SÍNTESE DE IMAGEM POR *RAY TRACING* DE OBJECTOS CRIADOS POR EXTRUSÃO APLICAÇÃO - TEXTO 3D

Paulo Sérgio Almeida
Universidade do Minho / INESC

Pedro Castro Borges
Universidade Católica Portuguesa / INESC

António Costa
Inst. Sup. Engenharia do Porto / INESC

Fernando Nunes Ferreira
Faculdade de Engenharia U.P. / INESC

Sumário
Neste trabalho é descrito um método apropriado para a síntese de imagem por *ray tracing* de uma classe de objectos definidos procedimentalmente.
A abordagem tem em consideração a eficiência de cálculo, a qualidade de imagem produzida, a utilidade da classe de objectos suportada na descrição de cenas e a adequação, em particular, à produção de imagem envolvendo texto.
Como exemplo directo e imediato de aplicação desta classe de objectos, foi construída uma biblioteca de rotinas para inclusão num *ray tracer* existente, que permite o tratamento de primitivas de texto. Apresentam-se exemplos de utilização das primitivas criadas, assim como alguns resultados obtidos.

Introdução

No campo da síntese de imagem de elevado realismo destaca-se uma técnica conhecida por *ray tracing*. É então útil que no âmbito desta técnica seja possível visualizar a maior diversidade possível de cenas. Acontece frequentemente os *ray tracers* possuírem poucas primitivas para descrever as cenas. Assim, o utilizador vê-se forçado a descrever um objecto à custa das poucas primitivas existentes como, por exemplo, um conjunto de polígonos. O resultado disso é não só uma maior quantidade de memória necessária para descrever a cena e um maior tempo de cálculo, como, principalmente, a fraca qualidade das imagens produzidas. Um exemplo em que verificamos isso são as cenas que contêm texto. Outro problema encontrado no caso do texto é a dificuldade em obter as descrições dos caracteres em termos de primitivas simples. A solução para todos estes problemas passa por dotar os *ray tracers* de primitivas de mais alto nível que permitam descrever famílias de objectos de grande utilidade e cuja especificidade permita

eficiente uso da memória, pequeno tempo de cálculo e boa qualidade das imagens produzidas. Neste trabalho desenvolveram-se primitivas que, além de terem utilidade geral, são apropriadas para descrever texto.

Extrusão

A família de objectos que decidimos suportar é baseada na extrusão de figuras 2D. Esta consiste em fazer a translação de uma curva plana segundo um vector, geralmente perpendicular ao plano de suporte da curva. Como exemplo, a partir de um rectângulo é possível obter um paralelepípedo, e a partir de uma circunferência é possível obter um cilindro. Esta família de objectos, designados por prismas, é apropriada para a descrição de texto, e também modelo para muitos outros objectos, por exemplo, na área de CAD/CAM.

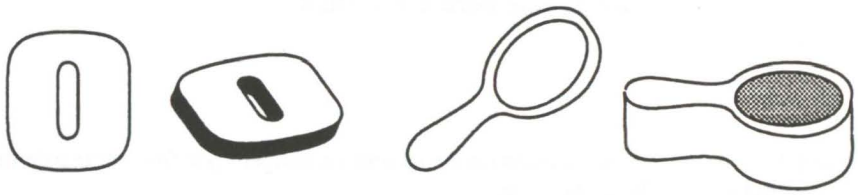


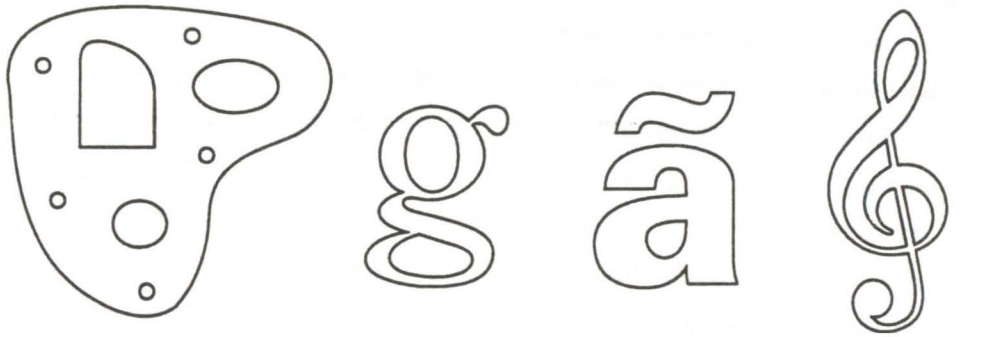
Figura 1 - Exemplos de extrusão.

A eficiência de uma implementação depende fundamentalmente da rotina de intersecção de um raio com os objectos. Este foi um dos motivos que nos levou a escolher os prismas como primitiva, pois existe um algoritmo bastante eficiente para o efeito, apresentado por James T. Kajiya [1]. Esse algoritmo permite que a intersecção raio-objecto seja feita à custa de uma intersecção 2D com a figura da qual é feita a extrusão.

A escolha do tipo de curvas 2D que servem de suporte aos objectos foi diferente. Kajiya baseou-se em *strip trees*; em vez disso utilizam-se segmentos de recta e cúbicas de Bézier. A escolha foi baseada, por um lado, no facto das *strip trees* serem mais apropriadas para descrever curvas digitalizadas e portanto com muitas descontinuidades na normal (o que afecta a qualidade das imagens) e, por outro lado, no facto de muitas *fonts* de texto 2D serem baseadas precisamente em segmentos de recta e cúbicas de Bézier.

Descrição de figuras 2D

Como referido, um prisma é resultado da extrusão de uma figura 2D. Para que um objecto assim gerado tenha sentido físico e fique correctamente definido, deve ser construído da extrusão de um conjunto de linhas fechadas. Exemplos:



9 linhas fechadas 3 linhas fechadas 3 linhas fechadas 4 linhas fechadas

Figura 2 - Exemplos de figuras 2D para extrusão.

Essas linhas fechadas são, por sua vez, constituídas por curvas cúbicas de Bézier e segmentos de recta. A condição das linhas serem fechadas permite construi-las partindo de uma descrição por uma lista de curvas de Bézier, descritas apenas por três pontos, e de segmentos de recta descritos por um único ponto. O ponto de chegada de um curva é sempre o ponto de partida da curva seguinte na descrição.

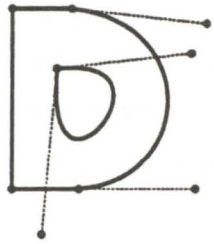


Figura 3

A figura 3, em que estão também marcados os pontos de controlo das cúbicas, pode ser utilizada para gerar um carácter 'D' por extrusão. A sua descrição poderia ser a seguinte: (A estrutura do formato implementado é, de facto, muito semelhante a esta) :

Os cantos inferior e superior esquerdos correspondem aos pontos (0,0) e (0,23).

```
letra 'D'                    ⇐ designação da figura
2                            ⇐ Nº de linhas fechadas que constituem a figura.
4                            ⇐ Nº de curvas que constituem a primeira linha fechada.
segmento_recta  0 0          ⇐ Segmento que parte do ponto (0,0) e termina no ponto
segmento_recta  0 23         ⇐ (0,23) - linha horizontal - onde se inicia novo segmento.
curva_bezier  8 23  25 21  23 0 ⇐ Curva de Bézier partindo de (8,23) e pontos de controlo.
segmento_recta  8 0          ⇐ Final da curva de Bézier e inicio de segmento de recta.
1                            ⇐ Nº de curvas que constituem a segunda linha fechada
curva_bezier  6 15  4 -6  23 17 ⇐ Uma só curva de Bézier que se fecha sobre si própria.
```

Intersecção de um raio com um prisma

Para o cálculo da intersecção de um raio com um prisma, evitando que esta seja feita no domínio 3D, em [1] é proposto o seguinte:

- São calculados os pontos de intersecção do raio com os planos de base e topo (a intersecção, se existir, estará entre esses 2 pontos).

- O raio é projectado no plano de base do prisma segundo a direcção de extrusão.

- São então determinadas as intersecções com a figura, no domínio 2D.

- As intersecções são ordenadas pela distância à origem do raio.

- A lista é percorrida. Parar quando um ponto que corresponda à intersecção com a base ou o topo estiver dentro da figura 2D ou quando uma intersecção com a figura 2D corresponda a um ponto entre os planos de base e topo.

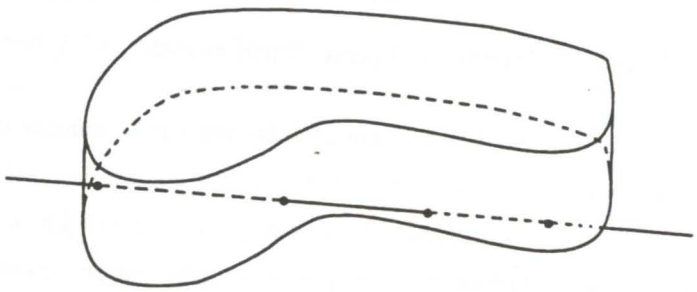


Figura 4 - Intersecção raio objecto

Na figura 4, apresenta-se um exemplo de uma intersecção raio-objecto. A apresentação do método de intersecção pode ser complementada com a figura 5. Nesta são apresentadas duas projecções do objecto e do raio. Indicam-se também as intersecções do raio com a figura de suporte do objecto (no domínio 2D), e com os planos de base e de topo do mesmo.

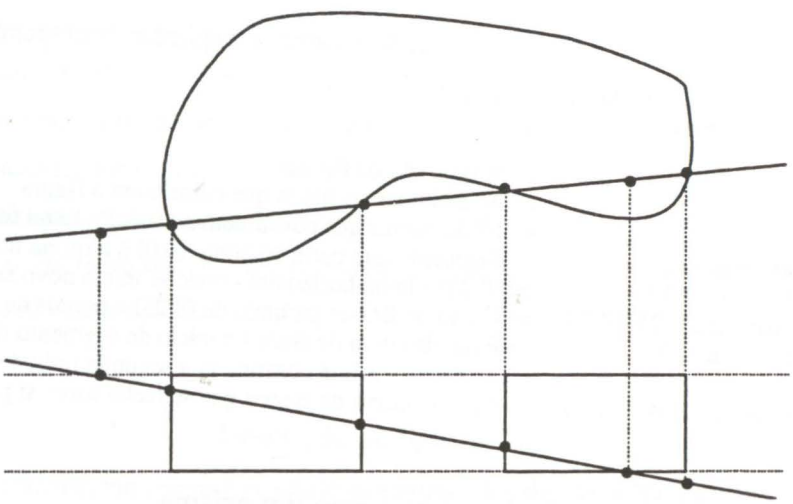


Figura 5 - Intersecção raio-objecto - Vista de cima e corte.

Esta implementação é baseada neste método, sendo no entanto em detalhe algo diferente.

Por exemplo, a direcção de extrusão não é necessariamente perpendicular ao plano de base.

Foi necessário determinar se o raio está a entrar ou a sair do objecto para calcular o sentido da normal. Isto porque não é exigida tal informação na descrição de uma linha 2D.

Para determinar se um ponto está dentro ou fora de uma figura 2D são contadas as intersecções até esse ponto e verificado se o resultado é um número ímpar ou par.

Houve o cuidado de ter em conta vários casos possíveis de intersecção raio-prisma. Alguns casos surgem por ser permitido que o raio possa ter a origem dentro do prisma (tal acontece por exemplo quando objectos transparentes contêm outros objectos).

É registada informação relevante sobre as intersecções com uma figura 2D, tais como quantas intersecções estão antes da origem do raio, quantas estão antes da zona entre os planos, quantas estão depois da origem do raio e entre os planos e, destas, a de menor distância à origem do raio.

Com esta informação é determinada a intersecção raio-prisma (caso exista) tendo em conta o caso apropriado. Sobre a intersecção são registadas várias informações tais como se o raio entra ou sai do do objecto, se é uma intersecção com o lado ou com a base ou topo, e sendo com o lado qual a linha fechada e qual a curva dessa linha que é intersectada. Tal informação é necessária para mais tarde calcular a normal, no caso de o objecto em questão ser o primeiro intersectado pelo raio.

Algoritmos sobre curvas 2D

Cada curva que constitui uma linha 2D pode ser de um de vários tipos possíveis. Por enquanto são suportados segmentos de recta e cúbicas de Bézier. Podem no entanto ser facilmente adicionados outros tipos, bastando para tal fornecer as rotinas de intersecção com uma recta e de cálculo da normal.

No caso dos segmentos de recta ambas as rotinas são triviais. Quanto às cúbicas de Bézier, enquanto o cálculo da normal é bastante simples, tal não acontece no caso da intersecção com uma recta.

Uma cúbica de Bézier pode ser definida parametricamente por

$$x(t) = a_x t^3 + b_x t^2 + c_x t + x_0$$

$$y(t) = a_y t^3 + b_y t^2 + c_y t + y_0$$

com t a variar entre 0 e 1. A projecção do raio por

$$x(d) = v_x d + r_x$$

$$y(d) = v_y d + r_y$$

Existirá uma intersecção raio-curva se forem satisfeitas

$$v_x d + r_x = a_x t^3 + b_x t^2 + c_x t + x_0$$
$$v_y d + r_y = a_y t^3 + b_y t^2 + c_y t + y_0$$

com t entre 0 e 1. Eliminando d obtém-se uma equação do 3º grau. Haverá intersecção se esta equação tiver raízes entre 0 e 1. Nesta implementação foi tomada a opção de determinar a solução exacta, em detrimento de métodos iterativos. Como aquela é computacionalmente pesada, houve a preocupação de determinar de um modo eficiente, se não existe intersecção com a curva, antes de tentar calcular a referida intersecção.

O facto de só terem significado raízes entre 0 e 1 possibilita algumas optimizações. É possível através de cálculos muito simples verificar, em alguns casos, se a equação não tem raízes neste intervalo.

Casos considerados em que não há raízes entre 0 e 1:

- Se todos os coeficientes forem positivos,
- Se os sinais da função (polinómio, nos pontos 0 e 1) não forem diferentes e,
  - se os sinais da primeira derivada forem diferentes e o sinal da função em 0 for igual ao da derivada nesse mesmo ponto, ou
  - se os sinais da primeira derivada forem iguais e acontecer o mesmo para a segunda derivada.

Esta análise resulta de considerações geométricas e produz testes extremamente simples (computacionalmente). Em alguns exemplos que envolveram um total de alguns milhões de cálculos de raízes, a percentagem de vezes em que foi determinado expeditamente que nenhuma intersecção existia, variou entre 31 e 48 por cento.

Aplicação - Texto 3D

Devido à importância que o texto pode ter em muitas imagens (como exemplo, ver a publicidade existente que utiliza imagem sintetizada) criaram-se facilidades que permitem tratar texto, ou de uma forma mais geral caracteres e conjuntos de caracteres (*fonts*). Criaram-se facilidades de nível superior, que permitem descrever, seleccionar e sintetizar imagens com caracteres de modo adequado e eficiente.

Note-se que não se perde generalidade:

- Quaisquer objectos desta classe de prismas podem ser tratados, desde que descritos no formato adequado, muito simples, pelas primitivas criadas para texto, pelo que o que se descreve para texto é extensível a todos os prismas, embora alguns conceitos, como o de *font*, possam perder sentido.

- Um carácter ou uma cadeia de caracteres é um objecto que, como outro qualquer objecto suportado pelo *ray tracer*, pode ser manipulado por diversas operações, como mudança de coordenadas, escalamento, operações CSG, etc. Este encapsulamento permite múltiplas possibilidades de composição e não limita extensões do *ray tracer*. Em termos de utilização de memória, houve a preocupação de instâncias diferentes do mesmo objecto (a menos de posição, orientação e factor de escala) não terem a descrição das curvas replicada.

Descrição dos ficheiros de *fonts*

A descrição de famílias de objectos, de caracteres neste caso, encontra-se em ficheiros de *font* (*font files*). Esses ficheiros são ficheiros de texto e têm uma estrutura muito simples, que pode ser utilizada para criar conversores adequados de formatos nos quais existam disponíveis *fonts* públicas 2D.

Para cada carácter o ficheiro inclui nome e descrição semelhante à apresentada na secção - Descrição de figuras 2D. Podem existir múltiplos ficheiros de famílias de objectos ou caracteres.

Exemplo 1

ficheiro Font_exemplo.ppf: ⇐ Nome do ficheiro

```
[...]                   # Outros caracteres
/at                     # designação do carácter '@', precedida de slash.
2                       # Nº de linhas fechadas
13                      # Nº de curvas da primeira linha fechada.
c .345931 .233333 .237597 .129167 .0875974 .191 # Na descrição das
c .141764 .408333 .195931 .595833 .362597 .65   # curvas '1' indica
l .433431 .545833                               # segmento de recta e
l .450097 .5875                                 # 'c' significa uma
l .529264 .5875                                 # curva cúbica de
c .425097 .266667 .395931 .1625 .608431 .270833 # Bézier. Ver secção:
c .600097 .445833 .600097 .570833 .495931 .6833 #   Descrição de
c .354264 .691667 .187597 .695833 .0417641 .554 #   figuras 2D.
c .0459307 .383333 .0625974 .0833333 .450097 -0.0125
l .600097 .241667
c .645931 .241667 .520931 -0.0666667 -0.00823593 .00416667
c .0 .408333 .00843074 .766667 .487597 .841667
c .620931 .570833 .733431 .35 .479264 .0666667

4       # Segunda linha fechada do carácter /at, composta por 4 curvas.
c .212597 .241667 .254264 .1875 .337597 .266667 # Cúbica de Bézier
l .337597 .279167                               # segmento de recta
c .408431 .483333 .433431 .5625 .329264 .6125   # Cúbica de Bézier
c .266764 .491667 .204264 .375 .187597 .258333  # Cúbica de Bézier

/A              # Outro carácter, sempre que possível utilizam-se
2               # Designações como definido em [4], para facilitar
3               # o mapeamento. Ver próxima secção.
c .245833 .4125 .316667 .4125 .379167 .4
l .441667 .3375
l .329167 .65
```

```
18
l .229167 .358333
c .158333 .170833 .141667 .120833 .175 .0833333
l .2125 .0833333
l .2125 .0666667
l 0 .0666667
c 0 .0833333 .0333333 .0833333 .0541667 .0916667
l .0708333 .145833
c .258333 .633333 .275 .675 .275 .7125
l .245833 .7125
l .245833 .729167
l .475 .729167
c .475 .7125 .441667 .7125 .441667 .666667
l .45 .641667
c .629167 .1375 .6375 .120833 .654167 .0833333
l .683333 .0833333
l .683333 .0666667
l .45 .0666667
c .45 .0833333 .591667 .1375 .433333 .35
[...]          # Restantes caracteres desta família.
```

Como foi dito podem-se criar ficheiros com a descrição de prismas que não sejam caracteres, ou sequer símbolos. O seguinte exemplo é fragmento de um ficheiro que conteria símbolos musicais:

Exemplo 2

ficheiro Musica.ppf

```
/Clave_sol
4
[...]          # A restante descrição do símbolo da clave de sol.
[...]          # Eventualmente outros símbolos.
```

Mapeamento de códigos para nomes de caracteres

Cada carácter deve ter uma designação única dentro de um ficheiro de *fonts*. Supondo, por exemplo, que se escolhem as designações referidas em [4], não é de esperar que um utilizador do *ray tracer* seja obrigado a decorar as designações para todos os caracteres que tem à sua disposição directamente no teclado. Por isso é conveniente criar um mapeamento dos códigos dos caracteres nas suas designações, facilitando o trabalho de geração de cenas. Para o exemplo 1, o ficheiro de mapeamento tem o seguinte aspecto:

ficheiro encoding_ascii.ppe

```
[...]
64/at          # O código 64 corresponde ao carácter '@'.
65/A           # O código 65 é mapeado para o carácter 'A'.
[...]
```

Caracteres que não sejam referidos no ficheiro de mapeamento corrente só podem ser acedidos através da designação original.

Utilização de primitivas de texto

Apresentam-se, em seguida, exemplos utilizando alguns comandos criados e incluídos no *ray tracer* disponível [2]. Esses comandos são aplicados ao nível de *string*, como espaçamento entre caracteres, orientação, direcção de extrusão, escalamento segundo as direcções de orientação e de extrusão, de selecção de *font* (de ficheiro de *font*), de selecção de vector de mapeamento e de posicionamento. No entanto ao *ray tracer* cada carácter será passado como um objecto. Tal facilita, por exemplo, a geração de englobamentos para cada carácter, melhorando o desempenho.

Não se impõe quaisquer restrições quanto às direcções de progressão, vertical e de extrusão.

Para o exemplo que temos vindo a designar como Exemplo 1:

ficheiro exemplo1.sff

```
[...]          # Definições de pontos de vista, luzes, superfícies...[6]
objects              # Definição dos objectos
7 2 1 -              # [6]
SPACING 0.1          # Espaçamento entre caracteres [5]
SCALE 3 3 3          # Escalamento segundo cada vector de orientação.
ORIENTATION 1 0 0 0 1 0 0 0 # Vectores de orientação: direcção de
                             # progressão, vertical (up vector [5]) e
                             # direcção de extrusão.
FONT Font_exemplo.ppf        # Selecção de uma família de caracteres.
ENCODING encoding_ascii.ppe  # Selecção de um ficheiro de mapeamento.
AT .8 0 -3 "@A"              # Posicionamento da cadeia de caracteres no
                             # espaço 3D e indicação da mesma.
```

Figura 6 - Resultado do exemplo 1.

Para o exemplo 2 o ficheiro que descreve a cena pode ser o seguinte:

ficheiro exemplo2.sff

```
[...]          # Definições de pontos de vista, luzes, superfícies...[6]
objects        # [6]
7 2 1 -        # [6]
SPACING 0.1    # Espaçamento entre caracteres, neste caso é irrelevante.
ORIENTATION 1 1 0 0 0 1 -1 1 0 # Vectores de orientação: direcção de
                               # progressão, vertical (up vector [5]) e
                               # direcção de extrusão.
```

```
SCALE 3 3 3          # Escalamento segundo a cada vector de orientação.
FONT Musica.ppf      # Selecção de ficheiro com a descrição do objecto.
AT 1 -3 -4 "/Clave_sol/"    # Posicionamento e indicação do objecto
                            # pela designação completa.
```

A imagem sintetizada que se obteve neste exemplo é apresentada à direita.

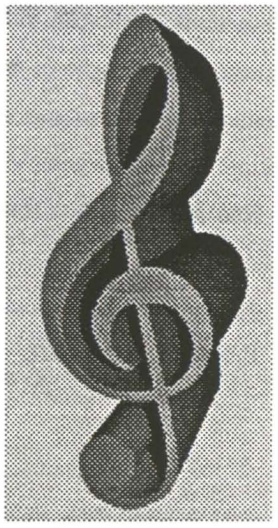


Figura 7

A modificação do escalamento e da orientação permite obter objectos com diferentes profundidades, distorção de objectos para obtenção de efeitos como a italização, extrusão não perpendicular, etc.

Conclusão

Referiu-se a adequação de objectos definidos por extrusão para a síntese de imagem de cenas, como na área de CAD/CAM e, particularmente, as que envolvem texto. Apresentou-se um algoritmo que permite a síntese de imagem por *ray tracing* de objectos definidos por extrusão.

Descreveu-se a estrutura de representação dos objectos e o papel dessa descrição, considerando-se a eficiência de cálculo, a qualidade de imagem produzida, a utilidade do tipo de objectos suportado e a adequação, em particular, à produção de imagem envolvendo texto.

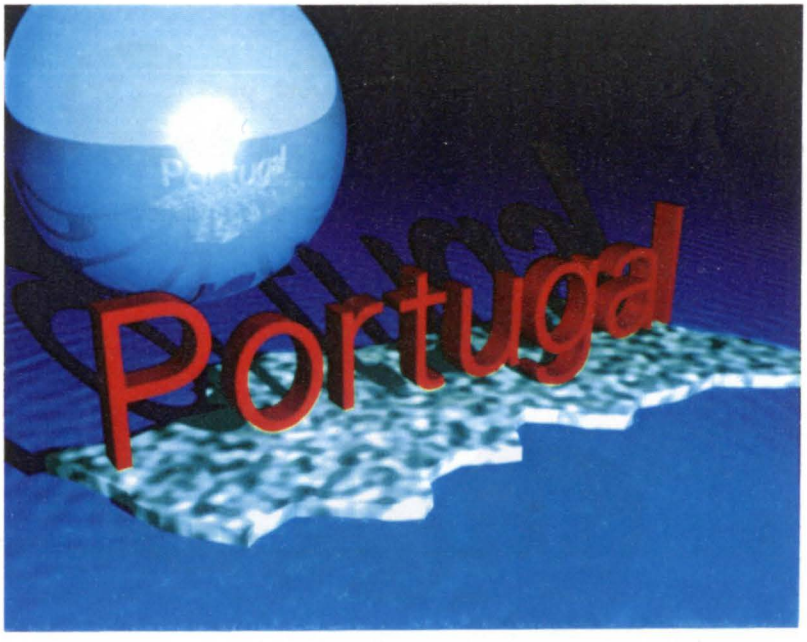
Apresentaram-se ainda exemplos que utilizam uma biblioteca de rotinas incluidas num *ray tracer* existente, em que se exemplifica a utilização de primitivas de texto. Nos utilizadores notou-se, naturalmente, um aumento de velocidade na produção de imagens envolvendo objectos desta classe, uma vez que deixou de ser necessário compor letras de menor qualidade com outras primitivas para cada cena criada. Os resultados obtidos consideram-se bastante satisfatórios.

Referências

[1] James T. Kajiya
New Techniques for Ray Tracing Procedurally Defined Objects
Computer Graphics, Vol. 17, Nº 3, Julho de 1983.

[2] F.Nunes Ferreira, A.Costa, A.Sousa, V.Branco;
3D Graphics Developments and Research at INESC.NORTE,
Computer&Graphics, Vol14 Nº1, 1990, Pergamon Press

Exemplo de utilização de texto.



Caracteres e mapa de Portugal gerados por extrusão

[3] Paulo S. Almeida, Pedro Borges;
Módulo de *Ray Tracer* para Suporte de Objectos Gerados por Extrusão,
MEEC - Sistemas Gráficos, Junho 1992, FEUP / INESC

[4] Adobe Systems Incorporated;
Postscript Language Reference Manual - second edition,
Addison Wesley

[5] T.L.J. Howard, W.T. Hewitt, R.J. Hubbold, K.M. Wyrwas;
A Practical Introduction to PHIGS and PHIGHS PLUS
Addison Wesley

[6] António Costa;
Manuais de *rtrace* e *scn2sff*.
INESC NORTE.