Tektronix, sobre o sistema gráfico CGI-INESC. Com a escrita de um "device driver" deste último para a placa AT&T Vista, possibilitou-se o transporte do ED2D para o ambiente MS-DOS já descrito, onde já residiam os restantes módulos do SA3D.

Refira-se por fim, que o sistema foi concebido com o cuidado de permitir um fácil transporte para outros sistemas gráficos e outros sistemas operativos.

8. Conclusões

Neste artigo descreveram-se com algum pormenor, os diversos módulos constituintes do Sistema de Animação por Computador SA3D, um sistema especificamente concebido para ambientes de recursos computacionais limitados. No sentido de testar as potencialidades do sistema foram desenvolvidas algumas sequências de animação. Nessas sequências são apresentadas imagens em modelo de arame, exemplificando a animação hierárquica numa aplicação de engenharia (robot de soldadura numa linha de montagem), e é testado, noutra sequência, o modelo de síntese de imagem (esfera de material plástico ilustrando o modelo de iluminação com as diferentes componentes da luz : ambiente, difusa e especular).

Os resultados já alcançados prespectivam a aplicação do sistema em diversas áreas donde se destacam : a simulação em ambiente industrial mediante a sua ligação a sistemas de simulação não gráficos; publicidade e filmes didácticos; ligação a outros sistemas de modelação geométrica, etc.

Bibliografia

[Amanatides87] Amanatides, "Realism in Computer Graphics : A Survey", Computer Graphics & Applications, January 1987, pp 49-56

[Barsky87] B. Barsky, R. Bartels, J. Beatty, An Introduction to Splines for use in Computer Graphics and Geometric Modeling, Morgan Kaufmann Publishers, Inc., 1987

[Foley84] J. D. Foley, A. Van Dam, *Fundamentals of Computer Graphics*, Addison-Wesley Publishing Company,1984

[INESC88] INESC, "Binding de CGI para C", 1988

[Kernighan78] B. Kernighan, D. Ritchie, The C Programming Language, Prentice-Hall, Inc., 1978

[Lopes87] P. F. Lopes, J. M. Salles Dias, "ANIMED, ANIMATION EDITOR",

[Lopes88a] P. F. Lopes, J. M. Salles Dias, "Animação por Computador de Objectos Sintetizados ou Adquiridos", 1º Simpósio Brasileiro de Computação Gráfica Processamento de Imagens, Rio de Janeiro Abril 1988

[Lopes88b] P. F. Lopes, M. R. Gomes, "Computer Animation in Portugal", 1st Luso-German Meeting, Eurographics Portuguese Chapter, October 1988

[M. Thalmann85] N. Magnenat-Thalmann, D. Thalmann, *Computer Animation* : Theory and Practice, Springer Verlag, 1985

[Watt89] A. Watt, *Fundamentals of Three-Dimensional Computer Graphics*, Addison-Wesley Publishing Company, 1989