GIRTD - UMA ARQUITETURA PARALELA PARA A GERAÇÃO DE IMAGENS REALÍSTICAS TRIDIMENSIONAIS EM TEMPO REAL

L.H. Loss, P.O.A. Navaux
Curso de Pós-Graduação em Ciência da Computação - CPGCC
DI - CPD - UFRGS
C.P. 1501, 90.001 - Porto Alegre-RS - Brasil

RESUMO

Uma grande dificuldade encontrada no desenvolvimento de estações gráficas de exibição a tempo real de imagens tridimensionais realísticas, para modelagem interativas de sólidos por exemplo, é o cálculo e geração de pixel para a formação destas imagens. Neste artigo é apresentado uma proposta de arquitetura paralela para o cálculo de pixels para a geração de imagens de objetos sólidos representados por B-rep (representação por fronteiras poligonais), com remoção de partes ocultas por algorítmo Z-buffer, sombreamento contínuo de Gouraud e antialiasing.

ABSTRACT

A great difficulty found in real time realistic 3-D graphic worktations development, is the pixel calculus for its image generation. Thi paper presents a parallel architecture for pixel rendering of boundary-represented solid images, with Z-buffer algorithm for hidden surface removal, Gouraud continous shading, and antialiasing.

1. INTRODUÇÃO

A computação gráfica tem respondido por um importante papel na nossa sociedade. Desenvolvida com o intuito de facilitar e dinamizar a comunicação entre homem e máquina, ela está presente em quase todas as áreas que fazem uso das facilidades oferecidas pelos computadores.

Por computação gráfica (C.G.) entende-se qualquer criação e manipulação de imagens gráficas por intermédio do computador. É uma das áreas das ciências da computação que mais tem evoluido desde a década de 60. Atualmente existem aplicações que vão desde as de mais baixa escala como vídeo-games, até sistemas altamente inteligentes de processamento de imagens, no reconhecimento de padrões, na reconstituição de imagens tridimensionais de partes do corpo humano para análise clínicas, e nos sistemas de CAD (computer aided design), entre outros.

A grande maioria destas aplicações depende de técnicas de exibição de imagens tridimensionais sobre uma tela plana de computador. Uma técnica especial, que vem sendo utilizada intensamente, é o emprego de arquiteturas paralelas de processamento para a geração, em alta velocidade, de imagens gráficas de objetos tridimensionais. O emprego desta técnica vem viabilizar aquelas aplicações em que a imagem tem que ser gerada instantaneamente, como em sistemas interativos rápidos, ou aplicações a tempo real, como simuladores de vôo e animação.

Este trabalho tem por objetivo apresentar o desenvolvimento de um projeto em processamento paralelo para a geração de imagens realísticas em sistemas gráficos baseados na exibição por tubos de varredura, denominado GIRTD (Gerador de Imagens Realísticas Tridimensionais).

2. DEFINIÇÃO DO GIRTD

Como parâmetros iniciais para a determinação do projeto foram estabelecidas aplicações para as quais o GIRTD será empregado. Poderão ser desde simuladores de vôo, até modelagem gráfica e interativa de sólidos com imagens realísticas, e com a possibilidade de um desempenho de tempo real, para aplicações com animação.

Com essas imposições preliminares, algumas conclusões relativas ao porte do projeto podem ser obtidas:

- um grande volume de memória de vídeo será necessário para abarcar tanto a alta resolução exigida, usualmente 1 Mpixel (1024x1024) ou mais, para estas aplicações, quanto a razoável quantidade de planos de memória, ou bits por pixel, necessários para o grande número de cores usadas pelo realismo;
- devido às exigências de tempo real dever-se-á adotar estruturas especiais de acesso a memória capazes de vencer o gargalo de tempo de acesso à estas;
- como unidades de processamento para o cálculo dos pixel é descartado o uso de processadores comerciais de uso geral.

A geração de pixels faz uso de algorítmos não muito complexos (tal como a remoção de faces ocultas por Z-buffer) mas que, para atender a demanda de pixels exigida pela geração a tempo real, têm que ser executados a uma velocidade que só pode ser atingida por processamento paralelo. Uma geração de imagens a uma velocidade de 20 Mpixels/s (50ns/pixel) é considerada

razoável para a animação a tempo real ([1],[2]) em uma tela de 1 Mpixel, portanto, este valor foi uma meta a ser alcançada no projeto.

Para conseguir-se um sombreamento das imagens sem quebras na coloração é necessário ter a disposição uma tal gama de cores de maneira que, numa escala crescente da variação de intensidades, cores vizinhas praticamente se confundam. O uso de 8 bits (256 níveis) para a representação de cada uma das componentes de cor (R, G e B) tem demonstrado na prática ser capaz de atender essa exigência. Este é outro parâmetro adotado neste projeto, ou seja, um total de 24 bits (3x8) ou 16 milhões de cores possíveis.

Este equipamento deverá servir a um computador hospedeiro. Neste computador rodará a interação com o usuário bem como alguns utilitários especiais que necessitem realimentar o usuário com informações gráficas de imagens dos objetos envolvidos neste aplicativo. Os dados são emitidos pelo computador numa forma de descrição dos objetos sólidos ou planos, A técnica aqui usada para descrever os objetos sólidos emitidos pelo hospedeiro é a B-rep poliédrico (representação por fronteiras poligonais).

3. O PROJETO GIRTD

O GIRTD gera imagens manipulando apenas os polígonos que descrevem os sólidos. Estes polígonos deverão passar por uma seqüência de processamentos, como a figura 1 mostra.

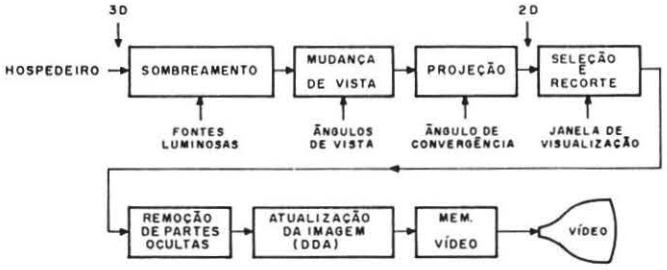

Figura 1. Pipeline para a geração de imagem baseada em B-rep

O primeiro estágio deste pipeline é responsável pelo sombreamento da imagem. Aí são aplicados modelos matemáticos parametrizados para a síntese de iluminação, sombreamento e transparência dos objetos. Esses cálculos usualmente são aplicados somente nos vértices dos polígonos para posterior aproximação linear do restante da área dos polígonos [3].

O segundo bloco executa a mudança de vista da imagem. Essa operação depende de qual ponto de vista o usuário deseja que a imagem seja exibida na tela gráfica. As coordenadas dos vértices dos poliedros são trocadas por outras relativas a um novo sistema de eixos coordenados associado aos ângulos de visualização da imagem.

O próximo bloco é de projeção da imagem. As coordenadas dos vértices são alteradas novamente para dar uma noção de profundidade à imagem. Os objetos mais ao fundo (longe do observador) ficarão proporcionalmente menores.

No estágio de seleção e recorte os polígonos que, após a mudança de vista, caírem fora da área de visão serão descartados enquanto que outros que estiverem parcialmente visíveis na tela terão que ser recortados, ou seja, transformados em outros polígonos que representam as porções visíveis destes.

Por fim, antes da atualização da memória imagem (ou memória de vídeo), deve-se remover total ou parcialmente os polígonos sobrepostos por outros, para uma correta exibição da imagem, através de testes de profundidade.

O último estágio do pipeline gráfico é responsável pelo preenchimento rápido, na memória de vídeo, dos polígonos finais que contribuem para a imagem.

No GIRTD a remoção de partes ocultas é feita por algorítmo Z-buffer, durante a atualização do conteúdo da memória de vídeo, portanto, os dois últimos estágios do pipeline são um só. O algarítmo Z-buffer é característico por possuir uma extensão à memória de vídeo, chamada de "Z-buffer". Esta armazena, para cada pixel o valor de sua profundidade Z, para auxiliar na detecção de pixels ocultos e que não devem ser atualizados na memória. Assim a remoção é feita sem a necessidade de pré-ordenação e, como é feita independentemente para cada pixel dos polígonos, problemas como faces interpenetrantes são aqui resolvidos.

Neste processo de atualização da memória também são executadas algumas tarefas desejáveis para o realismo, o sombreamento e o antialiasing. O sombreamento, no caso de B-rep, pode ser dado por interpolação linear das informações de cor e luminosidade calculadas nos vértices dos polígonos, conforme Gouraud [3]. Isso implica em que cada pixel poderá ter uma cor levemente diferente da do seu vizinho, necessitando portanto de processamento individual para cada pixel. Como a interpolação é linear, podemos empregar métodos incrementais para a obtenção dos atributos dos pixels. Estes métodos tem a grande vantagem de necessitar apenas de operações de soma e subtração para a obtenção da cor e luminosidade de um pixel vizinho. Esse processamento é feito por um circuito comumente chamado de DDA (Digital Differential Analyzer). O antialiasing é uma técnica para a redução do efeito serrilhado nas bordas das figuras através da atribuição de cores intermediárias aos pixels próximos a estas bordas, que será descrita mais adiante.

Um DDA é capaz de gerar somente um pixel de cada vez. Para que este seja capaz de gerar pixel a uma taxa de 20 Mpixels/s (50ns/pixel), que é a taxa exigida para as aplicações de animação a que se propõe a GIRTD, deve-se empregar arquiteturas paralelas.

Neste trabalho é apresentado uma proposta de arquitetura especial de processamento paralelo para a implementação de um DDA de alto desempenho, com uma taxa de pelo menos 50ns/pixel. Esta deve fazer uso da técnica de Z-buffer para a remoção das faces ocultas, e deve preencher polígonos com sombreamento contínuo e antialiasing.

3.1. O algorítmo DDA

A função básica do DDA é desenhar polígonos na memória de imagem, pixel a pixel, com características de coloração necessárias para o realismo. Para isso o DDA deverá receber, do seu estágio anterior no pipeline, parâmetros que desencadearão esse preenchimento da memória da maneira mais rápida possível. Esses parâmetros deverão ser bastante detalhados para que o DDA não perca tempo com operações afora o preenchimento da imagem.

No DDA o preenchimento se dará controlado por um algorítmo, escrito em micro-código, que percorra os pixels da imagem, determinando a cor desses e se esses devem ou não ser atualizados. Para isso é desejável que o algorítmo do DDA seja capaz de:

- identificas os limites de preenchimento dos polígonos
- andar em passos unitários de pixel evitando-se assim a necessidade de operações de multiplicação para a determinação dos atributos dos pixels longíncuos
- executar o anti-aliasing nas bordas dos polígonos
- usar aritmética inteira ou de ponto fixo
- identificar, através do Z-buffer, pixels sobrepostos

Por definição as informações de cor, R, G e B, e a profindidade Z serão aproximadas linearmente ao longo da área do polígono. O seu comportamento é então expresso por

$$P = a.x + b.y + c \qquad \text{ou}$$
$$P = Px'.x + Py'.y + Po$$

$$a = px' = dP/dX$$
$$b = Py' = dP/dy$$
$$c = Po = P(x=0,y=0)$$

x,y = endereço dos pixels na tela e P ∈ (R,G,B,Z)

O DDA então recebe do preparador, como parâmetros de entrada, os valores iniciais Pi e suas derivadas px' e Py'. Pi é o valor da iteração inicial, conforme o endereço da partida do pixel xi,yi, ou seja,

$$Pi = P(x=xi,y=yi)$$
$$= Px'.xi + Py'.yi + Po$$

O algorítmo DDA é mostrado na figura 2.

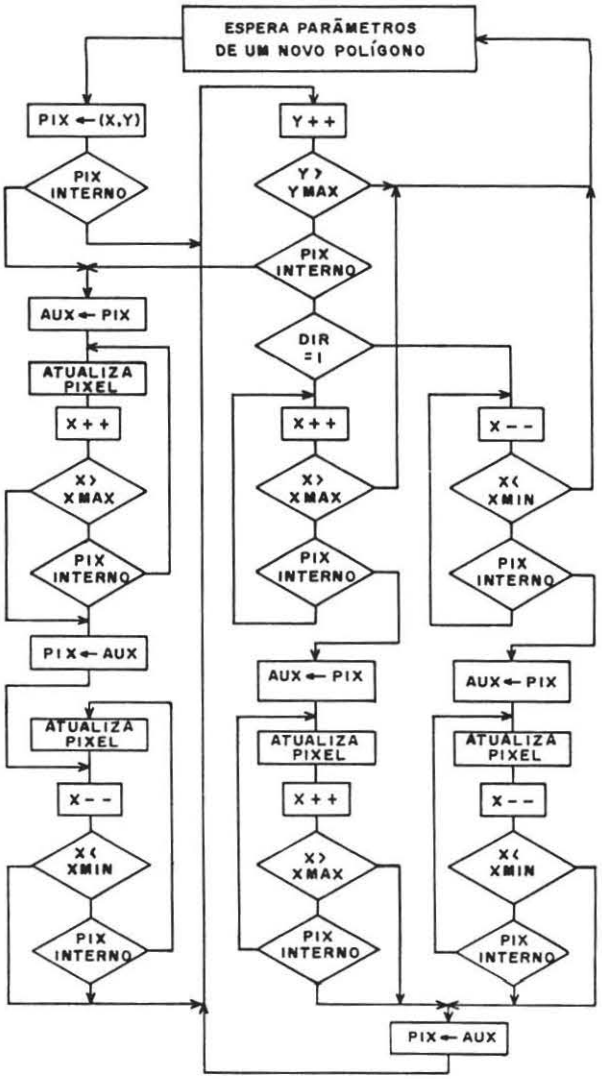

Figura 2. Algorítmo DDA

O algorítmo parte do pixel inicial xi,yi com sua profundidade inicial Zi e cores Ri, Gi e Bi. Para cada pixel visitado o algorítmo decide se este deve ou não ser atualizado na memória, dependendo da comparação de Z com Zb (Z buffer). A visita ao pixel seguinte é sempre feita por saltos unitários nas direções x ou y. Em virtude disso os valores P são atualizados por somas ou subtrações com as respectivas derivadas em relação a x ou y.

Se for implementado com uma série de ULAs (Unidades Lógicas e Aritméticas) simples, para a determinação incremental de R, G, B, X, Y e Z, operando em paralelo, pode-se atingir a taxa de pixels desejada (<50ns/pixel).

A figura 3 apresenta esta estrutura de hardware, contendo seis elementos de processamento - EPs - cada um contendo uma ULA e registradores.

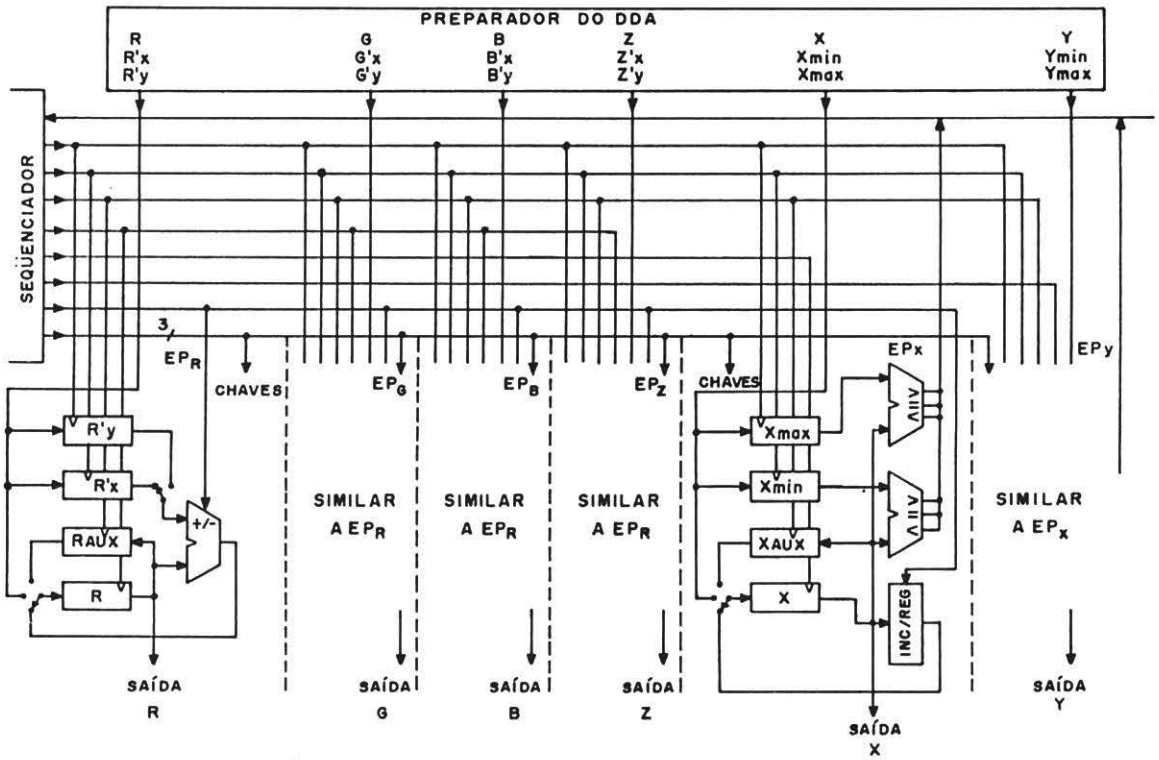

Figura 3. Bloco operacional do gerador de pixel

Em cada um dos EPs responsáveis pela determinação de R, G, B e Z encontra-se uma ULA, para as somas e subtrações das derivadas, e quatro registradores, um para armazenar o resultado calculado e de saída, dois contendo as duas derivadas, e um auxiliar para armazenamento de pixel postergado. As EPs X e Y são diferentes pois suas ULAs só calculam incremento e decremento, em paralelo com uma comparação de magnitude. Também possuem um registrador para o resultado de saída, e outro auxiliar para o endereço de pixel postergado, enquanto que os outros dois são os limites max e min.

O sequenciador possui o algorítmo DDA implementado em micro-código. O controle das EPs é dado por algumas linhas de controle emitidas pelo sequenciador. Uma delas avisa se a operação é de soma ou subtração, nas ULAs R, G, B e Z, e se é incremento ou decremento em X ou Y. Também informa às EPs X e Y se a comparação de magnitude é com o registrador max ou o min. Um outro sinal indica se a operação é na direção Y ou X . Este faz com que sejam selecionadas as devidas derivadas em R, G, B e Z, ao mesmo tempo que habilita ou a EP X ou a EP y para trabalho.

3.2. Detecção das fronteiras dos polígonos

O algorítmo DDA necessida de um mecanismo para detectar se o pixel visitado pertence ou não ao polígono.

A maneira pela qual isso é feito é através da propagação incremental, ao longo da geração de pixels, dos valores de distância entre o pixel sendo visitado e as retas formantes do polígono. Enquanto todas essas distâncias tiverem o mesmo sinal (todas positivas ou todas negativas) o pixel visitado estará dentro do polígono, caso contrário, estará fora (válido somente para polígonos convexos).

A distância de um ponto a uma reta é dada pelo módulo de:

$$D = \frac{A.x + B.y + C}{\sqrt{(A^2 + B^2)}}$$

onde x,y é um ponto, e A,B,C definem a orienção e posição da reta. A indicação de ponto interno é dada pela coincidência de todos os sinais de D iguais.

O cálculo incremental das distâncias pode ser feito da mesma forma que para R, G, B e Z. Para cada distância D das arestas há a necessidade de um EP, com uma ULA e quatro registradores, para execução em paralelo. Com isso o número de arestas permitido para cada polígono é limitado pelo número de EPs disponíveis para o cálculo das distâncias. Para resolver esse inconviniente optou-se por limitar os polígonos a triângulos e por intercalar-se um estágio no pipeline gráfico da figura 1, responsável pela decomposição dos polígonos em triângulos [4]. Essa decomposição resolve também o problema dos polígonos convexos. A partir desse estágio - o triangularizador - toda manipulação será sobre triângulos.

A figura 4 apresenta o circuíto das três EPs a serem adicionadas às outras já apresentadas (fig. 3). Apresenta também a detecção de pixel interno, escolhido pela identificação dos sinais positivos das distâncias.

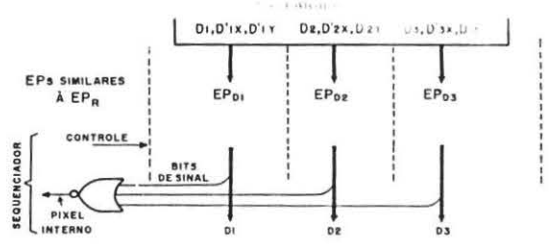

Figura4. Circuito de detecção de bordas

3.3. Detecção do "DIR"

O algorítmo DDA aqui proposto necessita de uma orientação, quando o pixel cai fora do triângulo por um salto positivo em Y, devendo decidir que direção tomar até encontrar novamente o interior do triângulo, se pela direita ou pela esquerda. Esta direção é identificada pelo sinal da derivada da distância, em relação a x, que se tornou negativa com a saída do pixel para fora do triângulo. Um sinal positivo indica que essa distância tornar-se-á novamente positiva se caminharmos no sentido crescente de x; quando negativo indica que deve-se tomar o sentido decrescente de x. Esta detecção pode ser feita por um circuito lógico simples, que avalia os bits de sinal dos três registradores que contém as distâncias e dos três registradores das derivadas das distâncias em relação a x.

3.4. Antialiasing

O antialiasing é uma técnica vastamente empregada em sistemas gráficos para a minimização do efeito serrilhado nas bordas dos objetos representados em uma imagem definida por matriz de pixels. Esta técnica é ilustrada na figura 5.

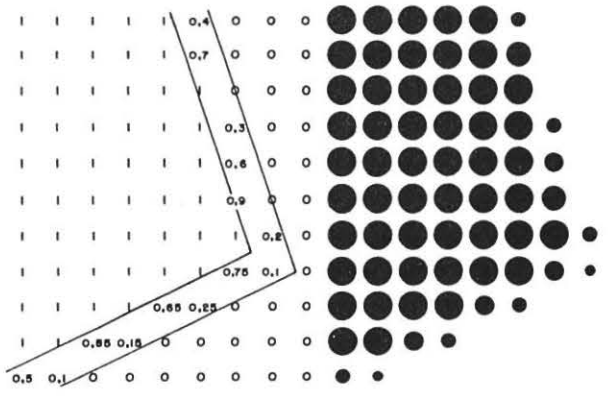

Figura 5. Antialiasing

A intensidade de cor atribuída a cada pixel é proporcional a área deste atingida pelo interior da figura. Esta proporção é representada, na figura 5 à esquerda, pelos valores centrados nos pixels. Na direita, a representação destes valores é pela proporção da área dos círculos escuros, dando uma noção visual do antialiasing. O antialiasing é executado no GIRTD pela variação gradativa da cor do pixel, nas bordas dos triângulos [5]. Ao invés de uma troca brusca da cor do pixel, conforme o seu centro caia dentro ou fora do triângulo em preenchimento, muda-se sua cor gradualmente a medida que seu centro se afasta das bordas do triângulo. A dosagem de cores varia linearmente de 100% a 0% a medida que o centro do pixel está afastado do triângulo de uma distância de 0 a 1 pixel, respectivamente. Para isto o GIRTD tira proveito da disponibilidade das distâncias D1, D2 e D3, de cada aresta. Estas são operadas de maneira a obter-se um fator de dosagem entre a cor de fundo e a cor interna do triângulo, conforme a expressão

CorPixel = n.Triângulo + (1-n). Fundo

onde n=1 no interior do triangulo e decai linearmente de 1 até 0 a medida que o centro do pixel está afastado de 0 a 1 pixels da borda do triângulo.

3.5. Transparência

A transparência é um efeito desejado quando se quer representar objetos translúcidos. Pode-se obtê-la fazendo-se com que a cor calculada para cada pixel seja combinada com a cor já presente na posição correspondente do pixel na memória de vídeo, segundo um fator T de dosagem. Este fator T varia de 0 a 1, indicando transparência total ou opacidade total respectivamente, e irá interagir junto com o fator de antialiasing para a determinação da cor do pixel.

T também pode ser gerado incrementalmente para uma transparência gradual, de acordo com os ângulos de transmissão. Para isso um EP adicional se faz necessário, com os mesmos quatro registradores e uma ULA, semlhante aos outros EPs, para processar a equação.

$$T = To + Tx'.x + Ty'.y.$$

4. COMENTÁRIOS FINAIS

A arquitetura paralela aqui apresentada possui dez unidades de processamento que atuam em paralelo para a obtenção de pixels para preenchimento de triângulo, com sombreamento contínuo, remoção de pixels ocultos, transparências e antialiasing, para a geração de imagens realísticas de sólidos a tempo real. Esta deverá calcular 20 Mpixels/s em média, ou seja 50ns/pixel. Para isto um clock de 30ns é necessário para compensar a ocorrência de estados inválidos na máquina do DDA.

Com todos os 10 EPs trabalhando em paralelo a uma velocidade de 33Mips cada um (30ns/instrução) a arquitetura do DDA pode atingir um desempenho teórico de 330Mips.

Para um correto atendimento da demanda por triângulos do DDA, o pipeline deverá manter um fluxo de dados capaz de suportar esta demanda. Os estudos dos algorítmos envolvidos no

pipeline indicam que unidades processadoras com 0,5-5 Mflops de desempenho, em cada um desses estágios, são capazes de suprir esta demanda.

O preenchimento da memória de vídeo a uma taxa de 50ns/pixel não pode ser feito dentro dos ciclos comuns de memórias de vídeo. Portanto uma memória cache rápida de 16 pixels foi projetada para compatibilizar esta velocidade com os ciclos de acesso das memórias de vídeo. Para tal a memória de vídeo é particionada em 16 bancos de 64k pixels cada (total 1M pixel ou 1024x1024). Com isso o acesso à memória de vídeo é dado em grupos de 16 pixels para permitir que as transferências entre os 16 pixels da memória cache e a memória de vídeo sejam feitas em um único ciclo. Para um melhor aproveitamento da cache estes 16 pixels são pixels vizinhos dispostos horizontalmente na imagem. A disposição horizontal se deve ao fato de que o algorítmo percorre os pixels, em deslocamentos unitários, preferencialmente na horizontal. Com isto garante-se uma maior utilização da memória cache antes que tenha que trocar seus dados com a memória de vídeo.

REFERÊNCIAS

[1] Burgoon, D., "Pipelined Graphics Engine Speeds 3D Image Control", Electronic Design, Rochelle Park 35(17), pag.143-150, Julho de 1987.

[2] Fujimoto, A., Perrot, C. G., Iwata, K., "A 3-D Graphics Display System with Deep Buffer and Pipeline Processor" IEEE Computer Graphics & Aplications, Los Alamos, 4(6), pag. 11-23, Julho de 1984.

[3] Gouraud, H., "Continuos Shading of Curved Surface", IEEE Transactions on Computers, New York, C20(6), pag. 623-629, Junho de 1971.

[4] Wordenweber, B., "Surface Triangulation for Picture Production", IEEE Computer Graphics & Aplications, Los Alamos, 3(8), pag. 45-51, Novembro de 1983.

[5] Fujimoto, A. & Iwata, K., "Jag-Free Image on Raster Displays IEEE Computer Graphics & Aplications, Los Alamos, 3(9):26-34, Dec. 1983.